Dic345 XH v196 XcS

Höhere Technische Bundeslehr- und Versuchsanstalt Anichstrasse
Abteilung
Elektronik & Technische Informatik

Ausgeführt im Sommer 2020 von:

Tom Teraflop 2.71aHEL
Susi Summe 3.14bHEL
Raudi Rechenzahn 0.7071aHEL
Berti Bitweiser 3.16cHEL

Projektpartner:
Fa. OOP Ollas Ohne Plan,
Umbriggler Alm

Betreuer/Betreuerin:

Lenni Leftshift PhD
Dipl.-Ing.^{se} Kristel Schönfrau
Dr Kuno Kernel

Innsbruck, am 01.Apr.0815

Abgabevermerk:

Datum:
01.Apr.18

Betreuer/in:
Schnapsi Murgs

Kompetenzbereiche
- 'Entwurf digitaler Systeme'
- 'Computerarchitekturen'
- 'Embedded Systems'
- 'Digitale Signalverarbeitung'
- ('Realisierung und Test von Systemem')

'Handlungsdimensionen'/270716:
- (AB) Wiedergeben und *Verstehen* (S.5) ↓
- (AB) Wiedergeben und *Beschreiben* (S.7)
- (C) Anwenden
- (D) Analysieren
- (E) Entwickeln

Themenbereiche KU'2223
- 1-Grundschaltungen
- 2-Programmierbare logische Schaltungen
- 3-Computerarchitekturen
- 4-Embedded Systems
- 5-Digitale Signalverarbeitung

'Anforderungsbereiche'/260416:
- I Reproduktion
- II Reorganisation & Transfer
- III Reflexion & Problemlösung

'Erwartungshorizonte'/040218: s.u.
'Kompetenzmodule': 5 .. 10
'Bereiche': ♯
'Operatoren': ♯
'Jahresfragen': ♯

Inhaltsverzeichnis

LinX:

http://ww1.microchip.com/downloads/en/ devicedoc/atmel-0856-avr-instruction-set-manual.pdf ... AVR Instruction Set neueres PDF
https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-Instruction-Set-Manual-DS40002198A.pdf ... AVR Instruction Set PDF
https://en.wikipedia.org/wiki/Atmel_AVR_ instruction_set ... AVR Instruction Set Webseite
http://www.getchip.net/wp-content/uploads/ATTiny13.pdf ... Datenblatt TN13
zuig/TN13A-VaO1.pdf ... Datenblatt TN13A auf zuig/TN13A-VaO1.pdf

Teil I
1-Digitale Grundlagen

Digitalschaltungen - Lowlevel

s. "https://de.wikipedia.org/wiki/Kategorie:Digitale_Schaltungstechnik"
s. Böhmer Kap.15 "Digitale Verknüpfungs- und Speicherschaltungen" S.236 (Aufl.16), Interfacing 246/247

1.0.1 Pegel: Logic Specs

Familie	V_{IL} max	V_{IH} min	V_{OL} max	V_{OH} min
TTL 74xx	0.8	2.0	0.4	2.4
LVTTL 3V3	0.8	2.0	0.4	2.4
TTL 74H	0.8	2.0	0.4	2.4
TTL 74S	0.8	2.0	0.5	2.5
TTL 74LS	0.8	2.0	0.5	2.7
TTL 74ALS	0.8	2.0	0.5	2.5
TTL 74F	0.8	2.0	0.5	2.5
74HC	1.35	3.15	0.26	3.98 @4mA
74HCT	0.8	2.0	0.26	3.98 @4mA
CMOS+5Vcc	1.5	3.5	0.5 @1mA	4.44 @1mA
→4001B@5Vcc	1.5	3.5	0.05 @1μA	4.95 @1μA
			0.4 @0.36mA	4.5 @0.36mA
LOCMOS HEF4000	1.5	3.5	0.05	4.95 @1μA
CMOS 2V5	0.7	1.7	0.2	2.3
CMOS 1V8	0.7	1.17	0.45	1.2
LVDS			1.0	1.4 @3.5mA/100Ω
ECL	-1.4	-1.2		

1.0.2 Pegel: Standard TTL

	High	Low
Ausgang:	≥ 2.4V @-400uA	≤ 0.4V @16mA
Eingang:	≥ 2.0V @40uA	≤ 0.8V @-1.6mA

Lies: "Bei $I_a = -400\mu A$ muss der Ausgang noch mindestens $U_a \geq 2.4V$ liefern"
(Strom zählt immer ins IC hinein)

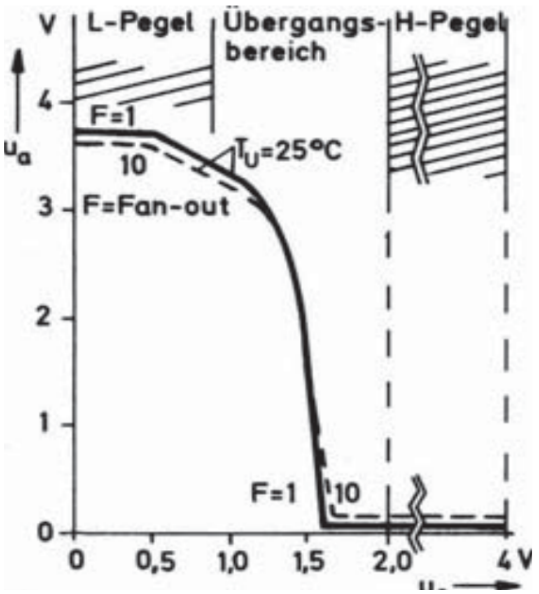
Fan-Out:
Die *Anzahl* von Eingängen (derselben Logikserie zB. 74HC, s.u.), die ein Ausgang maximal ansteuern kann (ohne Specs zu verletzen), nennt man ***fan-out*** $= {}^{I_{a,max}}/_{I_{e,max}}$.

Für *'standart TTL 7400'* ist *fan-out = 10* vorgegeben.
Das erklärt, warum in den Specs die Ausgangsströme 10-fach höher sind als die Eingangsströme.

1.0.3 Pegel: Communication

	High	Low
IIC:	≥ 0.7 V_{DD}	≤ 0.3 V_{DD}
CAN(diff.)	≥ 3.5V @3mA	≤ 1.5V @3mA
USB1(diff.)	2.8. . . +3.3V	-2.8. . . -3.3V
USB2(diff.)	2.8. . . +3.3V	-2.8. . . -3.3V
USB3(diff.)	+17.78mA (0.4V)±10%	-17.78mA (-0.4V ±10%)
D+ or D- line	≤ 0.3V	≥ 2.8 − 3.0V
RS232-RX:	≤ -3.0V	≥ +3.0V
RS232-TX:	≤ -5.0V @3kΩ	≥ +5.0V @3kΩ

1.0.4 TTL Inverter Übertragungskennlinie



1.0.5 rise/fall times

t_{rise}, t_{fall} von 10% bis 90% der Endpegel

[...] Anstiegs- und Abfallzeiten beschreiben in der Digitaltechnik und bei Schalttransistoren die beim Umschaltvorgang charakteristischen Zeiten, in denen das Signal nicht mehr den alten und noch nicht den neuen definierten Logikpegel („0“ bzw. „1“) bzw. Schaltzustand innehat. Siehe auch Flankensteilheit.

In der Digitaltechnik werden hierbei meistens die im ungünstigsten Fall (worst case) garantierten Zeiten genannt. Sie beschreiben die Zeit, die ein Signal (beispielsweise in einem Computerprozessor) zum sicheren Umschalten zwischen den beiden binären Zuständen benötigt. Die für ein Bauteil spezifizierten Anstiegs- und Abfallzeiten sind oft keine Messwerte, sondern ein Merkmal des Bausteins bzw. der betrachteten Logikfamilie bei bestimmten Parametern (Betriebsspannung, Temperatur), das durch das Design bzw. den Herstellungsprozess gesichert ist.

Sämtliche Digitaltechnik basiert letztlich auf analog arbeitenden Schaltungselementen, die für die Bearbeitung digitaler Signale optimiert sind. Dabei ist zu beachten, dass man bei den digitalen Signalen „1“ bzw. „0“ zwar „Strom ein“ bzw. „Strom 0“ unterstellt, in der Praxis aber meistens mit Spannungs- und Strompegeln gearbeitet wird, die ungleich dieser Idealwerte sind. Weiterhin ist das Verhalten der Schaltung bei Zustandswechsel meistens unsymmetrisch, weshalb Anstiegs- und Abfallzeit dann unterschiedlich lang sind.

Für ein Logiksignal in einer Schaltung ist es notwendig, Schwellenwerte festzulegen. Für logisch „0“ wird bei TTL beispielsweise ein zulässiger Bereich von 0. . . 0,4 V festgelegt und als logisch „1“ ein Bereich von 2. . . 5 V. Die tatsächliche Schaltschwelle der Logikbausteine (ca. 1,4 V) liegt im „verbotenen Bereich“ zwischen diesen beiden Werten und daher im Bereich der Anstiegs- und Abfallzeit. Daraus ergibt sich ein besonders schnelles Durchlaufen des verbotenen Bereiches, was für viele Logikbausteine wichtig und oft auch mit einer minimal zulässigen Spannungsanstiegs-Geschwindigkeit spezifiziert ist.

Anstiegs- und Abfall-Zeit beschreiben die Zeitintervalle, in denen dieser undefinierte „verbotene“ Zustand während des Umschaltens auftritt. Wichtig ist das insbesondere bei flankengetriggerten Schaltungen, d. h. Schaltungen, die auf die Änderung des Signals reagieren (z. B. flankengetriggertes Flipflop). Andernfalls kann es zu Fehlfunktionen kommen; siehe Race condition.

Um eine Sicherheit gegen Störungen zu erhalten, gelten die erlaubten Pegel auch für die Ausgänge dieser Bausteine. Beispielsweise werden für einen Gatterausgang in der klassischen TTL-Technik max. 0,4 V für „0“ und min. 2,4 V für „1“ garantiert.

Sehr kurze Anstiegs- und Abfallzeiten im Signal bedeuten auch, dass im Spektrum des Signals sehr hohe Frequenzanteile vorhanden sind, die zur Aussendung (Abstrahlung) von elektromagnetischen Wellen führen. Durch diese Störsignale können andere Schaltungsteile in ihrer Funktion beeinflusst werden. Um die elektromagnetische Verträglichkeit sicherzustellen, werden deshalb die Ausgänge von Digital- und Treiberschaltungen so ausgelegt, dass die Anstiegs- und Abfallzeiten nur so kurz wie unbedingt nötig sind. Dazu wird die Flankensteilheit (slew rate) des Ausgangstreibers begrenzt.

In der Messtechnik sowie zur Charakterisierung analoger und digitaler Schaltungen werden zur Spezifikation der Zeiten meistens die Werte von 10 % bzw. 90 % des Schaltpegels bzw. Sollsignales definiert. Auch Schalttransistoren und andere leistungselektronische Bauteile werden damit charakterisiert. Analoge Verstärker, Schaltverstärker, Leuchtdioden, Laser, Photodioden und Fototransistoren werden ebenfalls durch Anstiegs- und Abfallzeiten charakterisiert, die sie als Antwort auf eine Sprungfunktion liefern. Wenn ein exponentieller Verlauf $U(t) = e^{\frac{t}{\tau}}$ angenommen wird, beträgt die

Anstiegs- und Abfallzeit für einen Tiefpass 1. Ordnung oder vergleichbare Systeme jeweils
$t_{fall} = t(10\%) - t(90\%) = 2.3\tau - 0.1\tau = 2.2\tau$ *[...]*
(https://de.wikipedia.org/wiki/Anstiegs-_und_Abfallzeit 14Feb17)

1.0.6 Propagation Delay

[...] Als Gatterlaufzeit (engl. propagation delay) bezeichnet man in der Digital- und Elektrotechnik die für die verwendeten Bauteile (Gatter) charakteristische Laufzeit von Signalen von einem Eingang zu einem Ausgang des Gatters. Diese Zeit wird mit t_P oder t_{PD} abgekürzt, oder richtungsabhängig mit t_{PLH} für steigende Flanken bzw. t_{PHL} für fallende Flanken am Ausgang.

Gatter als diskrete Bauelemente haben Laufzeiten im Bereich weniger Nanosekunden (ns) bis über 100 ns, siehe Logikfamilie. Die Zeiten werden von den Herstellern in den Datenblättern für bestimmte Bedingungen angegeben, meist für eine dem Fan-Out entsprechende kapazitive Last und eine hohe Temperatur, manchmal getrennt für verschiedene Versorgungsspannungen. Meist sind je drei Werte angegeben, Minimalwert, typischer Wert und Maximalwert. Bei den Grenzen handelt es sich um Zusicherungen des Herstellers. Sie sind großzügig bemessen, da die Hersteller höchstens Stichproben messen.

Präzise gemessen werden Laufzeiten über die Frequenz eines Ringoszillator. Die Zeiten streuen innerhalb einer Charge und unterscheiden sich manchmal deutlich von Charge zu Charge und zwischen Herstellern.

Innerhalb integrierter Schaltungen sind Gatter für kleinere Spannungen und Ströme ausgelegt, die Eingänge haben eine geringere Kapazität, weil ohne Schutzbeschaltung. Daher sind die Gatterlaufzeiten viel kleiner. Für die auf Gatterebene konfigurierbaren FPGAs liegen sie im Bereich einiger 10 bis 100 Picosekunden, innerhalb von Prozessorkernen teilweise weit unter einer Picosekunde.

Zu Gatterlaufzeiten kommen Laufzeiten auf Signalwegen hinzu. Konkrete Methoden zur Ermittlung von Gesamtlaufzeiten durch beliebige Netzwerke stellt die sog. Laufzeittoleranzrechnung bereit. Die Folge einer Nichtbeachtung der Gatterlaufzeiten können Timingverletzungen und Glitches in der Schaltung sein. [...]
(https://de.wikipedia.org/wiki/Gatterlaufzeit 14Feb17)

TBD. (bedeutet 'to be done' ≡ in Arbeit)
Impedanz...

1.0.7 Glitch

kurzes, zwischenzeitliches Umschalten auf *'falsche'* Logik-Pegel (in Digitalsystemen; zB. aufgrund Laufzeitunterschieden, typisch etwa bei asynchronen Johnson-Zählern, Arithmetikbaugruppen (zB. Addierwerk), asynchronen Schaltwerken udgl.) Synchronisierung mit einem Taktsignal dient als Gegenmaßnahme.

[...] In der Elektronik bezeichnet man mit Glitch eine ***kurzzeitige Falschaussage in logischen Schaltungen*** *und temporäre Verfälschung einer booleschen Funktion.*

Diese tritt auf, weil die Signallaufzeiten in den einzelnen Gattern niemals vollkommen gleich sind. *Diese Verfälschung wird daher auch als Race Condition bezeichnet.* *Die Anfälligkeit für Glitches steigt mit der Komplexität, der Geschwindigkeitserhöhung und der Verkleinerung der Schaltungen, kann aber auch bereits bei sehr einfachen Schaltungen vorhanden sein. Sie stellen ein wesentliches Problem bei der Entwicklung moderner elektronischer Schaltungen und schneller Mikroprozessoren dar, das war aber auch schon bei der älteren elektromechanischen Relaistechnik so. [...]*
(https://de.wikipedia.org/wiki/Glitch_(Elektronik) 14Feb17)

1.0.8 Spike

kurzes **Störsignal**, oft ausserhalb der Logikpegel, aufgrund Nebensprechens oder Stromversorgungs-/bedarfs-schwankungen (Schaltvorgänge) im Zusammenwirken mit parasitären Reaktanzen des mechanischen Aufbaues
In electrical engineering, spikes are fast, short duration electrical transients in voltage (voltage spikes), current (current spikes), or transferred energy (energy spikes) in an electrical circuit. Fast, short duration electrical transients (overvoltages) in the electric potential of a circuit are typically caused by

- Lightning strikes
- Power outages
- Tripped circuit breakers
- Short circuits
- Power transitions in other large equipment on the same power line

- Malfunctions caused by the power company
- Electromagnetic pulses (EMP) with electromagnetic energy distributed typically up to the 100 kHz and 1 MHz frequency range.
- Inductive spikes ('inductive kickback')

In the design of critical infrastructure and military hardware, one concern is of pulses produced by nuclear explosions, whose nuclear electromagnetic pulses distribute large energies in frequencies from 1 kHz into the gigahertz range through the atmosphere.

The effect of a voltage spike is to produce a corresponding increase in current (current spike). However some voltage spikes may be created by current sources. Voltage would increase as necessary so that a constant current will flow. Current from a discharging inductor is one example. For sensitive electronics, excessive current can flow if this voltage spike exceeds a material's breakdown voltage, or if it causes avalanche breakdown. In semiconductor junctions, excessive electric current may destroy or severely weaken that device. An avalanche diode, transient voltage suppression diode, transil, varistor, overvoltage crowbar, or a range of other overvoltage protective devices can divert (shunt) this transient current thereby minimizing voltage. While generally referred to as a voltage spike, the phenomenon in question is actually an energy spike, in that it is measured not in volts but in joules; a transient response defined by a mathematical product of voltage, current, and time. Voltage spikes may be created by a rapid buildup or decay of a magnetic field, which may induce energy into the associated circuit. However voltage spikes can also have more mundane causes such as a fault in a transformer or higher-voltage (primary circuit) power wires falling onto lower-voltage (secondary circuit) power wires as a result of accident or storm damage. Voltage spikes may be longitudinal (common) mode or metallic (normal or differential) mode. Some equipment damage from surges and spikes can be prevented by use of surge protection equipment. Each type of spike requires selective use of protective equipment. For example a common mode voltage spike may not even be detected by a protector installed for normal mode transients. An uninterrupted voltage increase that lasts more than a few seconds is usually called a "voltage surge" rather than a spike. These are usually caused by malfunctions of the electric power distribution system.
(https://en.wikipedia.org/wiki/Voltage_spike 14Feb17)

1.1 Minimierung Boole'scher Ausdrücke

1.1.1 boolean expression

1.1.2 Wahrheitstabelle truth table

1.1.3 KV DG Karnaugh-Veitch

1.1.4 Minterm, Maxterm-, Ringsummen–Methode

es gelten

↪ $(A \wedge B) \vee (A \wedge \neg B) = A$

↪ $(A \vee B) \wedge (A \vee \neg B) = A$

↪ DeMorgan'sche Regel $\overline{A \vee B} = \overline{A} \wedge \overline{B}$

↪ DeMorgan'sche Regel $\overline{A \vee B} = \overline{A} \wedge \overline{B}$

Umformen der vorgegebenen Gleichungen mit den Theoremen für boolesche Gleichungen bis eine Minimalform erzielt wird. Hauptsächlich kommen hier die o. a. Kürzungsregel und die Theoreme von De Morgan zum Einsatz. Das Verfahren erfordern oftmals Probieren und intuitives Vorgehen und sind in der Regel nur sinnvoll, wenn die Möglichkeit der Minimierung relativ offensichtlich ist.

1.1.5 Quine-McCluskey method

- Von Willard Van Orman Quine und Edward J. McCluskey.
- Methode, um Boolesche Funktionen zu minimieren.
- Der Kern des Verfahrens wurde bereits von Quine vollständig beschrieben.
- Verfeinerungen von McCluskey: Praktische algorithmische Durchführbarkeit.
- Das Verfahren findet immer eine minimale Lösung.

Grundprinzip:
Unterscheiden sich zwei durch Disjunktion verknüpfte Konjunktionsterme nur durch die Negation einer einzigen Variablen, so kann man diese beiden Terme verschmelzen und dabei die betreffende Variable entfernen.

$(a\&b\&c\& - d)V(a\&b\& - c\& - d)$

ergibt

$a\&b\& - d$

1.2 Entwurf von Automaten

Bei einer Entwicklung ist meist eine Aufgabe gegeben und hierzu soll eine Schaltung entworfen werden. Der Ablauf beim Entwurf umfasst die folgenden Schritte:

1.Spezifikation des Verhaltens
2.Aufstellen der Zustandsfolgetabelle
3.Minimierung der Zustände
4.Codierung der Zustände
5.Aufstellen der Ansteuerungstabelle
6.Logikminimierung

1.2.1 Minimierung der Zustände

Eine Vereinfachung ist möglich, wenn äquivalente (also gleichbedeutende) Zustände zusammengefasst werden können. Zwei Zustände sind äquivalent, wenn für alle Eingangskombinationen die Folgezustände gleich oder äquivalent und die Ausgangswerte gleich sind.

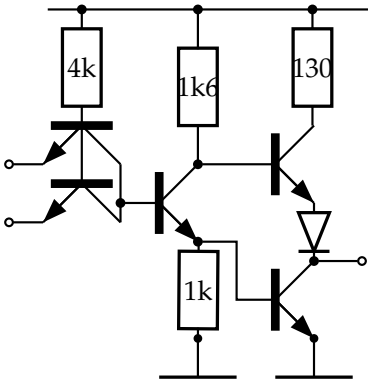
1.3 Simulation

s.Hwe3-5, Dic3-5

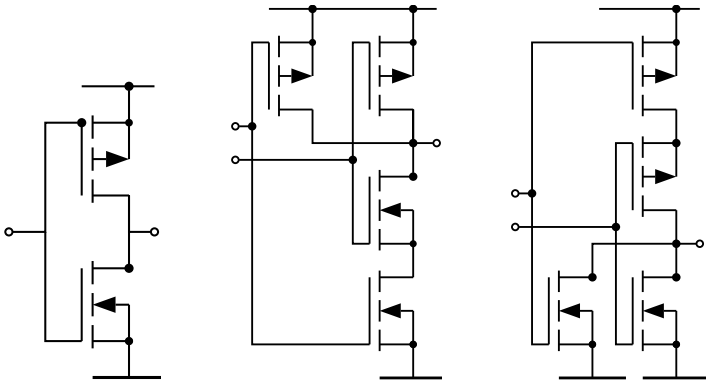
→ Anwendung 'Altium'
→ SPICE net list u. 'ngspice'
→ AVR-Simulator (AVR-Studio)
→ Fsst-XH1 u. -2: 'FE-Simulation', 'Raketenstart' s.Kap.31.4, S.467
→ 'numerische Integration' Dic4a'1516 s.Kap31, S.466
→ 'diskrete (Logistik-) Computersimulation'
→ Pseudo-Zufallszahlen, PseudoRandomNoise s.Dic5 Kap.23, S.395
→ 'Prozessor Simulator' Diplomarbeit P.Achrainer'1617

1.4 TTL NAND

Vcc	Vcc	0	DC 5.0
R1	Vcc	R1u	4k
Q1a	Q2B R1u	In1	NPN
Q1b	Q2B R1u	In2	NPN
R2	Vcc	Q2C	1.6k
Q2	Q2C Q2B	Q2E	NPN
R3	Q2E	0	1k
R4	Vcc	Q3C	130
Q3	Q3C Q2C	Q3E	NPN
D1	Q3E	Out	SiDiode
Q4	Out Q2E	0	NPN



1.5 CMOS Inv, Nand, Nor



keine passiven Bauteile!
Die machen alles mit
der Kanalgeometrie
und der Dotierung.

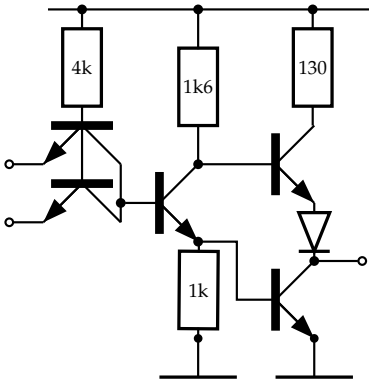
1.6 Logikfamilien

Die bekanntesten Logik-'Familien' sind:

TTL 74'er	74xx	als 74HC.., 74AC.., 74LV.., 74F.. aktuell
		ICs waren wie Gräber am Friedhof *'in Reih und Glied'* auf grossen Platinen plaziert, weswegen man von einem *'TTL-Friedhof'* sprach :^))
CMOS 4000'er	40xx	für langsame(1MHz), (asynchrone) Schaltwerke ideal sehr sparsam
ECL 10'er		im Aussterben begriffen
ECL 100'er		-"- , sind/waren die schnellsten
TTL 93'er		ausgestorben

'Familen' nennt man sie, weil dieselbe Schaltungs- und Fertigungstechnik, Betriebsspannung, I/O-Spezifikation, Pinbelegung (weitgehend: zB 7400:4fachNand, 7401:4fachNandOpenCollector, 7408:4fachAND, 74132:4fachNandSchmitttrigger, 7402:4fachNor, 7432:4fachOR, 7486:4fachExor) bei verschiedener Gatterfunktion angewendet sind.

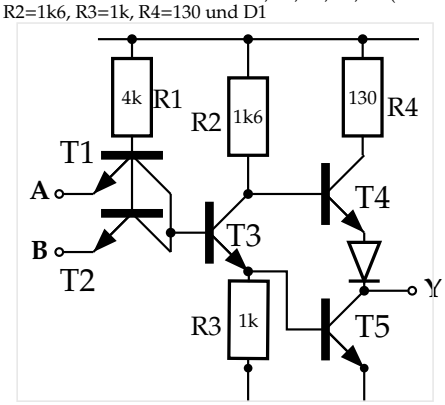
1.7 TTL NAND dimensionieren



Bauteilmäßig ist ja die Schaltung fertig dimensioniert - iXH möchte die Knotenspannungen und Kantenströme (Stromstärken entlang der Kichhoff'schen Maschen-Kanten) bei spezifikationskonformem In- und Output

⊕ den Doppel-Emitter-Eingangstransistor der originalen TTL-NAND Schaltung gibt es nicht (als Einzelbauelement) zu kaufen. Auch ist er im *'Böhmer'* nicht beschrieben und in keiner Bauteilbibliothek der gängigen Spielzeugsimulatoren enthalten (Du musst ggf. mit dem richtigen 'SPICE' IC-Simulator arbeiten und selber Dotierungsprofile anlegen)
→ wir nehmen eine gleichwertige Ersatzschaltung mit zwei an B und C parallelgeschalteten NPN.

⊕ Die Bauteile nennen wir nun T1, T2, T3, T4, T5 (alle NPN), R1=4k, R2=1k6, R3=1k, R4=130 und D1



⊕ innerhalb eines IC lassen sich die Transistoren mit der Zonenbreite verschieden *'kräftig'* gestalten, mit der Collectorzonenlänge verschieden spannungsfest, mit der Dotierung verschieden vorgespannt und verstärkend ua. anfertigen

⊕ Der Emitter von T1 bildet den **Eingang 'A'**, der Emitter von T2 den Eingang 'B'. Am Collector von T5 ist der **Ausgang 'Y'**.

⊕ Wir nehmen (vereinfachend) die NPN-Basis-Emitterspannung im leitenden Zustand näherungsweise als $U_{BE} = 700mV$, die NPN-Collector-Emitter-Sättigungsspannung näherungsweise als $U_{CE_{sat}} = 200mV$ anstelle einer Ungleichung $U_{CE_{sat}} < 200mV$.

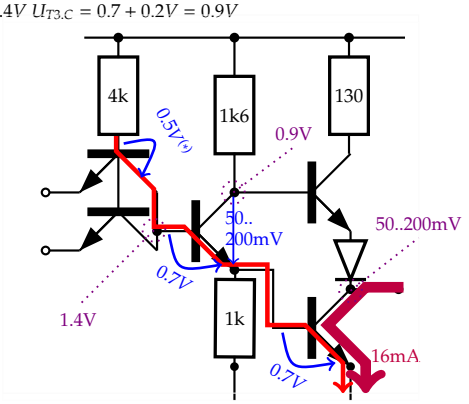
⊕ Ein NPN ist *'gesättigt'*, wenn die Steigerung des Basisstroms keine weitere Steigerung des Collectorstroms (und keine weitere Senkung der Collector-Emitter-Spannung) bewirkt. Bei *'näherungsweisem Sättigungszustand'* spricht man von einer *'quasi-Sättigung'* ($500mV < U_{CE_{sat}} < 1V$)

⊕ Es sind

U_{IL}	`voltage input low`	`<0.8V`
U_{IH}	`voltage input high`	`>2.0V`
I_{IL}	`current input low`	`-1.6mA`
I_{IH}	`current input high`	`40uA`
U_{OL}	`voltage output low`	`<0.4V`
U_{OH}	`voltage output high`	`>2.4V`
I_{OL}	`current output low`	`16mA`
I_{OH}	`current output high`	`-400uA`

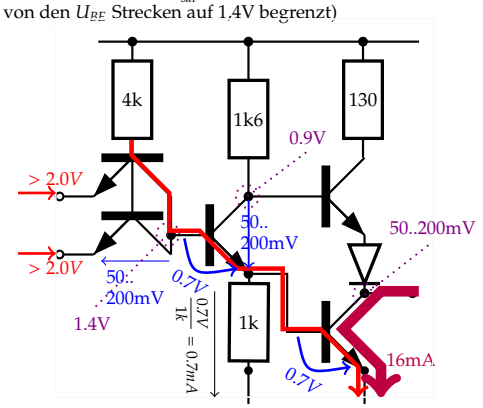
Zählrichtung von Stromstärken ist (außer bei Zweipolen) immer ins Bauteil hinein.

⊕ bei -wie gezeichnet- fehlendem Input fließt in T1+T2 kein Emitterstrom $I_E = 0$, die Basis ist über R4=4k mit Vcc verbunden → der Basisstrom fließt wieder beim Collector hinaus und bildet so auch den Basisstrom für T3, ⇒ T3 wird leitend, T3.Emitterstrom speist T5.Basis → T5 wird leitend. $U_{T5.Basis} = 0.7V$ $U_{T3.Basis} = 0.7 + 0.7 =$ $1.4V$ $U_{T3.C} = 0.7 + 0.2V = 0.9V$

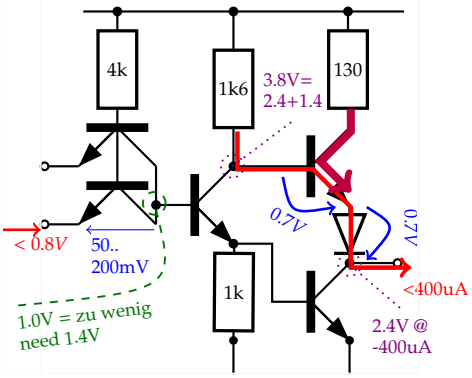


(*) Wegen B≈200 (Stromverstärkung) fallen an der B->C 'junction' bei Ic=1mA 1.4V weniger ab, sodass bereits ab ca. $U_{CE} < 0.46V$ der Sättigungseffekt auftritt, was bekanntlich in der Lowpower-Schottky (LS) Technik ausgenutzt wird

⊕ Bei Input-Hi ($U_{IH} > 2.0V$) ändert sich wenig: Es ergäbe sich T3.U_B = $U_{IH} > 2.0V$+T1.$U_{CE_{sat}}(200mV) > 2.0V+200mV = 2.2V$ (wird aber von den U_{BE} Strecken auf 1.4V begrenzt)



⊕ Bei $U_{IL} < 0.8V$ hingegen:



⊕ von einem sehr hilfreichen Zeitgenossen habi eine Zusatz-Fragenliste erhalten mit lauter Details, die iXH übersehen habe:

1. Auswirkung der Verbindung der zwei Transistoren bzw. Funktion der 2x Emitter-Transistors
   Antwort:
   Im Logik-IC isch des a 'Doppel-Emitter-Transistor'. Da wird die Emitter-Zone aufgespalten, wodurch sich eine sehr gleichmäßige Stromaufteilung ergibt. Den gibts nit als Einzelbauelement zu kaufen und auch nit als Olzium-Bauelement zu simulieren. (Sowas gibts auch bei Mehrfach-Collector-Transistoren für Stromspiegel und bei Mehrfach-Source-Zonen mit sehr ungleicher Aufteilung für Strom-Mess-Zwecke).
   Wir simulieren und realisieren das mit zwei parallelgeschalteten Einzeltransistoren; die Funktion ist dieselbe, lediglich ist die Stromaufteilung weniger präzis, was aber wurscht is
2. Was sind die Kirchhoff'schen Maschen-Kanten
   Antwort:
   In der Graphen-Theorie ist eine 'Kante' die Verbindung zweier 'Knoten'. Die Kirchhoff'schen Regeln sind 'Summe aller Ströme in einem Knoten'

und 'Summe aller Spannungen in einer Masche (Schleife)'. Somit reden wir vom Strom entlang einer Kante Kante in einer Kirchhoff-Masche, einfacher gesagt vom Strom zwischen zwei Knoten.

3. Was ist mit Zonenbreite und Zonenlänge gemeint?
   Antwort:
   Da sind p- bzw. n-dotierte Collector-, Emitter-, Drain- und Source-Zonen gemeint. Die Grenze von einem Collector zu einer Basis zB. ist nicht unendlich scharf machbar, sondern verschwimmt, weshalb wir von 'Zonen' reden.
4. Was ist mit 50..200mV gemeint?
   Antwort:
   Maximal leitende ('gesättigte') Bipolartransistoren haben zwischen C und E die sog. '$U_{CE,sat}$'-Spannung von ca. 50..200mV, je nach Collectorstrom I_C.
5. Wie kommt man auf Ubc=0,5 bei T1?
   Antwort:
   Sei die GLeichstromverstärkung B=100, also I_C= 100*I_B — Für 1/100-stel des Stromes reichen 120mV weniger Spannung, des ergibt dann U_{BC} = 0,62-0,12 = 0,5V.
   Dann hat ma
   $U_{BE} = 0,62V$
   $U_{BC} = 0,5V$
6. Wofür is die Diode?
   Antwort:
   Ohne diese Diode wäre $U_A = U_E(T4)50mV$ und $U_B(T4) = 0.9V$ wodurch T4 leitend würde, obwohl der Ausgang LOW zu sein hat. Jetzt tun die dem T4 diese Diode zum Emitter in Serie, damit die 0.9V Basisspannung ganz klar zu wenig ist und T4 sicher gesperrt bleibt.
7. Was genau muss ich da (vor)rechnen können?
   → alle Spannungen und Ströme (Knotenspannungen und Stromstärken zwischen den Knoten ≡ Kirchhoff'sche Maschen-Kanten)
8. Und sollte bei NAND nicht wenn A und B 0 ist bei Q 1 rauskommen? Weil bei der Grafik is nur bei A=0 B=1 am Ausgang 1
   Antwort:
   a NAND b == NOT(a AND b)

a	b	Q
0	0	1
0	1	1
1	0	1
1	1	0

Deine Verknüpfung (NOT a) AND (NOT b) wäre ein NOR (nach DeMorgan: NOT(NOTa AND NOT b) == a OR b)

a	b	Q
0	0	1
0	1	0
1	0	0
1	1	0

Um mir Zeichen-Arbeit zu sparen, habi nit alle identischen Fälle gezeichnet (wäre auch irritierend)

9. Was ist eine "mäandrierende Leitungsführung"?
   Antwort:
   'Mäander' sind nebeneinanderliegemde 'S'-Schlingen, ein 'zickzack', bei Leiterbahnen zB. zwecks Längen-Ausgleich.
10. Es wird doch 'bipolar' eh nichts mehr gefertigt, also zu wos lerni den alten Kas?
    Antwort:
    Abgesehen davon, dass Lernen als Gehirntraining immer gut is, haben diese Entwicklungen aus den 1960-iger Jahren nach wie vor Auswirkungen bis heute: Nachdem sich die 7400-IC-Serie ordentlich durchgesetzt hatte, kamen **kompatible** Verbesserungen mit den S-, L-, LS-, F-, HC-, HCT-, ALS-, AHC- usw. Serien mit Vcc=5V, dann mit Vcc=3.3V, 2.5V, 1.8V, 1.65V . . . 0.8V. Immer wurde die **Kompatibilität** zur bisherigen Entwicklung im Auge behalten. Zuletzt wurden auch Einzel-Gatter-ICs (zB. 74LVC1G00: 1 Stk Dual-NAND in Low-Voltage-CMOS-1.65V) entwickelt — sie vereinfachen Placement und Routing und erlauben so kürzere Zuleitungen. Heute herrschen CMOS-Schaltungen mit Vcc=3.3V und den Hi/Lo-Umschaltpegeln 1/3 Vcc (1.1V) bzw. 2/3 Vcc (2.2V) vor, aber die 5V leben zB. im 'Arduino' oder im 'USB' munter weiter; der 2.2V-Pegel (3.3V-Hi) is ziemlich TTL (ein 5V/3.3V-TTL-Hi:2.4V überschreitet 3.3V-CMOS-Hi:2.2V und 5V/3.3V-TTL-Lo:0.8V unterschreitet 3.3V-CMOS-Lo:1.1V), der Cypress-PSOC5LP kommt mit einem eigenen 3.3V-zu-5V-Wandler-FPGA uvam.

1.8 LVDS - Low Voltage Differential Signalling

1.9 LVDS low voltage differential signalling

- differenzielle Spannungspegel
- relativ geringe Spannungspegel (englisch low voltage)
- die Signale werden mit einer Konstantstromquelle erzeugt

- Year created 1994
- 655 Mbit/s (up to 1-3 Gbit/s possible)
- TIA/EIA-644
- Serial ATA (SATA), PCI Express (PCIe), FireWire, HyperTransport,
- Videoschnittstellen wie DisplayPort
- Feldbusse wie SpaceWire und RapidIO
- Digital Visual Interface (DVI) oder HDMI basieren auf dem prinzipiell ähnlichen Transition-Minimized Differential Signaling (TMDS).
- Spannungshub 350 mV Unterschied
- absolute Spannung gegen Masse: 1200 mV
- Treiberseite: Konstantstromquelle 3,5 mA
- Empfängerseite: Abschlusswiderstand von 100 Ohm
- Empfänger: Spannungsänderung von +350 mV zu −350 mV
- Gegentaktstörung: bis zu 1000 mV tolerant

(frei nach https://de.wikipedia.org/wiki/Low_Voltage_Differential_Signaling 02Jan'22)

Bei dem englischen Begriff Low Voltage Differential Signaling (LVDS) handelt es sich um einen Schnittstellen-Standard für Hochgeschwindigkeits-Datenübertragung. LVDS ist standardisiert nach ANSI/TIA/EIA-644-1995. Es beschreibt die physische Schicht, nicht die höheren darauf aufsetzenden Protokoll-Schichten. Wichtige physikalische Merkmale sind:

- differenzielle Spannungspegel
- relativ geringe Spannungspegel (englisch low voltage)

– die Signale werden mit einer Konstantstromquelle erzeugt

Die hauptsächlichen Anwendungen liegen bei seriellen Hochgeschwindigkeitsübertragungen mit einigen GBit/s. Typische Anwendungen von LVDS sind Serial ATA (SATA), PCI Express (PCIe), FireWire, HyperTransport, Videoschnittstellen wie DisplayPort und auch Feldbusse wie SpaceWire und RapidIO. Weiterhin basieren meist proprietäre digitale Schnittstellen von Flüssigkristallbildschirmmodulen im Embedded-Anwendungsbereichen und bei Laptops, wo das Display in das Gehäuse fix integriert ist, auf LVDS. Digitale Videoschnittstellen zwischen PC und einem externen Monitor wie Digital Visual Interface (DVI) oder HDMI hingegen basieren auf der physikalischen Ebene auf dem ähnlichen aber zu LVDS unterschiedlichen Transition-Minimized Differential Signaling (TMDS).

LVDS arbeitet mit einem Spannungshub von 350 mV. Differenzielle Signalübertragung bedeutet, dass zwei Leitungen verwendet werden und die Differenz der Spannungen für den Logikzustand ausschlaggebend ist. Bei LVDS beträgt der Unterschied 350 mV, während die absolute Spannung gegen Masse bei etwa 1200 mV liegt. Ein Logikwechsel wird durch entgegengesetzte Änderungen der Spannung auf beiden Leitungen erzeugt. Dies wird als symmetrische Signalübertragung bezeichnet. Die Änderungen der Signalpegel auf den Einzelleitern haben immer entgegengesetztes Vorzeichen.

Logikpegel:

Vee	VOL	VOH	Vcc	VCMO
GND	1,0 V	1,4 V	2,5–3,3 V	1,2 V

Auf der Treiberseite erzeugt eine Konstantstromquelle einen Strom von 3,5 mA. Dieser wird abhängig vom Logikpegel des Eingangssignals zwischen den beiden Signalleitungen umgeschaltet. Dabei wird die jeweils andere Leitung mit dem Nullpegel verbunden. Auf Empfängerseite fließt der Strom durch einen Abschlusswiderstand von 100 Ohm. Dieser Wert entspricht dem Wellenwiderstand der Leitung. Dadurch wird eine Reflexion zurück zur Signalquelle weitgehend vermieden. Der Signalstrom erzeugt im Empfänger eine Spannungsänderung von +350 mV zu −350 mV und umgekehrt.

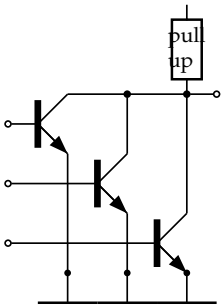
Die niedrigen Spannungspegel bewirken, dass LVDS-Signale gegenüber elektromagnetischen Störungen empfindlich sind. Ein geeignetes Layout kann der Störempfindlichkeit entgegenwirken. Es empfiehlt sich Hin- und Rückleiter eng beieinander zu führen, oder aber einen Leiter über einer Massefläche derart zu gestalten, dass sich durch die Geometrie und das Dielektrikum der Leiterplatte der Leitungswellenwiderstand einstellt. Durch die geringe Fläche, welche die dicht beieinander geführten Leiter aufspannen, kann auch ein nur geringer magnetischer Fluss eines elektromagnetischen Feldes eine Spannungsdifferenz als Gegentaktstörung auf die Leitung einprägen. Gegenüber Gleichtaktstörungen sind die Empfänger bei einer LVDS-Übertragung bis zu 1000 mV tolerant.

Die eng beieinander geführten Leitungen bewirken auch eine nur geringe Abstrahlung des Gegentakt-Nutzsignals. Trotzdem können sich bei unzweckmäßiger Ausführung einer Schaltung auch entlang einer gut geführten Leitungsanordnung unerwünschte Gleichtaktsignale einstellen, die zur ungewollten Abstrahlung einer elektromagnetischen Welle führen. Diese lassen sich aber durch eine insgesamt EMV-gerechte Gestaltung der Schaltung und nicht nur durch die Leitungsführung beeinflussen. Bei Übertragungsstrecken im GBit/s-Bereich sind Laufzeitunterschiede zwischen den Pfaden und eventuell auch zu anderen LVDS-Kanälen zu vermeiden. Gleiche Leiterlängen sind daher zu einer synchronen Signalübertragung erforderlich. Diese Bedingung kann mit mäandrierenden Leiterzügen erreicht werden. LVDS wird z. B. für Ein- und Ausgangskanäle in FPGAs integriert. Hier geben Hersteller wie Xilinx die Leitungslängen innerhalb der Gehäuse als sogenannte „Flight-Time“ an. Die Bezeichnung suggeriert zwar eine Zeitangabe, es handelt sich aber um die Leitungslängenangabe in Millimeter.

Die maximale Datenrate einer LVDS-Schnittstelle hängt von der Kabelqualität ab. Mit Cat-5-Kabel UTP ist typischerweise eine Leitungslänge von etwa 2 m bei einer Datenrate von 200 MBit/s möglich. Nach dem derzeitigen Stand der Technik liegt die Grenze bei mehreren GBit/s. (aus de.wikipedia.org/wiki/LVDS 28Jan18)

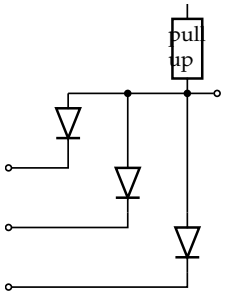
1.10 wired AND, OR

'Manchmal', zB. wenn die Pegel nicht zu Prozessoren oder Logikfamilien passen oder sich deren Einsatz *"nicht lohnt"*, greift man zu *selbstverdrahteten* Logikverknüpfungen mit Pullups, Dioden, Transistoren oder ICs mit *'open Collector'/'open Drain'* Ausgängen (zB. 'ULN2803' oder wie beim IIC-Bus).

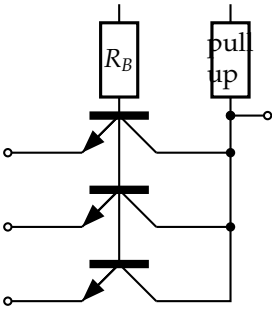


Sagen wir, ich müsste in einem 24V- oder 48V- Zwischenkreis einer Automatisierungstechnik eine 3-fache 'NOR' Verknüpfung realisieren, dann ginge das wohl am einfachsten so:

Ein 'AND' so:



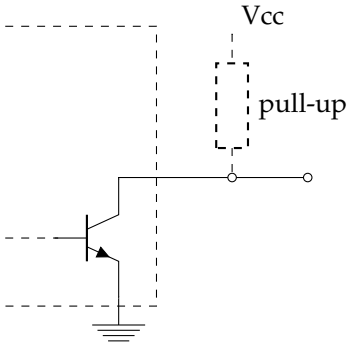
oder so:



und so weiter,
je nach
I/O Specs

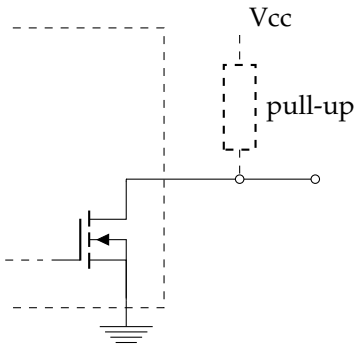
1.11 open Collector

Ausgänge sind nicht als push-pull (=Gegentakt), sondern nur *pull* (also nur der 'untere' BJT) ausgeführt, um mehrere solche Ausgänge als *wired OR* zusammenschalten oder durch Variation des *Pull-Up* anpassen zu können.



1.12 open Drain

dasselbe wie *open collector*, aber mit FETs (MOS Technik)



1.13 Pullup (I2C)

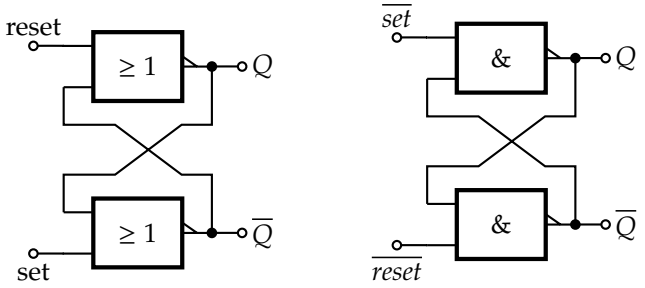
Der open-Drain Inter-Integrated-Circuit (IIC, I2C, I^2C) Bus spezifiziert Pullups von

$$R_p = \frac{t_{rise}}{0.8473 \times C_{bus}},$$

$$R_p = \frac{V_{DD} - V_{OL,max}}{I_{OL}},$$

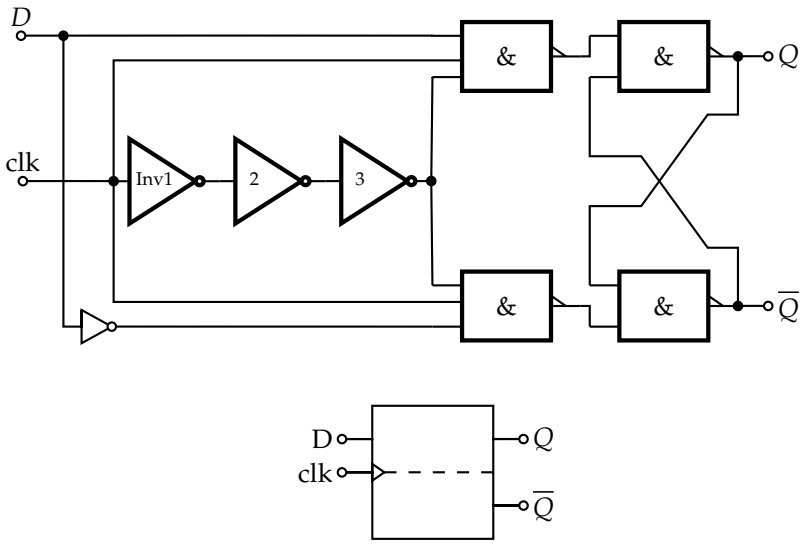
(1k8 beim Raspberry Pi - IIC Interface)
auf seinen *serial data (SDA)* und *serial clock (SCL)* Leitungen

1.14 RS Flipflop



Nand-RS ist *low-aktiv*

1.15 D Flipflop (DFF)



1.16 VLSI

SSI	small scale integration	<10 components wie zB.CA3046, CA3028, LM709 einfache Logikgatter wie 7400
MSI	medium scale integration	<100 components Decoder,Latches,Register,Rechenschaltungen
LSI	large scale integration	<1000 components umfangreichere Logikschaltungen und frühe Speicher (4116 und davor)
VLSI	very large scale integration	<100.000 components; Speicher-IC und Prozessoren
ULSI	ultra large scale integration	<1 Mio components;
SLSI	super large scale integration	<10 Mio components;
ELSI	extra large scale integration	<100 Mio components;
GLSI	giant large scale integration	>100 Mio components;

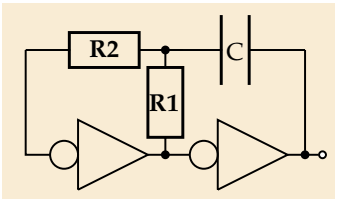
. . . alles *'Schnee von gestern'* oder vorgestern. Man hat die Entwicklungsstufen der IC- Integration

klassifiziert, vermutlich kennt niemand einen guten Grund dafür (ausser Marktschreierei natürlich), und noch weniger einen, das zu lernen...

1.17 D-/S-RAM, E/E/P/ROM, (Multilevel-) NAND-/NOR-FLASH

1.18 Digital-Oszillator (Taktgeber) mit Gatterschaltungen

1.18.1 mit 2 Stk Inverter



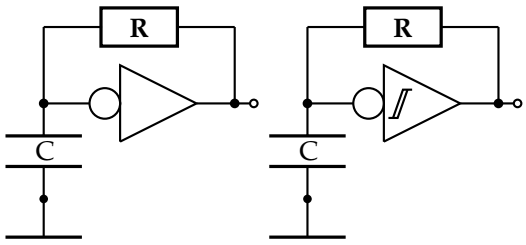
Oszillatorfrequenz:
R2 im Labor egal
@400Hz..400kHz

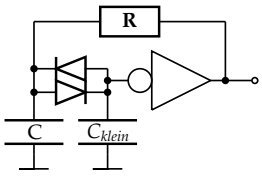
$$f \approx \frac{1}{3R_1C_1}$$

RC-Oszillatoren mit Gatterschaltungen wie
- NOT Gatter (Inverter)
- NAND / NOR Gatter
- Schmitt-Trigger
- Flipflops

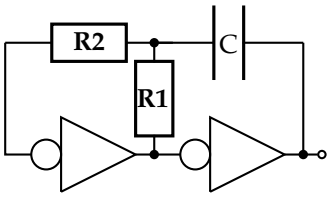
werden in digitalen Logikschaltungen als *Clock*generatoren und *Timer* verwendet, da sie besser kompatibel und *integrierbar* sind als lineare, analoge HF-LC-Oszillatoren, deren Signal man zudem noch in Logikpegel umsetzen müsste.

1.18.2 mit 1 Stk Inverter





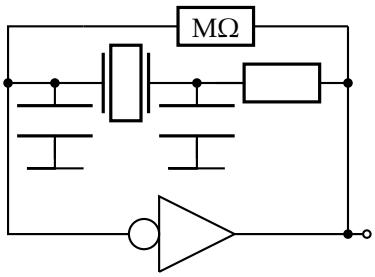
1.18.3 mit 2 Stk Inverter



Oszillatorfrequenz:
$74HC : 10R_1 > R_2 > 2R_1$
@400Hz..400kHz

$$f \approx \frac{1}{3R_1C_1}$$

1.18.4 mit Xtal



2 Zahlendarstellungen (number representation)

2.1 Binärzahlen

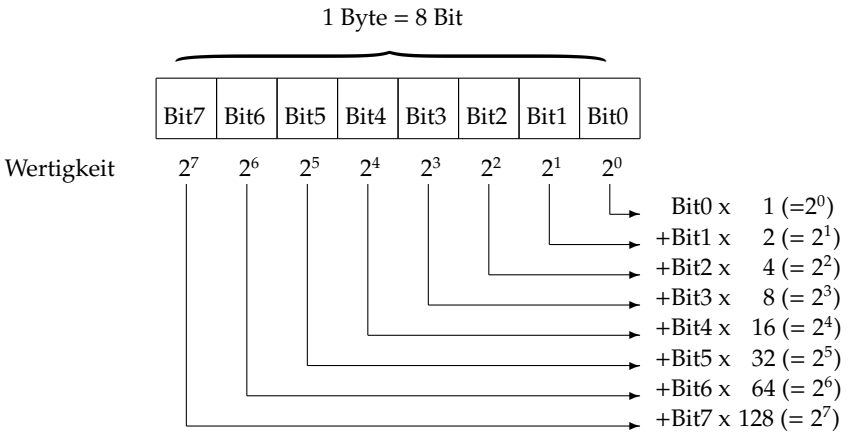
vorzeichenlose (unsigned) Binärzahl Das Dualsystem (lat. dualis „zwei enthaltend“), auch Zweiersystem oder Binärsystem genannt, ist ein Zahlensystem, das zur Darstellung von Zahlen nur zwei verschiedene Ziffern benutzt.

Die Zahldarstellungen im Dualsystem werden auch Dualzahlen oder Binärzahlen genannt. Letztere ist die allgemeinere Bezeichnung, da diese auch einfach für binärcodierte Zahlen stehen kann. Der Begriff Binärzahl spezifiziert die Darstellungsweise einer Zahl also nicht näher, er sagt nur aus, dass zwei verschiedene Ziffern verwendet werden.

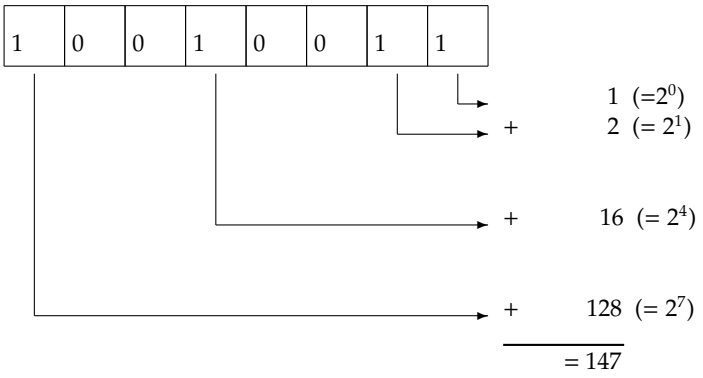
Im üblichen Dezimalsystem werden die Ziffern 0 bis 9 verwendet. Im Dualsystem hingegen werden Zahlen nur mit den Ziffern des Wertes null und eins dargestellt. Oft werden für diese Ziffern die Symbole 0 und 1 verwendet. Das Dualsystem ist das Stellenwertsystem mit der Basis 2, liefert also die dyadische (2-adische) Darstellung von Zahlen (Dyadik)

Aufgrund seiner Bedeutung in der Digitaltechnik ist es neben dem Dezimalsystem das wichtigste Zahlensystem.

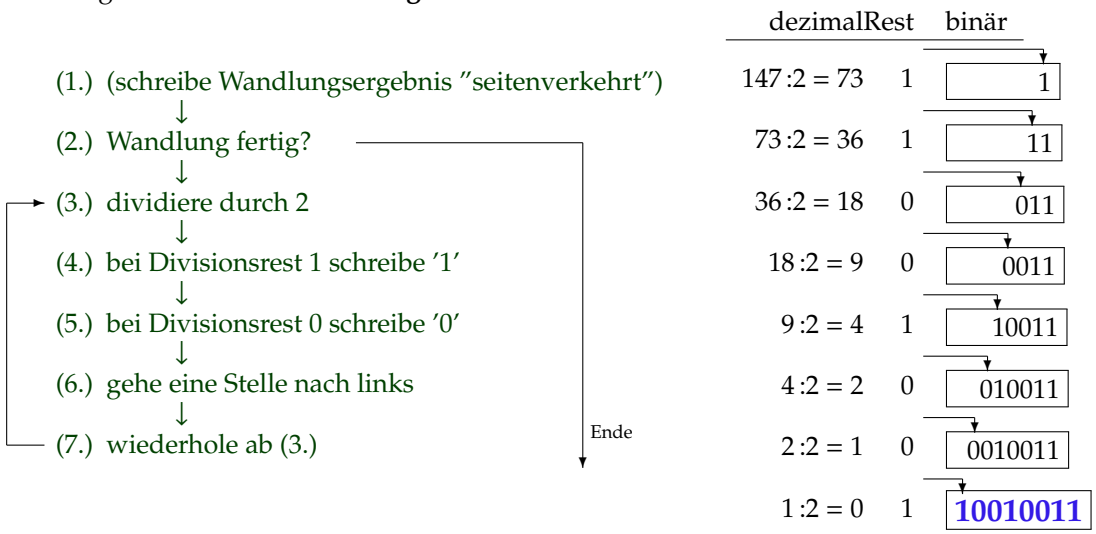
(aus https://de.wikipedia.org/wiki/Dualsystem 22Aug'20)
—



Beispiel:
Umwandlung **binär** → **dezimal**:



Beispiel:
Umwandlung **dezimal → binär – "fortgesetzte 2er-Division"**:



2er-Komplement Darstellung = vorzeichenbehaftet (signed) Das Zweierkomplement (auch 2-Komplement - verallgemeinert b-Komplement (b Basis) -, Zweikomplement, B(inär)-Komplement, Basiskomplement, two's complement) ist eine Darstellungsweise für **negative** Integer-Zahlen im Dualsystem, die keine zusätzlichen Zeichen wie '+' und '-' benötigt. Dies ist in der Digitaltechnik (insbesondere in Computern) von Bedeutung, da das Zweierkomplement es erlaubt, die Rechenart Subtraktion auf die Addition zurückzuführen und im Rahmen eines Addierwerks durchzuführen. Das Zweierkomplement setzt ein beschränktes, vordefiniertes Format (Bit-Länge) für die Darstellung von Binärzahlen voraus, da die Vereinbarung eine feste Bedeutung für das höchstwertige Datenbit verlangt.

Das Zweierkomplement kann als eine Interpretationsweise formatierter binärer Bitfolgen gesehen werden, welche für negative Werte von Integer-Variablen auftritt, für die ein in positiv und negativ geteilter Wertebereich definiert ist. Dies sind die sogenannten "signed integer" im Gegensatz zu den "unsigned integer" in Programmiersprachen. Für letztere tritt ein Zweierkomplement nicht auf. [...]

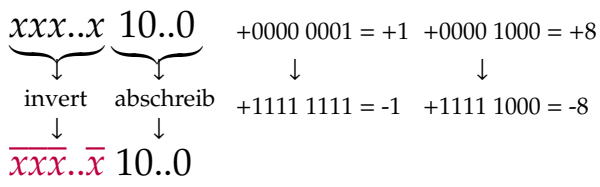
Bei der Codierung in der Zweierkomplementdarstellung ist die explizite Unterscheidung zwischen einem ausgezeichneten Vorzeichenbit und den Bits, die den Betrag beschreiben, nicht notwendig. Negative Zahlen sind daran zu erkennen, dass das höchstwertige Bit den Wert 1 hat. Bei 0 liegt eine positive Zahl oder der Wert 0 vor. Der Vorteil dieses Zahlenformates besteht darin, dass für Verarbeitung in digitalen Schaltungen keine zusätzlichen Steuerlogiken notwendig sind. [. . .]

Umwandlung positiv ↔ negativ per Hand:
a)
Alternative Faustregel:
1) Invertiere alle Stellen
2) Addiere 1

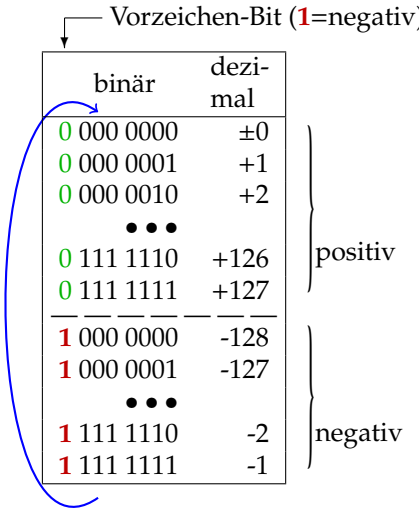
b)
Trick zur schnelleren Umwandlung (einer negativen in eine positive Binärzahl oder umgekehrt) von Hand:

Von rechts angefangen,
alle Nullen und die erste Eins abschreiben und
alle nachfolgenden Stellen invertieren:
1 Fange bei der rechten Stelle (niedrigstwertiges Bit) an.
2 Wenn diese Stelle eine 0 ist, schreibe eine 0 und gehe zu Punkt 3;
Wenn diese Stelle eine 1 ist, schreibe eine 1 und gehe zu Punkt 4.
3 Gehe ein Zeichen nach links und wiederhole Punkt 2.
4 Invertiere alle restlichen Stellen bis zum höchstwertigen Bit.

(aus https://de.wikipedia.org/wiki/Zweierkomplement 22Aug'20)



2er Komplement



positive Zahl
↓
1) Invertiere alle Stellen
2) Addiere 1
↓
negative Zahl
↓
1) Invertiere alle Stellen
2) Addiere 1
↓
wieder positive Zahl

Rechne zB.
'+1'(0000 0001) + '-1'(1111 1111) !
Ohne diese Verschiebung um '1'
hätte man
eine '+0' = 0000 0000 und
eine '-0' = 1111 1111,
sodass ein extra '+0' == '-0' Vergleicher
einzubauen wäre, und
'-0' +1 wäre '+0' !!
ungut!

int8_t . . . uint8_t . . . Integers

The C language defines several integer data types: integer, short integer, long integer, and character, all in both signed and unsigned varieties. The GNU C compiler extends the language to contain long long integers as well. The C integer types were intended to allow code to be portable among machines with different inherent data sizes (word sizes), so each type may have different ranges on different machines. The problem with this is that a program often needs to be written for a particular range of integers, and sometimes must be written for a particular size of storage, regardless of what machine the program runs on. To address this problem, the GNU C Library contains C type definitions you can use to declare integers that meet your exact needs. Because the GNU C Library header files are customized to a specific machine, your program source code doesn't have to be.

These typedefs are in stdint.h.

If you require that an integer be represented in exactly N bits, use one of the following types, with the obvious mapping to bit size and signedness:

int8_t
int16_t
int32_t
int64_t
uint8_t
uint16_t
uint32_t
uint64_t

If your C compiler and target machine do not allow integers of a certain size, the corresponding above type does not exist. If you don't need a specific storage size, but want the smallest data structure with at least N bits, use one of these:

int_least8_t
int_least16_t
int_least32_t
int_least64_t
uint_least8_t
uint_least16_t
uint_least32_t
uint_least64_t

If you don't need a specific storage size, but want the data structure that allows the fastest access while having at least N bits (and among data structures with the same access speed, the smallest one), use one of these:

int_fast8_t
int_fast16_t
int_fast32_t
int_fast64_t
uint_fast8_t
uint_fast16_t
uint_fast32_t
uint_fast64_t

If you want an integer with the widest range possible on the platform on which it is being used, use one of the following. If you use these, you should write code that takes into account the variable size and range of the integer.

intmax_t
uintmax_t

The GNU C Library also provides macros that tell you the maximum and minimum possible values for each integer data type. The macro names follow these examples: INT32_MAX, UINT8_MAX, INT_FAST32_MIN, INT_LEAST64_MIN, UINTMAX_MAX, INTMAX_MAX, INTMAX_MIN. Note that there are no macros for unsigned integer minima. These are always zero. Similiarly, there are macros such as INTMAX_WIDTH for the width of these types. Those macros for integer type widths come from TS 18661-1:2014.
There are similar macros for use with C's built in integer types which should come with your C compiler. These are described in Data Type Measurements.

Don't forget you can use the C sizeof function with any of these data types to get the number of bytes of storage each uses.

(aus https://www.gnu.org/software/libc/manual/html_node/Integers.html 23Aug'20)

short int long: 'C' Datentypen [...] Datentypen sind in C Deklarationen für Speicheradressen oder Variablen, die bestimmen, welche Operatoren und Funktionen auf die Adressen und Variablen angewendet werden können.

C bietet grundlegende arithmetische Datentypen zur Speicherung von Ganzzahlen und Gleitkommazahlen, sowie die nötige Syntax zur Erstellung von Feldern und Verbundtypen. Etliche Header-Dateien der C-Standard-Bibliothek bieten darüber hinaus Definitionen weiterer Datentypen, die jeweils über bestimmte nützliche Eigenschaften verfügen.

Grundlegende arithmetische Datentypen

C verfügt über die vier arithmetischen Datentypen char, int (beide für ganze Zahlen), float und double (beide für Kommazahlen). Die Auswahl eines dieser Datentypen beeinflusst die Größe des reservierten Speichers und die Größe der darstellbaren Werte. Darüber hinaus sind für die verschiedenen Datentypen unterschiedliche Operatoren und Funktionen zugelassen.

Der Verzicht auf festgeschriebene Größen und Wertebereiche, um möglichst viele Architekturen zu unterstützen, wird durch definierte minimale Wertebereiche und die folgende feste Relation abgemildert:

signed char ≤ short int ≤ int ≤ long int ≤ long long int.

("≤" bedeutet dabei, dass der rechts stehende Typ alle Werte des links stehenden Typs aufnehmen kann.)

Character

Zum Speichern eines Zeichens verwendet man in C den Datentyp Character, geschrieben als char. Vom Computer tatsächlich gespeichert wird nicht das Zeichen, sondern eine gleichbedeutende, acht Bit lange, Binärzahl (dadurch ergeben sich 256 verschiedene Werte, die einem Character zugewiesen werden können). Der Programmierer kann sich die Binärzahl leicht vom Computer in ein Zeichen oder eine dezimale Ganzzahl übersetzen lassen. Die Übersetzung einer Zahl in ein Zeichen und umgekehrt geschieht dabei anhand einer Tabelle (z. B.: ASCII-Tabelle oder EBCDIC).

Ein Character repräsentiert die kleinste adressierbare Einheit in C, in der Regel acht Bit. Deshalb wird die Größe von Objekten und Typen oft als ganzzahliges Vielfaches eines Characters angegeben. Je nach Compiler kann char entweder gleichbedeutend sein mit signed char (-128 bis 127, fast immer der Fall) oder mit unsigned char (0 bis 255). Um auch Zeichen aus Zeichensätzen aufnehmen zu können, die mehr Zeichen umfassen als der relativ kleine ASCII-Zeichensatz, wurde mit wchar_t bald ein zweiter für Zeichen konzipierter Datentyp eingeführt. Er umfasst in fast allen Implementierungen mehr als acht Bit.

Integer

Zum Speichern einer ganzen Zahl verwendet man eine Variable vom Datentyp Integer, geschrieben als int. Um den Wertebereich eines Integers zu verkleinern oder zu vergrößern, stellt man ihm einen der Qualifizierer short, long oder long long voran. Das Schlüsselwort int kann dann auch weggelassen werden, so ist long gleichbedeutend mit long int. Um zwischen vorzeichenbehafteten und vorzeichenlosen Ganzzahlen zu wechseln, gibt es die beiden Qualifizierer signed und unsigned. Für einen vorzeichenbehafteten Integer kann der Qualifizierer aber auch weggelassen werden, so ist signed int gleichbedeutend mit int.

Die Größe eines Integers ist vom jeweiligen Compiler abhängig, der C-Standard garantiert aber eine minimale Größe von 16 Bit. Die tatsächliche Größe beträgt heutzutage (je nach Prozessorarchitektur und Betriebssystem) meist 32 Bit, oft aber auch schon 64 und manchmal noch 16 Bit. In 16 Bit lassen sich 65536 verschiedene Werte speichern. Um die Verwendung von negativen Zahlen zu ermöglichen, reicht der Wertebereich bei 16 Bit gewöhnlich von -32768 bis 32767. Werden keine negativen Zahlen benötigt, kann der Programmierer mit unsigned int aber einen vorzeichenlosen Integer verwenden. Bei 16 Bit großen Integern ergibt das einen Wertebereich von 0 bis 65535.

Die tatsächliche Größe eines Integers ist in der Headerdatei <limits.h> abgelegt. INT_MAX ersetzt der C-Präprozessor beispielsweise durch den Wert, den der Typ int maximal annehmen kann.

Die C-Standard-Bibliothek ergänzt diese Datentypen über die plattformunabhängige Header-Datei <stdint.h> in der eine Fülle von Ganzzahltypen mit fester Länge definiert ist.

float, double und long double

Zahlen mit Nachkommastellen werden in einem der drei Datentypen float, double und long double gespeichert.

Unter Genauigkeit ist hierbei nicht die Anzahl der signifikanten Nachkommastellen zu verstehen - vielmehr schreibt der Standard mit FLT_DIG,DBL_DIG,LDBL_DIG lediglich die Anzahl der (direkt aufeinanderfolgenden) signifikanten Dezimalziffern als Genauigkeitskriterium vor, und hierbei mind. jeweils 6/10/10. Die Lage dieser Ziffernfolge (vor Komma, nach Komma, geteilt über Komma) legt der Standard nicht fest.

In den meisten C-Implementierungen entsprechen die Datentypen float und double dem international gültigen Standard für binäre Gleitpunktarithmetiken (IEC 559, im Jahr 1989 aus dem älteren amerikanischen Standard IEEE 754 hervorgegangen). Unter dieser Annahme implementiert float das "einfach lange Format", ein double das "doppelt lange Format". Dabei umfasst ein float 32 Bit, ein double 64 Bit. doubles sind also genauer. floats werden aufgrund dieses Umstands nur noch in speziellen Fällen verwendet. Die Größe von long doubles ist je nach Implementierung unterschiedlich, ein long double darf aber auf keinen Fall kleiner sein als ein double.

Die genauen Eigenschaften und Wertebereiche auf der benutzten Architektur können über die Headerdatei <float.h> ermittelt werden.

Komplexe Zahlen

Zusätzlich existieren seit C99 noch drei Gleitkomma-Datentypen für komplexe Zahlen, welche aus den drei Gleitkommatypen abgeleitet sind: float _Complex, double _Complex und long double _Complex. Ebenfalls in C99 eingeführt wurden Gleitkomma-Datentypen für rein imaginäre Zahlen: float _Imaginary, double _Imaginary und long double _Imaginary.

In einer hosted-Umgebung müssen die _Complex-Datentypen vorhanden sein; die _Imaginary-Typen sind optional. In einer freestanding-Umgebung sind diese sechs Datentypen optional.

bool

Bis zum C99-Standard gab es keinen Datentyp zum Speichern eines Wahrheitswerts. Erst seit 1999 können Variablen als _Bool deklariert werden und einen der beiden Werte 0 (falsch) oder 1 (wahr) aufnehmen. Die Größe einer _Bool-Variablen ist plattformabhängig und kann 8 Bit übersteigen. Inkludiert man den Header stdbool.h kann auch der Alias bool statt _Bool verwendet werden, sowie false und true statt 0 und 1.

void

Der Datentyp void wird im C-Standard als „unvollständiger Typ“ bezeichnet. Man kann keine Variablen von diesem Typ erzeugen. Verwendet wird void erstens, wenn eine Funktion keinen Wert zurückgeben soll, zweitens für die Deklarationen einer leeren Parameterliste für eine Funktion und drittens als Teil des regulären aber anonymen Datenzeigertyps void*, der Zeiger aller Datentypen (keine Funktionen) aufnehmen kann.

Datenmodell

Die C-Sprachnorm legt die Größe (und damit den Wertebereich) der einzelnen Basisdatentypen nicht fest, sondern definiert lediglich Relationen zwischen den Größen der Basisdatentypen und fordert für jeden Basisdatentyp jeweils Mindestgrößen. Daraus ergeben sich in der Praxis mehrere Ausgestaltungsmöglichkeiten, welche man Datenmodell oder auch Programmiermodell nennt.

Der Datentyp int wird auf einer Plattform in der Regel so festgelegt, dass seine Größe der natürlichen Datenwortgröße der CPU entspricht. Die Größe der Zeigertypen richtet sich nach der Größe des Speicherbereichs, der vom Programm aus adressierbar sein soll. Dieser Speicherbereich kann kleiner, aber auch größer sein, als der von der CPU-Architektur adressierbare Speicherbereich.

Auf heutigen Architekturen ist ein char meist 8 Bit groß, die anderen Datentypen müssen somit ein ganzzahliges Vielfaches von 8 Bit groß sein.

(aus https://de.wikipedia.org/wiki/Datentypen_in_C 23Aug'20)

2.2 Decimale Zahlen im ASCII Code

("decimal" bedeutet hier das *Zehnersystem* und nicht 'Kommazahlen')

Die ASCCI-Code-Werte 48 (0x30) bis 57 (0x39) entsprechen den *"druckbaren"* Dezimalziffer-Symbolen '0' bis '9'. Diese Repräsentation eignet sich bestens zur Ausgabe auf Monitoren und Druckern, zum Rechnen wegen des Offsets von 48 jedoch gar nicht - es ist zeitaufwendig auf binäre Kodierung umzuwandeln.

ASCII

binär	hexadecimal	decimal	Zeichen
0011 0000	0x30	48	'0'
0011 0001	0x31	49	'1'
0011 0010	0x32	50	'2'
0011 0010	0x33	51	'3'
0011 0100	0x34	52	'4'
0011 0101	0x35	53	'5'
0011 0110	0x36	54	'6'
0011 0111	0x37	55	'7'
0011 1000	0x38	56	'8'
0011 1001	0x39	57	'9'

Noch schlimmer wäre der (lange vorherrschende) EBCDIC *('European Binary Coded Decimal Interchange Code')*
[...] ist eine von IBM entwickelte 8-Bit-Zeichenkodierung, [...] EBCDIC wird fast ausschließlich auf Großrechnern verwendet. Die EBCDIC-Codepage gibt es in diversen Varianten. Erkennbar ist der historische Zusammenhang mit der 80-Zeichen-Lochkartenkodierung von IBM, bei der die Buchstaben A-I, J-R und S-Z die Positionen 1 bis 9 bzw. 2 bis 9 in der numerischen Zone benutzen. Eine Konsequenz daraus ist, dass - im Gegensatz zum ASCII-Zeichensatz - die Buchstaben A-Z nicht lückenlos aufeinander folgen, was die alphabetische Sortierung komplizierter macht.

Ähnlich wie das Unicode Transformation Format UTF-8 eine Kodierung von Unicode unter Beibehaltung der Kodierung des ASCII-Zeichensatzes darstellt, gibt es eine Unicode-Kodierung, die auf EBCDIC aufbaut. Dieses Format heißt UTF-EBCDIC.

(aus https://de.wikipedia.org/wiki/Extended_Binary_Coded_Decimal_Interchange_Code)

EBCDIC

binär	hexadecimal	decimal	Zeichen
1111 0000	0xF0	240	'0'
1111 0001	0xF1	241	'1'
1111 0010	0xF2	242	'2'
1111 0010	0xF3	243	'3'
1111 0100	0xF4	244	'4'
1111 0101	0xF5	245	'5'
1111 0110	0xF6	246	'6'
1111 0111	0xF7	247	'7'
1111 1000	0xF8	248	'8'
1111 1001	0xF9	249	'9'

2.3 Decimale Zahlen in BCD Code

Der BCD (Binary Coded Decimal) codiert nur die Ziffern 0 . . . 9 mit der $'0'_d$ beginnend bei 00000000_b

BCD

binär	hexadecimal	Ziffer
0000 0000	0x00	0
0000 0001	0x01	1
0000 0010	0x02	2
0000 0011	0x03	3
0000 0100	0x04	4
0000 0101	0x05	5
0000 0110	0x06	6
0000 0111	0x07	7
0000 1000	0x08	8
0000 1001	0x09	9

2.4 packed BCD Code

Da ja im BCD Bit_7 bis Bit_4 immer Null, also ungenutzt sind bringt man im packed BCD Code zwei Ziffern pro Byte unter. (zum Rechnen müssen sie aber wieder aufgetrennt werden).

packed BCD

binär	hexadecimal	Ziffern
0000 0000	0x00	0 0
0000 0001	0x01	0 1
0000 0010	0x02	0 2
[...]		
0000 1000	0x08	0 8
0000 1001	0x09	0 9

→

0001 0000	0x10	1 0
0001 0001	0x11	1 1
0001 0010	0x12	1 2
[...]		
0001 1001	0x19	1 9
0010 0000	0x20	2 0
0010 0001	0x21	2 1
[...]		
0010 1001	0x29	2 9
0011 0000	0x30	3 0
0011 0001	0x31	3 1
[...]		
1001 1000	0x98	9 8
1001 1001	0x99	9 9

2.5 BCD Kommazahlen

Vor allem auf kaufmännischen Systemen und Datenbanksprachen (ie. SQL) verbreitet sind *Fixkommazahlen* im BCD-Format. (Der Dezimalpunkt wird aber nicht als '.'-Zeichen mitgespeichert). BCD-Fließkommazahlen mit getrenntem Exponenten sind unüblich, da sie mehr im technischen als kaufmännischen Bereich erforderlich sind, wo das IEE754-Format (s.u.) vorherrscht.

Das *MySQL Reference Manual 5.1* sagt:

The FLOAT and DOUBLE data types are used to represent **approximate** numeric data values. For FLOAT, the SQL standard allows an optional specification of the precision (but not the range of the exponent) in bits following the keyword FLOAT in parentheses. MySQL also supports this optional precision specification, but the precision value is used only to determine storage size. A precision from 0 to 23 results in a four-byte single-precision FLOAT column. A precision from 24 to 53 results in an eight-byte double-precision DOUBLE column.

The DECIMAL and NUMERIC data types are used to store **exact** numeric data values.
In MySQL, NUMERIC is implemented as **DECIMAL.** These types are used to store values for which it is important to preserve exact precision, for example with **monetary** data. MySQL 5.1 stores DECIMAL and NUMERIC values in binary format. Before MySQL 5.0.3, they were stored as strings. When declaring a DECI-

MAL or NUMERIC column, the precision and scale can be (and usually is) specified; for example: salary DECIMAL(5,2)
In this example, 5 is the precision and 2 is the scale. The precision represents the number of significant digits that are stored for values, and the scale represents the number of digits that can be stored following the decimal point. If the scale is 0, DECIMAL and NUMERIC values contain no decimal point or fractional part. The maximum number of digits for DECIMAL or NUMERIC is 65, [...]

DAA Decimal Adjust instruction wird zur Umwandlung von Binärzahlen in die BCD-Darstellung (und damit auch in die dezimale ASCII-Codierung) verwendet: 4-Bit-Binärzahlen von 000 bis 1001 entsprechen den Ziffern '0' bis '9'und sind simpel durch Addition von '0011 0000'$_b$ (48_d) nach ASCII umwandelbar, **nicht aber** '1010' bis '1111'$_b$. Durch 'Überspringen' der Hexadezimalziffern A..F ≡ Addition der Zahl **6** (Ziffer +6)$_{mod16}$ erreicht man

a) den Übertrag auf die nächste Stelle
b) die richtig korrigierte BCD-Ziffer

zB. '0010 1011' (0x2B) 'elf-a-zwanzg' ⟶ '0011 0001' (31)

'DAA' wird meist direkt hinter eine ADD- oder SUB-Instruction codiert.

2.6 IEEE-754 floating point numbers

IEEE-754-1985
IEEE-754-2008
IEEE-754-2019
Die Norm IEEE 754 (ANSI/IEEE Std 754-1985; IEC-60559:1989 - International version) definiert Standarddarstellungen für binäre Gleitkommazahlen in Computern und legt genaue Verfahren für die Durchführung mathematischer Operationen, insbesondere für Rundungen, fest. Der genaue Name der Norm ist englisch *IEEE Standard for Binary Floating-Point Arithmetic for microprocessor systems* (ANSI / IEEE Std 754-1985).

Die aktuelle Ausgabe ist unter der Bezeichnung **ANSI / IEEE Std 754-2019 im Juli 2019** veröffentlicht worden. Weiter ist die Norm **IEEE 854-1987**, mit dem engl. Titel Standard for radix-independent floating-point arithmetic, in der **IEEE 754-2008 vollständig integriert** worden.

[...]
Überblick
In der Norm IEEE 754-1989 werden zwei Grunddatenformate für binäre Gleitkommazahlen mit 32 Bit (single precision) bzw. 64 Bit (double precision) Speicherbedarf und zwei erweiterte Formate definiert. Die IEEE 754-2008 umfasst die binären Zahlenformate mit 16 Bit als Minifloat, 32 Bit als single, 64 Bit als double und neu 128 Bit (quadruple). Zusätzlich kamen noch die dezimalen Darstellungen mit 32 Bit als Minifloat, 64 und 128 Bit hinzu.

Schließlich gab es Vorschläge und Implementierungen von weiteren Zahlenformaten, die nach den Prinzipien der IEEE 754-1989 Norm gestaltet sind und deshalb oft als IEEE-Zahlen bezeichnet werden, obwohl sie das nach der alten Definition streng genommen nicht sind. Dazu gehören die in den neuen Ausgaben integrierten Minifloats, die für die Ausbildung gedacht sind. Minifloats mit 16 Bit werden gelegentlich in der Grafikprogrammierung verwendet (auch von 11-Bit-FP hatma schon gheart, Anm:XH). Ebenso gehören auch mehrere nicht von IEEE 754-1989 definierte Zahlenformate mit mehr als 64 Bit, etwa das 80-Bit-Format (Extended Precision Layout...), welches die IA-32-Prozessoren intern in ihrer klassischen Gleitkommaeinheit (Floating Point Unit, FPU) verwenden, dazu.

Zahlenformate und andere Festlegungen des IEEE-754-Standards
IEEE 754 unterscheidet vier Darstellungen: einfach genaue (single), erweiterte einfach genaue (single extended), doppelt genaue (double) und erweiterte doppelt genaue (double extended) Zahlenformate. Bei den erweiterten Formaten ist nur jeweils eine Mindestbitzahl vorgeschrieben. Die genaue Bitzahl und der Biaswert bleiben dem Implementierer überlassen. Die Grundformate sind vollständig definiert.

Vor allem die Anzahl der Exponentenbits legt Maximum und Minimum der darstellbaren Zahlen fest. Die Anzahl der Mantissenbits bestimmt die (relative s. u.) Genauigkeit dieser Zahlen (und nur in geringem Maß das Maximum und das Minimum).

(aus https://de.wikipedia.org/wiki/IEEE_754, 22Aug'20)

*[...] Die Norm **IEEE 754** (ANSI/IEEE Std 754-1985; IEC-60559:1989 - International version)*
*definiert **Standarddarstellungen für binäre Gleitkommazahlen in Computern** und legt genaue Verfahren für die Durchführung mathematischer Operationen, insbesondere für Rundungen, fest. Der genaue Name der Norm ist englisch IEEE Standard for Binary Floating-Point Arithmetic for microprocessor systems (ANSI/IEEE Std 754-1985). Die aktuelle Ausgabe ist unter der Bezeichnung ANSI/IEEE Std 754-2008 im August 2008 veröffentlicht worden und umfasst neben*

der 754-1985 eine Erweiterung um zusätzlich ein binäres und zwei dezimale Datenformate. Weiter ist die Norm IEEE 854-1987, mit dem Titel englisch Standard for radix-independent floating-point arithmetic, in der IEEE 754-2008 vollständig integriert worden.

Überblick

In der Norm IEEE 754-1989 werden zwei Grunddatenformate für binäre Gleitkommazahlen mit 32 Bit (single precision) bzw. 64 Bit (double precision) Speicherbedarf und zwei erweiterte Formate definiert. Die IEEE 754-2008 umfasst die binäre Zahlenformate mit 16 Bit als Minifloat, 32 Bit als single, 64 Bit als double und neu 128 Bit. Zusätzlich kamen noch die dezimale Darstellungen mit 32 Bit als Minifloat, 64 und 128 Bit hinzu. Schließlich gab es Vorschläge und Implementierungen von weiteren Zahlenformaten, die nach den Prinzipien der IEEE 754-1989 Norm gestaltet sind und deshalb oft als IEEE-Zahlen bezeichnet werden, obwohl sie das nach der alten Definition streng genommen nicht sind. Dazu gehören die in den neuen Ausgaben integrierten Minifloats, die für die Ausbildung gedacht sind. Minifloats mit 16 Bit werden gelegentlich in der Grafikprogrammierung verwendet. Dazu gehören auch mehrere nicht von IEEE 754-1989 definierte Zahlenformate mit mehr als 64 Bit, etwa das 80-Bit-Format (Extended Precision Layout...), welches die IA-32-Prozessoren intern in ihrer klassischen Gleitkommaeinheit (Floating Point Unit, FPU) verwenden.

[...]

Null

Null repräsentiert die absolute Null. Auch Zahlen, die zu klein sind, um dargestellt zu werden (Unterlauf), werden auf Null gerundet. Ihr Vorzeichen bleibt dabei erhalten. Negative kleine Zahlen werden so zu −0,0 gerundet, positive Zahlen zu +0,0. Beim direkten Vergleich werden jedoch +0,0 und −0,0 als gleich angesehen.

Normalisierte Zahl

Die Mantisse besteht aus den ersten n wesentlichen Ziffern der Binärdarstellung der noch nicht normalisierten Zahl. Die erste wesentliche Ziffer ist die höchstwertige (d. h. am weitesten links stehende) Ziffer, die von 0 verschieden ist. Da eine von 0 verschiedene Ziffer im Binärsystem nur eine 1 sein kann, muss diese erste 1 nicht explizit abgespeichert werden; gemäß der Norm IEEE 754 werden nur die folgenden Ziffern gespeichert, die erste Ziffer ist eine implizite Ziffer oder ein implizites Bit (englisch „hidden bit“). Dadurch wird gewissermaßen 1 Bit Speicherplatz "gespart".

[...]

Unendlich

Der Gleitkommawert "Unendlich" repräsentiert Zahlen, deren Betrag zu groß ist, um dargestellt zu werden. Es wird zwischen +"Unendlich" und −"Unendlich" unterschieden. Die Berechnung von 1,0/0,0 ergibt per Definition ebenfalls +"Unendlich".

Keine Zahl (NaN)

Damit werden ungültige (oder nicht definierte) Ergebnisse dargestellt, z. B. wenn versucht wurde, die Quadratwurzel aus einer negativen Zahl zu berechnen. Einige „unbestimmte Ausdrücke“ haben als Ergebnis „keine Zahl“, zum Beispiel 0,0/0,0 oder „Unendlich“ − „Unendlich“. Außerdem werden NaNs in verschiedenen Anwendungsbereichen benutzt, um „Kein Wert“ oder „Unbekannter Wert“ darzustellen. Insbesondere der Wert mit dem Bitmuster 111...111 wird oft für eine „nicht initialisierte Gleitkommazahl“ benutzt. IEEE 754 fordert zwei Arten von Nichtzahlen: stille NaN (NaNq – quiet) und signalisierende NaN (NaNs – signalling). Beide stellen explizit keine Zahlen dar. Eine signalisierende NaN löst im Gegensatz zu einer stillen NaN eine Ausnahme (Trap) aus, wenn sie als Operand einer arithmetischen Operation auftritt. IEEE 754 ermöglicht dem Anwender das Deaktivieren dieser Traps. In diesem Falle werden signalisierende NaN wie stille NaN behandelt. Signalisierende NaN können genutzt werden, um uninitialisierten Rechnerspeicher zu füllen, so dass jedes Verwenden einer uninitialisierten Variable automatisch eine Ausnahme auslöst. Stille NaN ermöglichen den Umgang mit Rechnungen, die kein Ergebnis erzeugen können, etwa weil sie für die angegebenen Operanden nicht definiert sind. Beispiele sind die Division Null durch Null oder der Logarithmus aus einer negativen Zahl. Stille und Signalisierende NaN unterscheiden sich im höchsten Mantissenbit. Bei stillen NaN ist dieses 1, bei signalisierenden NaN 0. Die übrigen Mantissenbits können zusätzliche Informationen enthalten, z. B. die Ursache der NaN. Dies kann bei der Ausnahmebehandlung hilfreich sein. Allerdings schreibt der Standard nicht fest, welche Informationen in den übrigen Mantissenbits enthalten sind. Die Auswertung dieser Bits ist daher plattformabhängig. Das Vorzeichenbit hat bei NaN keine Bedeutung. Es ist nicht spezifiziert, welchen Wert das Vorzeichenbit bei zurückgegebenen NaN besitzt.

Rundungen

IEEE 754 unterscheidet zunächst zwischen binären Rundungen und binär-dezimalen Rundungen, bei denen geringere Qualitätsforderungen gelten. Bei binären Rundungen muss zur nächstgelegenen darstellbaren Zahl gerundet werden. Wenn diese nicht eindeutig definiert ist (genau in der Mitte zwischen zwei darstellbaren Zahlen), wird so gerundet, dass das niederwertigste Bit der Mantisse 0 wird. Dies passiert statistisch in 50 % der Fälle, so dass der von Knuth beschriebene statistische Drift in längeren Rechnungen vermieden wird. Eine zu IEEE 754 konforme Implementierung muss drei weitere vom Programmierer einstellbare Rundungen bereitstellen: Rundung gegen +Unendlich (immer aufrunden), Rundung gegen −Unendlich (immer abrunden) und Rundung gegen 0 (Ergebnis immer betragsmäßig verkleinern).

Exceptions, Flags und Traps

Treten bei der Berechnung Ausnahmen (Exceptions) auf, werden Status-Flags gesetzt. Im Standard wird vorgeschrieben, dass der Benutzer diese Flags lesen und schreiben kann. Die Flags sind „sticky“: werden sie einmal gesetzt, bleiben sie so lange erhalten, bis sie explizit wieder zurückgesetzt werden. Das Überprüfen der Flags ist beispielsweise die einzige Möglichkeit, 1/0 (=Unendlich) von einem Überlauf zu unterscheiden. Des Weiteren wird im Standard empfohlen, Trap Handler zu ermöglichen: Tritt eine Ausnahme auf, wird der Trap Handler aufgerufen, anstatt das Status-Flag zu setzen. Es liegt in der Verantwortung solcher Trap Handler, das entsprechende Status-Flag zu setzen oder zu löschen. Ausnahmen

werden im Standard in 5 Kategorien eingeteilt: Überlauf, Unterlauf, Division durch Null, ungültige Operation und Ungenau. Für jede Klasse steht ein Status-Flag zur Verfügung.

Geschichte

In der 1960er und frühen 1970er Jahren hatte jeder Prozessor sein eigenes Format für Gleitkommazahlen und seine eigene FPU oder Gleitkommasoftware, mit der das jeweilige Format verarbeitet wurde. Dasselbe Programm konnte auf verschiedenen Rechnern unterschiedliche Resultate liefern. Die Qualität der verschiedenen Gleitkommaarithmetiken war ebenfalls sehr unterschiedlich. Intel plante um 1976 für seine Mikroprozessoren eine eigene FPU und wollte die bestmögliche Lösung für die zu implementierende Arithmetik. Unter der Federführung der IEEE begannen 1977 Treffen, um FPUs für Gleitkommaarithmetik für Mikroprozessoren zu normieren. Die erste Version der Norm wurde 1985 verabschiedet und 2008 erweitert.

[...]

(aus http://de.wikipedia.org/w/index.php?oldid=114932354, 22.Mar'13)

Allgemeines

Die Darstellung einer Gleitkommazahl

$$x = s \cdot m \cdot b^e$$

besteht aus:

- Vorzeichen *s* (fast ausnahmslos 1 Bit)
- Basis *b* (bei normalisierten Gleitkommazahlen nach IEEE 754 ist $b = 2$)
- Exponent *e* (r Bits), nicht zu verwechseln mit dem „biased exponent" bzw. der Charakteristik
- Mantisse *m* (p Bits), manchmal als Signifikant bezeichnet

Das Vorzeichen $s = (-1)^S$ wird in einem Bit S gespeichert, sodass S = 0 positive Zahlen und S = 1 negative Zahlen markiert.

Der Exponent *e* ergibt sich aus der nichtnegativen Binärzahl *E* (*E* wird manchmal auch als Charakteristik oder biased exponent bezeichnet) durch Subtraktion eines festen Biaswertes *B*: $e = E - B$. Der Biaswert (engl: Verzerrung) berechnet sich durch $2^{r-1} - 1$, wobei *r* die Anzahl der Bits von *E* darstellt. Der Biaswert *B* dient also dazu, dass negative Exponenten durch eine vorzeichenlose Zahl (die Charakteristik *E*) gespeichert werden können, unter Verzicht auf alternative Kodierungen wie z. B. das Zweierkomplement. (vergleiche auch Exzesscode)

Schließlich ist die Mantisse $1 \leq m < 2$ ein Wert, der sich aus den p Mantissenbits mit dem Wert M als $m = 1 + M/2^p$ berechnet. Einfacher ausgedrückt denkt man sich an das Mantissenbitmuster M links eine "1," angehängt: m = 1,M.

- $s = (-1)^S$
- $e = E - B$

- $m = 1, M = 1 + M/2^p$

Dieses Verfahren ist möglich, weil durch Normalisierung (s. u.) die Bedingung $1 \leq m < 2$für alle darstellbaren Zahlen immer eingehalten werden kann. Da dann die Mantisse immer links mit 1. beginnt, braucht dieses Bit nicht mehr gespeichert zu werden. Damit gewinnt man ein zusätzliches Bit Genauigkeit.

Für Sonderfälle stehen spezielle Bitmuster zur Verfügung. Um diese Sonderfälle zu kodieren, sind zwei Exponentenwerte, der maximale ($E = 11\ldots111 = 2^r - 1$) und die Null ($E = 00\ldots000$) reserviert. Mit dem maximalen Exponentenwert werden die Sonderfälle NaN und ∞ kodiert. Mit Null im Exponenten wird die Gleitkommazahl 0 und alle denormalisierten Werte kodiert.

Werte außerhalb des normalen Wertebereichs (zu große bzw. zu kleine Zahlen) werden durch ∞ bzw. −∞ dargestellt. Diese Erweiterung des Wertebereichs erlaubt auch im Falle eines arithmetischen Überlaufs häufig ein sinnvolles Weiterrechnen. Neben der Zahl 0 existiert noch der Wert −0. Während $\frac{1}{0}$das Ergebnis ∞ liefert, ergibt $\frac{1}{-0}$den Wert −∞. Bei Vergleichen wird zwischen 0 und −0 nicht unterschieden.

Die Werte NaN (für engl. „not a number“, „keine Zahl“) werden als Darstellung für undefinierte Werte verwendet. Sie treten z. B. auf als Ergebnisse von Operationen wie $\frac{0}{0}$oder ∞ −∞. NaN werden in Signal-NaN (signalling NaN, NaNs) für Ausnahmebedingungen und stille NaN (quiet NaN, NaNq) unterteilt.

Als letzter Sonderfall füllen denormalisierte Zahlen (in IEEE 754r als subnormale Zahlen bezeichnet) den Bereich zwischen der betragsmäßig kleinsten normalisierten Gleitkommazahl und Null. Sie werden als Festkommazahlen gespeichert und weisen nicht dieselbe Genauigkeit auf wie die normalisierten Zahlen. Konstruktionsbedingt haben die meisten dieser Werte den Kehrwert ∞.

Zahlenformate und andere Festlegungen des IEEE-754-Standards

IEEE 754 unterscheidet vier Darstellungen: einfach genaue (single), erweiterte einfach genaue (single extended), doppelt genaue (double) und erweiterte doppelt genaue (double extended) Zahlenformate. Bei den erweiterten Formaten ist nur jeweils eine Mindestbitzahl vorgeschrieben. Die genaue Bitzahl und der Biaswert bleiben dem Implementierer überlassen. Die Grundformate sind vollständig definiert.

Die Anzahl der Exponentenbits legt den Wertebereich der darstellbaren Zahlen fest (s. u.). Die Anzahl der Mantissenbits legt die Genauigkeit dieser Zahlen fest.

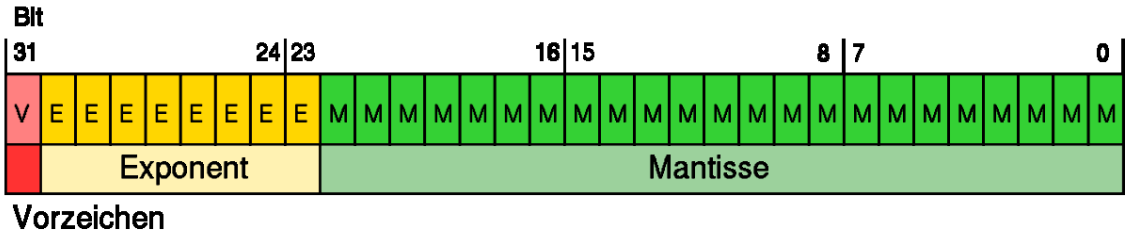
Die beiden letzten Beispiele demonstrieren ein minimales erweitertes Format.

Typ	Größe (1+r+p)	Exponent (r)	Mantisse (p)	Werte des Exponenten (e)	Biaswert (B)
single	32 bit	8 bit	23 bit	-126 ≤ *e* ≤ 127	127
double	64 bit	11 bit	52 bit	-1022 ≤ *e* ≤ 1023	1023
single extended	≥ 43 bit	≥ 11 bit	≥ 31 bit		
double extended	≥ 79 bit	≥ 15 bit	≥ 63 bit		
single extended, minimum	43 bit	11 bit	31 bit	-1022 ≤ *e* ≤ 1023	1023
double extended, minimum	79 bit	15 bit	63 bit	-16382 ≤ *e* ≤ 16383	16383

Für die angegebenen Formate ergibt sich die folgende Beschränkung des jeweiligen Zahlenbereichs. Die betragsmäßig kleinsten Zahlen sind hierbei nicht normalisiert. Der relative Abstand zweier Gleitkommazahlen ist größer als ϵ und kleiner gleich 2 $\cdot\epsilon$. Konkret ist der Abstand (und in diesem Fall auch der relative Abstand) der Gleitkommazahl 1 zur nächstgrößeren Gleitpunktzahl gleich 2 $\cdot\epsilon$. Dezimalstellen beschreibt die Anzahl der Stellen einer Dezimalzahl die ohne Genauigkeitsverlust gespeichert werden können. Die Mantisse ist rechnerisch durch das implizite Bit um eins größer als gespeichert.

Typ	ϵ	Dezimal-stellen	(betragsmäßig) kleinste Zahl	Größte Zahl
single	$2^{-(23+1)}$ $\approx 5{,}960 \cdot 10^{-8}$	7 … 8	$2^{-127} = 5{,}877 \cdot 10^{-39}$	$(1-2^{-24}) \times 2^{128}$ $\approx 3{,}403 \cdot 10^{38}$
double	$2^{-(52+1)}$ $\approx 1{,}110 \cdot 10^{-16}$	15 … 16	$2^{-1023} = 1{,}1125 \cdot 10^{-308}$	$(1-2^{-53}) \times 2^{1024}$ $\approx 1{,}798 \cdot 10^{308}$
single extended, minimum	$2^{-(31+1)}$ $\approx 2{,}328 \cdot 10^{-10}$	9 … 10	$2^{-31} \times 2^{-1022} = 2^{-1053}$ $\approx 1 \cdot 10^{-317}$	$(1-2^{-32}) \times 2^{1024}$ $\approx 1{,}798 \cdot 10^{308}$
double extended, minimum	$2^{-(63+1)}$ $\approx 5{,}521 \cdot 10^{-20}$	19 … 20	$2^{-63} \times 2^{-16382} =$ 2^{-16445} $\approx 4 \cdot 10^{-4951}$	$(1-2^{-64}) \times$ 2^{16384} $\approx 1{,}190 \cdot 10^{4932}$

Die Anordnung der Bits einer *single* zeigt die nachfolgende Abbildung. Die bei einer Rechenanlage konkrete Anordnung der Bits im Speicher kann von diesem Bild abweichen und hängt von der jeweiligen Bytereihenfolge (little/big endian) und weiteren Rechnereigenheiten ab.



Die Anordnung mit *Vorzeichen – Exponent – Mantisse* in genau dieser Reihenfolge bringt (innerhalb eines Vorzeichenbereiches) die dargestellten Gleitkommawerte in dieselbe Reihenfolge wie die durch dasselbe Bitmuster darstellbaren Signed-Integer-Werte. Damit können für die Vergleiche von Gleitkommazahlen dieselben Operationen wie für die Vergleiche von Signed-Integers verwendet werden. Kurz: die Gleitkommazahlen können lexikalisch sortiert werden.

Hierbei ist jedoch zu beachten, dass für steigende negative Signed-Integer-Werte der entsprechende Fließkommawert gegen minus unendlich geht, die Sortierung also umgekehrt ist.

Auch wenn in diesem Artikel hauptsächlich das Zahlenformat erörtert wird, liegt die Bedeutung der Norm IEEE 754 auch darin, dass für Gleitkommazahlen genaue Vorschriften für

- Rundung
- arithmetische Operationen
- Wurzelberechnung
- Konversionen
- Ausnahmebehandlung (Exception handling)

festgelegt wurden.

Beispiele

Berechnung Dezimalzahl → IEEE754-Gleitkommazahl

Die Zahl $18{,}4_{10}$ soll in eine Gleitkommazahl umgewandelt werden, dabei nutzen wir den Single IEEE-Standard.

1. Umwandlung der Dezimalzahl in eine duale Festkommazahl ohne Vorzeichen

```
18,4

18/2 = 9 Rest 0 (Least-Significant Bit)
 9/2 = 4 Rest 1
 4/2 = 2 Rest 0
 2/2 = 1 Rest 0
 1/2 = 0 Rest 1 (Most-Significant-Bit)
     = 10010

0,4*2 = 0,8 -0 (Most-Significant-Bit)
0,8*2 = 1,6 -1
0,6*2 = 1,2 -1
0,2*2 = 0,4 -0
0,4*2 = 0,8 -0
0,8*2 = 1,6 -1 (Least-Significant-Bit)
      *
      *
      *
      = 0,0110011001100110011...

18,4  = 10010,0110011001100110011...
```

2. Normalisieren und Bestimmen des Exponenten

```
Bias = 1x '0' + (r-1)x '1', für r=8: 01111111
```

```
10010,0110011001100... * 2^(01111111-01111111) =
1001,0011001100... * 2^(10000000-01111111) =
100,10011001100... * 2^(10000001-01111111) =
10,0100110011001100... * 2^(10000010-01111111) =
1,0010011001100... * 2^(10000011-01111111)

Mantisse: 1,0010011001100...
Exponent mit Bias: 10000011
```

3. Vorzeichen-Bit bestimmen

```
positiv → 0
```

4. Die Gleitkommazahl bilden

```
1 Bit Vorzeichen + 8 Bit Exponent + 23 Bit Mantisse

0 10000011 00100110011001100110011 → die Vorkomma-Eins wird als Hidden Bit weggelassen;
                                      da dort immer eine 1 steht, braucht man diese nicht zu speichern.
```

Berechnung IEEE754-Gleitkommazahl → Dezimalzahl

Nun soll die Gleitkommazahl von oben wieder in eine Dezimalzahl zurück gewandelt werden, gegeben ist also folgende IEEE754-Zahl:

```
0 10000011 00100110011001100110011
```

1. Berechnung des Exponenten

```
Umwandeln des Exponenten in eine Dezimalzahl:
10000011 -> 131

Da dies aber der Biased Exponent ist, der zuvor um den Bias verschoben wurde, wird nun der Bias
wieder abgezogen:
131-127 = 4 ist also der Exponent
```

2. Berechnung der Mantisse

```
Da es sich um eine normalisierte Zahl handelt, wissen wir, dass sie eine 1 vor dem Komma hat:
1,00100110011001100110011

Nun muss das Komma um 4 Stellen nach rechts verschoben werden:
10010,0110011001100110011
```

3. Umwandlung in eine Dezimalzahl

```
Vorkommastellen:
10010 (2) = 18 (10)

Nachkommastellen:
0,0110011001100110011 (2) ≈ 0.39999961853 (10)
(Um den Wert der Nachkommazahl zu erhalten, muss man denselben Prozess durchführen wie bei ganzen Zahlen,
nur in umgekehrter Richtung. Also von links nach rechts. Dabei muss der Exponent negativ sein und
mit einer 1 beginnen.
In der Form 0*2^-1+1*2^-2+1*2^-3+0*2^-4+0*2^-5+1*2^-6+1*2^-7+0*2^-8+0*2^-9+1*2^-10+1*2^-11+0*2^-
12+0*2^-13+1*2^-14+1*2^-15+0*2^-16+0*2^-17+1*2^-18+1*2^-19)
Weil die 0,4 im binären eine periodische Darstellung hat, kann sie nicht mehr genau umgewandelt
werden
```

4. Vorzeichen

```
Vorzeichenbit ist eine 0, also ist es eine positive Zahl
```

5. Dezimalzahl "zusammensetzen"

```
18,39999961853
```

normalized FPnumbers

rounding (Rundung)

2.7 komplexe Zahlen (complex numbers)

2.8 Vektoren (vectors)

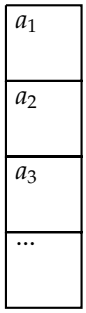
Bekanntlich gibt es (in der Mathematik) Zeilenvektoren

$\vec{v} = (v_1, v_2, v_3, \dots)$

und Spaltenvektoren

$$\vec{v} = \begin{pmatrix} v_1, \\ v_2, \\ v_3, \\ \dots \end{pmatrix}$$

Die Elemente eines Vektors nennt man *Komponenten* [1]
Dem RAM-Speicher ist das wurscht, wie Du es hinschreibst - er muss es nur speichern und wiederfinden, und das geht so:



Speicheradresse + 0

Speicheradresse + 1

Speicheradresse + 1

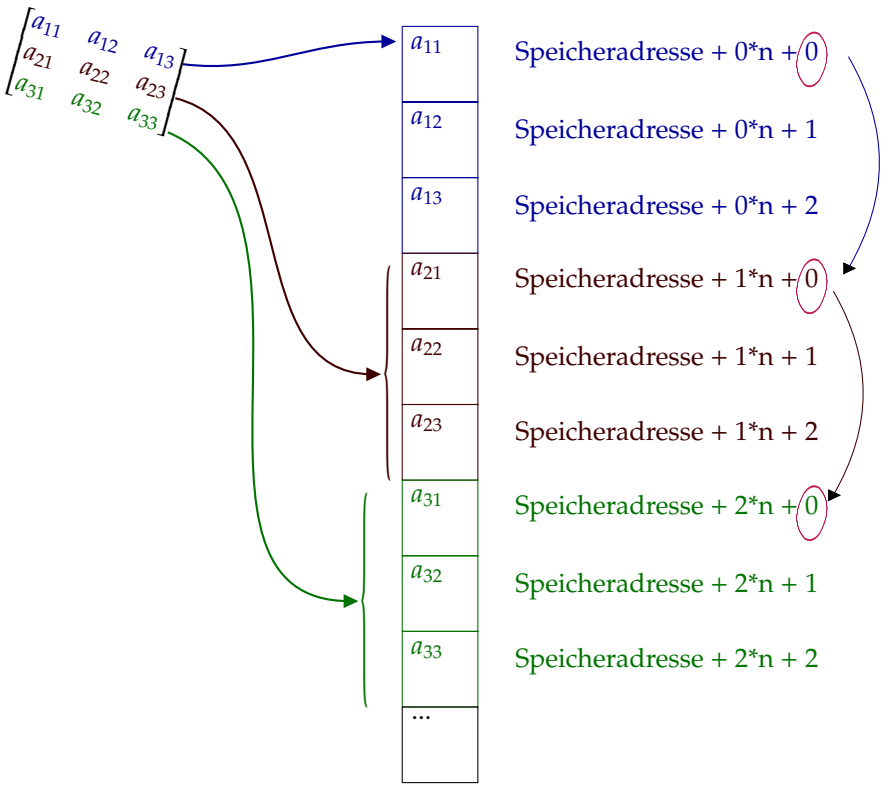
[1] (von lat. 'com' = zusammen und 'ponere' = legen, stellen https://www.frag-caesar.de/lateinwoerterbuch/ponere-uebersetzung.html)

2.9 Matrizen (matrices)

Eine **Matrix** (Mehrzahl: Matrizen) hat Zeilen und Spalten wie eine Tabelle

$$\underline{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Im **Gegensatz zu M$-Excel-Tabellen** wird **zuerst der Zeilenindex** und dann erst der Spaltenindex angegeben, also '3B' (Mathe) und nicht 'B3' (Excel/Calc), ausserdem indiziert die Mathematik nur mit Nummern, zB. [3,2], nicht [3,B].
Eine Matrix mit m Zeilen und n Spalten nennt man eine **m x n Matrix**:



Hierzu besonders geeignet / gedacht sind zB. die AVR-Instructions

- 'LDD' (load with displacement) und
- 'STD' (store with displacement)

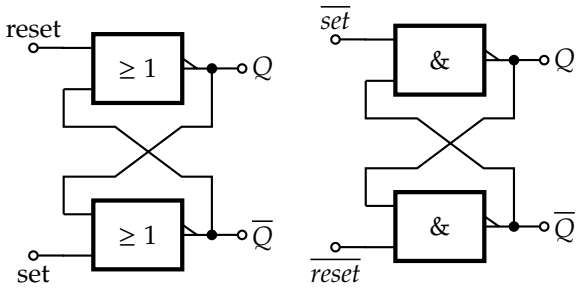
s. → AVR8-instruction-set-manual.pdf

Teil II

Sequentielle Logik

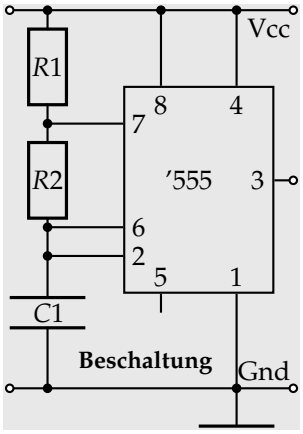
3 Flipflop (FF)

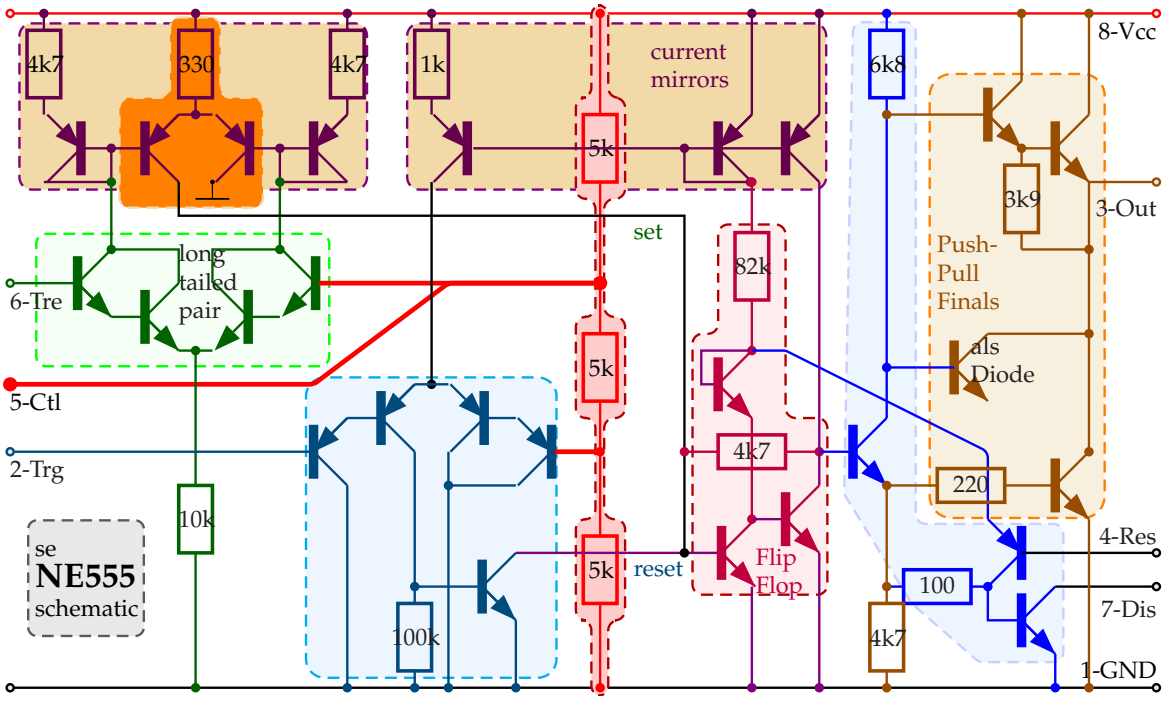
3.1 RS-FF



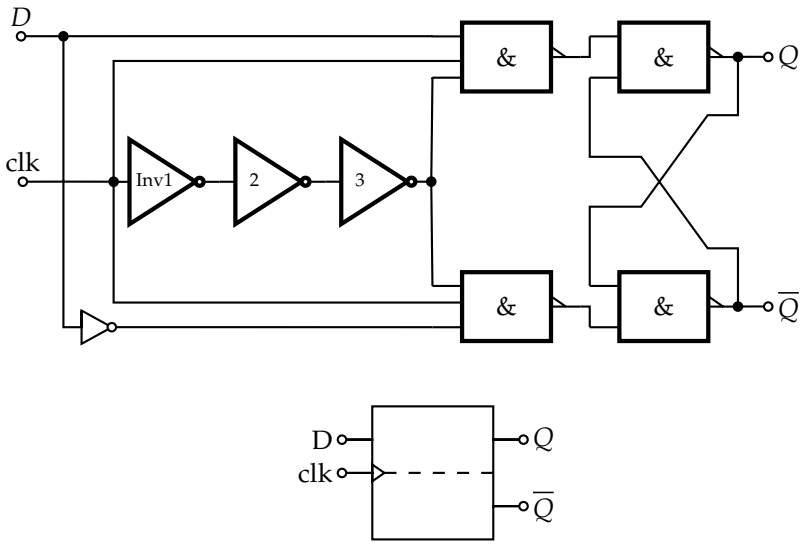
3.2 Hysterese

3.3 NE555-Flipflop

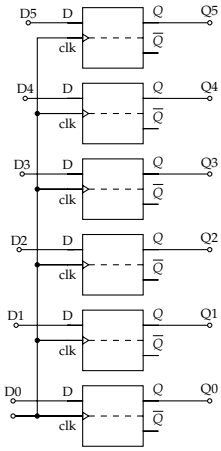




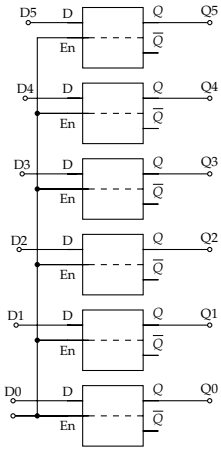
3.4 DFF



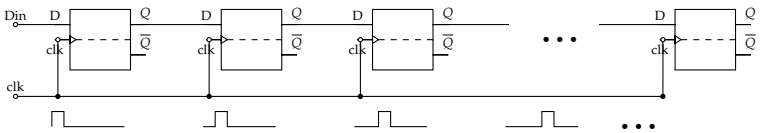
4 Register



4.1 Latch

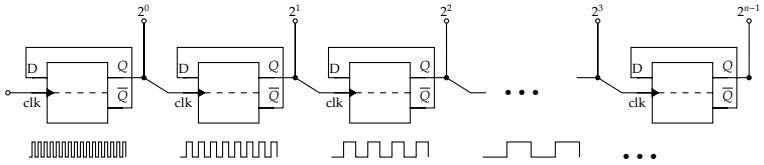


4.2 Schieberegister



5 Counter

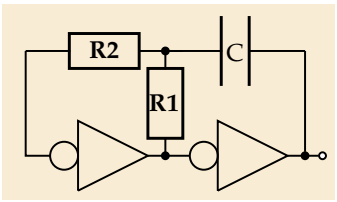
5.1 Johnson Ripple Counter



5.2 Synchronzähler

6 Digital-Oszillator (Taktgeber) mit Gatterschaltungen

6.1 mit 2 Stk Inverter



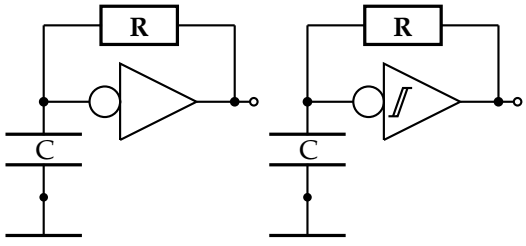
Oszillatorfrequenz:
R2 im Labor egal
@400Hz..400kHz

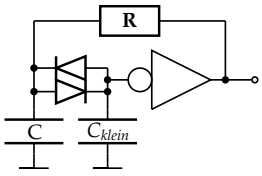
$$f \approx \frac{1}{3R_1C_1}$$

RC-Oszillatoren mit Gatterschaltungen wie
- NOT Gatter (Inverter)
- NAND / NOR Gatter
- Schmitt-Trigger
- Flipflops

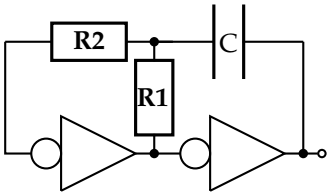
werden in digitalen Logikschaltungen als *Clock*generatoren und *Timer* verwendet, da sie besser kompatibel und *integrierbar* sind als lineare, analoge HF-LC-Oszillatoren, deren Signal man zudem noch in Logikpegel umsetzen müsste.

6.2 mit 1 Stk Inverter





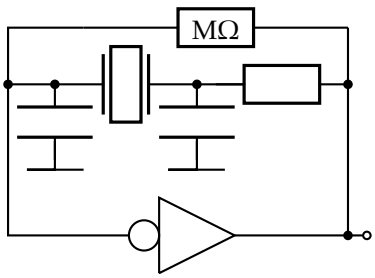
6.3 mit 2 Stk Inverter



Oszillatorfrequenz:
$74HC : 10R_1 > R_2 > 2R_1$
@400Hz..400kHz

$$f \approx \frac{1}{3R_1C_1}$$

6.4 mit Xtal



7 Schaltwerke

Schaltwerke FSM Finite State larger[2]Machine (häufigsten FSM sind Prozessoren und Zähler)

- ≡ *Schrittschaltwerk*
- ≡ *Automat*
- ≡ ***FSM** - finite state machine*
- ≡ *sequential circuit*
- ≡ *Automaton*
- ≡ *Ablaufsteuerung*

genannt.

Die (dzt.)werden auch Schaltwerke (*sequential circuit*) haben **Zustände** (≡Flipflops).

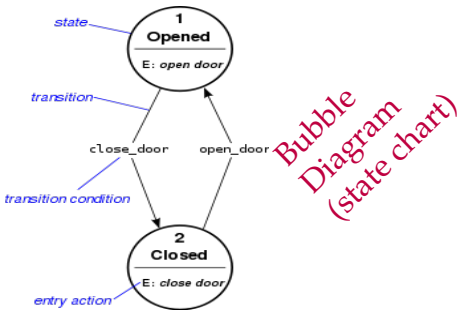
(Schaltnetze (*combinational circuit*) sind kombinatorische Schaltungen und haben **keine** Zustände)

FSM ::= $(\mathbb{Z}, \mathbb{I}, \mathbb{O}, f_S, f_o) \quad \epsilon \quad \{\mathbb{Z} \times \mathbb{I} \times \mathbb{O} \times f_S \times f_o\}$

$\mathbb{Z}$ ... Menge der Zustände
$\mathbb{I}$ ... Menge der Inputs
$\mathbb{O}$ ... Menge der Outputs
f_s ... Zustands-Überführungs-Funktion
f_o ... Output-Funktion

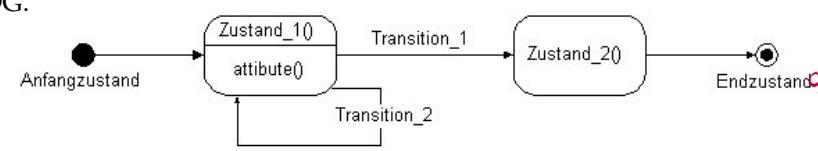
FSM entwerfen wir anhand

→ "Bubble-Diagrams" == "state charts"



Bubble Diagram (state chart)

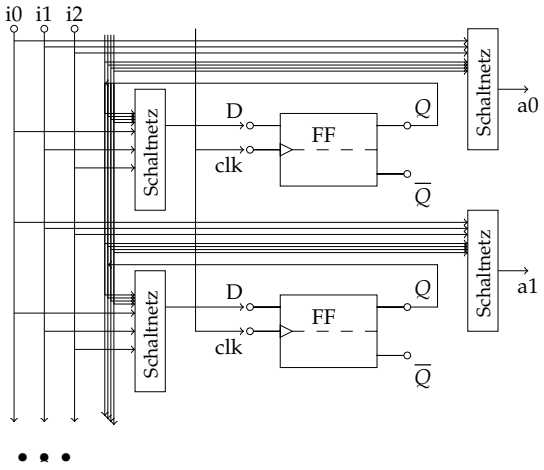
→ ≡ Zustands-DG.



Statechart Diagram

FSM realisieren wir mit

- \+ Flipflops als **Speicherelemente** (FF)
- \+ plus kombinatorischen **Input (i)** Schaltnetzen
- \+ plus kombinatorischen **Output (a)** Schaltnetzen



7.1 Zustands- u. Output Tabellen

FSM werden zur maschinellen Verarbeitung mittels Tabellen (statt Diagrammen) charkterisiert. (eignen sich insbesondere auch beim Software-Engineering (SE)!)

Zustandsfunktions- Tabelle (Zustandsüberführungsfunktion, state transition table) Eine Zustands-Abfolge wird auch *Zustands-Trajektorie* genannt.

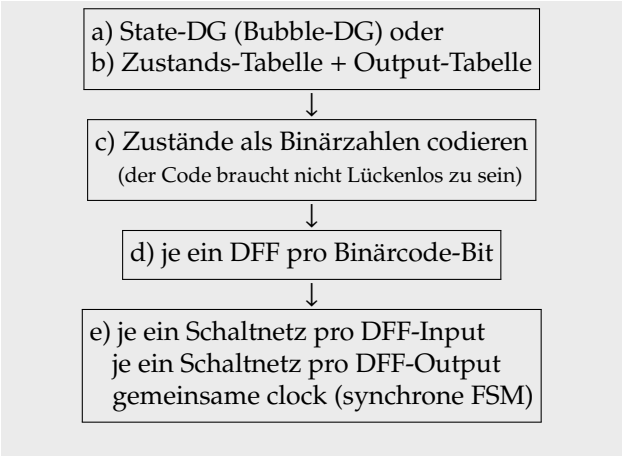
Zustand	Inputs			
	i_1	i_2	...	i_N
z_1	z_{11}	z_{12}	...	z_{1N}
z_2	z_{21}	z_{22}	...	z_{2N}
...		...		
z_M	z_{M1}	z_{M2}	...	z_{MN}

Tabelle der Folgezustände 'z'

Outputfunktions- Tabelle

Zustand	Inputs			
	i_1	i_2	...	i_N
z_1	o_{11}	o_{12}	...	o_{1N}
z_2	o_{21}	o_{22}	...	o_{2N}
...		...		
z_M	o_{M1}	o_{M2}	...	o_{MN}

Tabelle der Outputs 'o'



7.2 Entwurfsmethoden für Schaltwerke:

. Schaltwerk-Minimisierung s.zB. https://www.youtube.com/watch?v=GZAjyENwXkk (29Okt'22)

. **Row Equivalence Method** is a basic method for state minimization (This method does not always lead to optimize number of states).

Row Equivalence Method for State Minimization
By DR. SHIRSHENDU ROY / 10th September 2021 / Uncategorized

Row Equivalence Method for State Minimization is a very simple technique to reduce number of states of an FSM. In the row equivalence method, it is checked that rows of a state table are equivalent or not. Here, a comparatively strict definition of state equivalence is used. The conditions for two states S_1 and S_2 to be equivalent are

1. The outputs must be same for both the states.
2. The k-successors must be same for all the input conditions.

present state	next state x=0	x=1	Output
S0	S1	S2	0
S1	S3	S4	0
S2	S5	S6	0
S3	S3	S4	1
S4	S5	S6	0
S5	S3	S4	0
S6	S5	S6	0

Table 1: State table example for state minimization.

present state	next state		Output
S0	S1*	S2*	0
S1*	S3	S2*	0
S2*	S1*	S2*	0
S3	S3	S2*	1

Table 2: Minimization of state table shown in Table 1 using Row Equivalent

Row Equivalence Method is explained here with the help of the state table shown in Table 1. Here, states S_1 and S_5 can be said equivalent as their output is same and their k-successors are also same. Similarly, the states S_2, S_4 and S_6 are equivalent. Thus using this simple state minimization technique the state table is reduced to the state table as shown in Table 2. Here, S_1^* is written for states S_1 and S_5. Similarly, S_2^* is written for states S_2, S_4 and S_6.

. **Implication Chart Method** is a rigorous technique for state minimization. Though the chart preparation is difficult, it supports machine implementation.

Implication Chart Method for State Minimization
By DR. SHIRSHENDU ROY / 10th September 2021 / Uncategorized

Implication Chart Method for State Minimization is very popular for reducing the steps of an FSM and it is more machine friendly method than Row Equivalence Technique. In this method, a chart is prepared to find the equivalent steps. The implication chart is shown in Figure 1. States are written along the x-axis as S_0, S_1,. . . ,S_n and the states are written along the y-axis in the reverse order. A square X_{ij}, contains the equivalent states between S_i and S_j. The Implication Chart can be modified as $X_{ij} = X_{ji}$ and thus the triangle above the diagonal can be removed. Also, the diagonal can be removed as there is no sense to find equivalence to a state and itself. The reduced implication chart is shown in Figure 2.

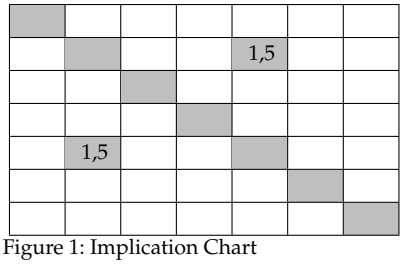


Figure 1: Implication Chart

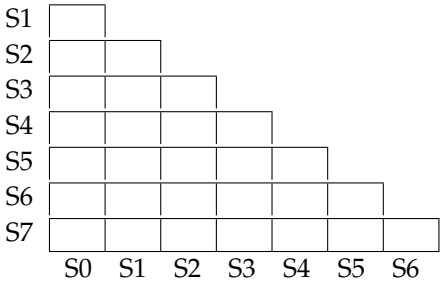


Figure 2: Reduced Implication Chart

First step of Implication Chart Method for State Minimization is to fill the squares of the chart correctly. According to the state table shown in Table 1, an implication chart is shown in Figure 3. The squares are filled as by two rows. First row consists of 0-successors and the second row consists of 1-successors. In the topmost square from the left side, first row is $S_3 - S_1$ and the second row is $S_5 - S_2$. These pairs are called as implied state pairs. During the filling of squares, some boxes are crossed for states which can not be combined. For example, S_2 and S_0 can not be combined as their output is different.

present state	next state x=0	x=1	Output
S0	S1	S2	1
S1	S3	S5	1
S2	S5	S4	0
S3	S1	S6	1
S4	S5	S2	0
S5	S4	S3	0
S6	S5	S6	0

Table 1: State table example for state minimization.

$x = 0, S_1 \rightarrow S_3 \; and \; S_0 \rightarrow S_1$
$x = 1, S_1 \rightarrow S_5 \; and \; S_0 \rightarrow S_2$

S1	$S_3 - S_1$ $S_5 - S_2$					
S2	✕	✕				
S3	$S_1 - S_1$ $S_6 - S_2$	$S_1 - S_3$ $S_6 - S_5$	✕			
S4	✕	✕	$_5 - S_5$ $S_4 - S_4$	✕		
S5	✕	✕	$_4 - S_5$ $S_3 - S_4$	✕	$S_4 - S_5$ $S_3 - S_2$	
S6	✕	✕	$S_5 - S_5$ $S_6 - S_4$	✕	$S_5 - S_5$ $S_6 - S_2$	$S_5 - S_4$ $S_6 - S_3$

Figure 3: Implication chart after filling of squares and marking cross for not related states.

State minimization by Implication chart method is accomplished by some passes until no further combination is possible. Searching of equivalent states is done from the top to the bottom starting from the left side of the chart. The state S_1 can be combined with state S_0 or S_3 as shown in Figure 4. But state S_2 and S_5 can not be combined as the square corresponding to the states S_3 and S_4 is marked crossed. So, the square block corresponding to S_2 and S_5 is cross marked. Similarly for the square block X_{65} as state S_6 and S_3 can not be combined. At the end of the first pass it can be concluded that S_0, S_1 and S_3 can be equivalent. Also, $S_2, S_4 \; and \; S_6$ can be combined. The state S_5 can not be combined any other states.

S1	$S_3 - S_1$ $S_5 - S_2$					
S2	✕	✕				
S3	$S_1 - S_1$ $S_6 - S_2$	$S_1 - S_3$ $S_6 - S_5$	✕			
S4	✕	✕	$S_5 - S_5$ $S_4 - S_4$	✕		
S5	✕	✕	~~$S_4 - S_5$~~ ~~$S_3 - S_4$~~	✕	~~$S_4 - S_5$~~ ~~$S_3 - S_2$~~	
S6	✕	✕	$S_5 - S_5$ $S_6 - S_4$	✕	$S_5 - S_5$ $S_6 - S_2$	~~$S_5 - S_4$~~ ~~$S_6 - S_3$~~

Figure 4: Implication chart after pass 1.

Second pass started similarly from the top to bottom starting from the left to right as shown in Figure 5. Here, the square block X_{10} is cross marked as we have seen in the earlier pass that S_2 can not be combined with S_5. Similarly, the square block X_{31} is cross marked because of state S_5. After the second pass it can be concluded that S_0 and S_3 are equivalent. Also, S_2, S_4 and S_6 are equivalent.

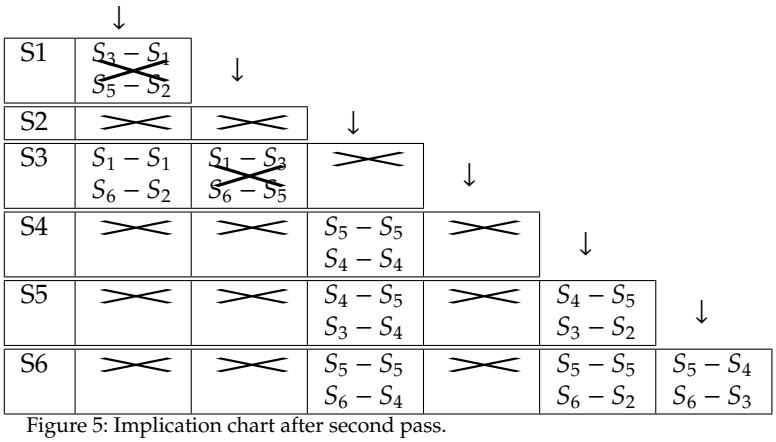


Figure 5: Implication chart after second pass.

Another pass can be run but after the third pass there is no new state for reduction. Thus finally the modified state table can be formed. Here, S_0 and S_3 are combined as S_0^*. The states S_2, S_4 *and* S_6 are combined as S_2^*. The modified state table is shown below in Table 2. Here, initially there were seven states and after minimization there are now four states.

present state	next state	Output
S0*	S1 S2	1
S1	S0* S5	1
S2*	S5 S2*	0
S5	S0* S0*	0

. **State Partition Method** is simple but another rigorous method for state minimization. It is also machine realizable.

State Partition Method for State Minimization By DR. SHIRSHENDU ROY / 10th September 2021 / Uncategorized

is also powerful as the Implication Chart Method and better than Row Equivalence method. In this technique, the states are partitioned into groups based on the possibility that they can be combined. Lets consider the example state table as shown in Table 1 for state minimization using state partition method. This method also perform state minimization after some passes. These passes are described below.

present state	next state x=0 x=1	Output
S0	S1 S2	1
S1	S3 S5	1
S2	S5 S4	0
S3	S1 S6	1
S4	S5 S2	0
S5	S4 S3	0
S6	S5 S6	0

Table 1: Example State table

Start: In the first pass there is only one group and this is P = ($S_0\,S_1\,S_2\,S_3\,S_4\,S_5\,S_6$).

First Pass: In the first pass, the states which have different outputs are partitioned in separate groups. Here, S_0, S_1 *and* S_3 have same output and thus grouped in $P_1 = (S_0\,S_1\,S_3)$. The rest of the states have same output and thus they grouped as $P_2 = (S_2\,S_4\,S_5\,S_6)$.

Second Pass: In this pass, the states are partitioned based upon their k-successors. In order to combine two states, their k-successors should be in the same partition or group.

Consider the first partition P_1 and their k-successors are
0-successors – $S_0 \rightarrow S_1, S_1 \rightarrow S_3 and S_3 \rightarrow S_1$. Here, S_1 *and* S_3 belong to the same group P_1. States S_0, S_1 *and* S_3 can be combined.
1-successors – $S_0 \rightarrow S_2, S_1 \rightarrow S_5 and S_3 \rightarrow S_6$. Here, S_2, S_5 *and* S_6 belong to the same group P_2. States S_0, S_1 *and* S_3 can be combined.

Now consider the second partition P_2 and their k-successors are

0-successors – $S_2 \rightarrow S_5, S_4 \rightarrow S_5, S_5 \rightarrow S_4 and S_6 \rightarrow S_5$. Here, S_4 and S_5 belong to the same group P_2. States S_2, S_4, S_5 *and* S_6 can be combined.
1-successors – $S_2 \rightarrow S_4, S_4 \rightarrow S_2, S_5 \rightarrow S_3 and S_6 \rightarrow S_6$. Here, S_2, S_4 *and* S_6 belong to group P_2 but S_3 belong to the group P_1. Thus states S_2, S_4 *and* S_6 can be combined but S_5 is a different state and it is assigned to another partition P_3.

Third Pass: In the third pass we have three partitions $P_1 = S_0 S_1\,S_3$, $P_2 = S_2\,S_4\,S_6$ and $P_3 = S_5$. Same steps are followed in this pass also.

Consider the partition P_1 and their k-successors are
0-successors – The 0-successors are S_1 and S_3 which belong to same group.
1-successors – The 1-successors are S_2, S_5 *and* S_6. Here, S_2 and S_6 belong to P_2 but S_5 belong to P_3. Thus S_1 can not be combined with S_0 and S_3. The state S_1 is must be kept in another partition.

Similar analysis can be run for partition P_2 and P_3.

After the third pass , the partitions are updated as $P_1 = S_0\,S_3$, $P_2 = S_1$, $P_3 = S_2\,S_4\,S_6$ and $P_4 = S_5$. Further passes can be run but after the third pass there is no change in the partitions. Thus final states are same as result of the third pass. The state minimization result is same as the Implication chart produces as shown in Table 2.

present state	next state	Output
S0*	S1 S2	1
S1	S0* S5	1
S2*	S5 S2*	0
S5	S0* S0*	0

Table 1: Optimized State table

. **Kalman-Zerlegung (Kalman decomposition)** in einen *beobachtbaren* und einen *steuerbaren* Teil (LTI Systeme allgemein)

Die Beschreibung eines *zeitkontinuierlichen* linearen Systems lautet:

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t)$$
$$y(t) = C \cdot x(t) + D \cdot u(t)$$

mit
x ... "Zustandsvektor"
y ... "Ausgabevektor"
u ... "Eingabe- (Steuer-) Vektor"
A ... "Zustandsmatrix"
B ... "Eingabematrix"
C ... "Ausgabematrix"
D ... "Durchführungsmatrix"

In ähnlicher Weise kann ein *zeitdiskretes* lineares Steuersystem beschrieben werden als

$$x(k+1) = A \cdot x(k) + B \cdot u(k)$$
$$y(k) = C \cdot x(k) + D \cdot u(k)$$

mit ähnlichen Bedeutungen für die Variablen. Somit kann das System unter Verwendung des Tupels beschrieben werden, das aus vier Matrizen besteht. Dann wird die Kalman Zerlegung als eine Transformation des Tupels (A,B,C,D) definiert: $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$:

$$\hat{A} = TAT^{-1}$$
$$\hat{B} = TB$$
$$\hat{C} = CT^{-1}$$
$$\hat{D} = D$$

T^{-1} ... invertierbare Matrix $n \times n := \begin{bmatrix} T_{r\bar{o}} & T_{ro} & T_{\bar{r}\bar{o}} & T_{\bar{r}o} \end{bmatrix}$
r ... reachable (steuerbar), o ... obervable (beobachtbar)

mit

$T_{r\bar{o}}$ ... Matrix, deren Spalten den Unterraum erreichbarer, aber nicht beobachtbarer Zustände bilden.

T_{ro} ... Spalten bilden eine Basis für den erreichbaren Unterraum.

$T_{\overline{ro}}$ ... Spalten bilden eine Basis für den nicht beobachtbaren Unterraum: $\begin{bmatrix} T_{r\bar{o}} & T_{\overline{ro}} \end{bmatrix}$

$T_{\bar{r}o}$ ... so, dass $\begin{bmatrix} T_{r\bar{o}} & T_{ro} & T_{\overline{ro}} & T_{\bar{r}o} \end{bmatrix}$ invertierbar

Konstruktionsbedingt ist die Matrix invertierbar. Es kann beobachtet werden, dass einige dieser Matrizen die Dimension Null haben können. Wenn das System beispielsweise sowohl beobachtbar als auch steuerbar ist, werden die anderen Matrizen zur Dimension Null: $T^{-1} = \begin{bmatrix} T_{ro} \end{bmatrix}$

(https://de.wikibrief.org/wiki/Kalman_decomposition, 29Okt'22)

7.3 Beispiel: primitiv-Verkehrsampel

7.3.1 Zustands-Tabelle primitiv-Verkehrsampel

Zustand	Inputs		
	$zeit_1$	$zeit_2$	$zeit_3$
GnLight	GnBlnk	-%-	-%-
GnBlink	-%-	YeLight	-%-
YeLight	-%-	-%-	RdLight
RdLight	YeRd	-%-	-%-
YeRd	-%-	-%-	GnLight

Tabelle der Folgezustände 'z'

"-%-"... unverändert ≡ derselbe Zustand

Riesenvorteile dieser Tabellen:
- man muss überall was reinschreiben - ma kann nix vergessen
- maschinell hervorragend verarbeitbar

7.3.2 Output-Tabelle primitiv-Verkehrsampel

Zustand	Inputs		
	$zeit_1$	$zeit_2$	$zeit_3$
GnLight	oBlnk	oGn	oGn
GnBlink	oBlnk	oYe	oBlnk
YeLight	oYe	oYe	oRd
RdLight	oYe	oRd	oYeRd
YeRd	oYeRd	oYeRd	oGn

Tabelle der Outputs 'o'

Ein **Moore**-Automat

ist ein endlicher Automat, dessen Ausgabe im Gegensatz zu einem Mealy-Automaten **ausschließlich von seinem Zustand** abhängt. Beim Erreichen eines Zustandes wird eine Ausgabe erzeugt, welche unabhängig vom Übergang in diesen Zustand ist. Moore-Automaten sind nach dem Mathematiker Edward F. Moore (1925–2003) benannt *(und nicht nach Gordon Moore, dem Intel-Gründer, wie XH gern erzählt)*. Sie können deterministisch oder nichtdeterministisch sein (theoretisch!).

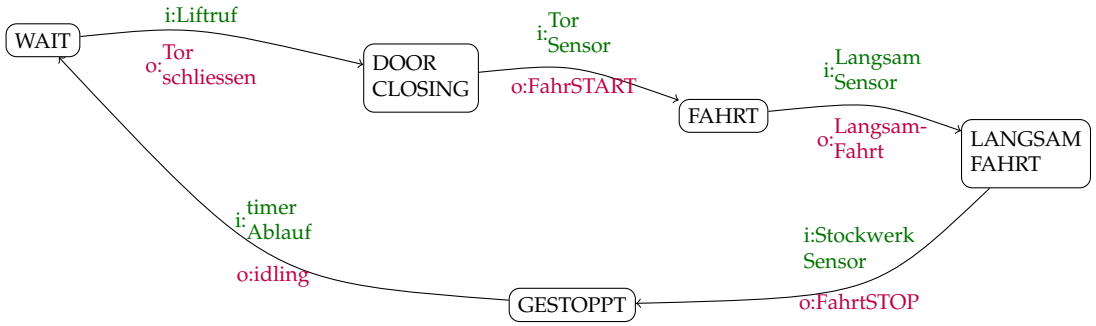
Eine Realisierung des Moore-Automaten ist mittels Digitaltechnik möglich. Hierfür sind zwei Schaltnetze und ein getakteter Speicherblock erforderlich. Neben den auf einer Leiterplatte verdrahteten Logikbausteinen erfolgt die Umsetzung häufig mittels programmierbarer Logik und Anwendung einer Hardwarebeschreibungssprache.

Die Verarbeitung mit Logikschaltkreisen erfordert die Umwandlung des Ein- und Ausgabealphabets in einen Binärcode.

Ein **Mealy**-Automat

ist ein deterministischer endlicher Automat, dessen Ausgabe von seinem Zustand **und** (im Gegensatz zu einem Moore-Automaten) **seiner Eingabe abhängt.** Anschaulich bedeutet das, dass jeder Kante im Zustandsdiagramm ein Ausgabewert zugeordnet wird. Der Name geht auf George H. Mealy (1927-2010) zurück, der für die Verwendung dieser Ausprägung eintrat.

7.4 Aufgaben - StateDG/BubbleDG



7.5 Aufgaben - Ampel

1. Zustände der Verkehrsampel mit Fußgängertaste

wie die mitarbeitende Bevölkerung der Klasse schon festgestellt hat, können externe Time-Counter auch mit in die FSM integriert werden, wodurch wir gewaltig mehr Zustände erhalten:

durchaus richtig, es so zu machen!

Besonders für automatisierten Entwurf — das Synthese-Tool hat so die größten Freiheiten

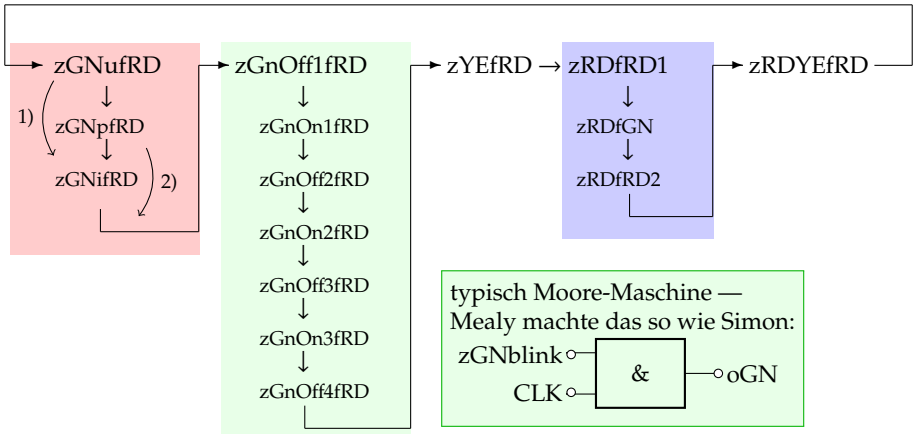
(amerikanische Lehre: *"Befolge unreflektiert stur die Methode"*)

Die Zustandstabelle wird riesengross . . .

Moore:
zGNtakt1
↓
zGNtakt2
↓
zGNtakt3
↓
. . . . . .
↓
zGNtakt_N
↓
zGNblinkTakt1
↓
zGNblinkTakt2
↓
usw.

Der europäische Weg verkürzt die Methode hingegen immer wieder durch Gehirn-Zwischenschaltung, fasst hier Ähnliches/Identisches zusammen und lagert es aus - wie hier diese Phasen-Zähler. Das verkompliziert uns aber den Fußgängertasten-Unterbrechungs-Mechanismus (womöglich?)

Wir machen es "europäisch":
Die **Moore**-Maschine
steuert den Output allein mit den Zuständen (ohne Inputs), braucht also entsprechend mehr Zustände.



1) Fußgängertaste-nicht-gedrückt:
der Zustand *zGNpfRD* speichert den Fußgängertastendruck im *GN_nicht_unterbrechbar* Zustand. Wenn die Fußgängertaste *nicht* gedrückt wird, ist nach *Timerablauf* der Folgezustand:*GN_unterbrechbar*
senkrecht: Übergang von Timerablauf ausgelöst
2) Fußgängertaste-gedrückt:
Wenn jedoch die Fußgängertaste im *GN_nichtunterbrechbar* Zustand gedrückt wurde, wird (nach Timerablauf) der *GN_unterbrechbar* Zustand gleich übersprungen

2. entwirf die Liste der Inputs der Verkehrsampel mit Fußgängertaste
   clock **CLK**
   "reset" **RES**
   Fußgängertaste **F**
   TimerAblauf-für-zGNu green, uninterruptible
   TimerAblauf-für-zGNi green, interruptible
   TimerAblauf-für-zGNp green, buttonPressed
   TimerAblauf-für-zYE
   TimerAblauf-für-zRDfRD1
   TimerAblauf-für-zRDfGN
   TimerAblauf-für-zRDfRD2
   TimerAblauf-für-zRDYE
3. entwirf die Liste der Outputs der Verkehrsampel mit Fußgängertaste
   · grün oGN
   · rot oRD
   · gelb oYE
   · Fußgänger rot foRD
   · Fußgänger grün foGN
4. entwirf die Zustandstablelle der Verkehrsampel mit Fußgängertaste

nach Moore	nach Mealy
zGN	zGN
zGNoff1	|
zGNon1	|
zGNoff2	|
zGNon2	|
zGNoff3	|
zGNon3	|
zGNoff4	|
zYE	zYE
zRDfRD1	zRD
zRDfGN	|
zRDfRD2	|
zRDYE	zRDYE

   Die ganze Zustandstabelle hat natürlich soviele Spalten wie Inputs und Zeilen wie Zustände.
5. Nummeriere die Zustände mit Binärzahlen
   Dabei kann man *schlau* vorgehen und die Bits/Flipflops *geschickt* in unabhängige Gruppen zB. Counter aufteilen *(oder eben amerikanisch)*.
6. entwirf die Outputtablelle der Verkehrsampel mit Fußgängertaste

...						
zRDYE	...	oRD oYE ofRD	...			
...						
zRdfRD1	...		oRD ofRD	...		
zRdfGN	...			oRD ofGN	...	
zRdfRD2	...				oRD ofRD	...
...						

7. entwirf die Input- und Output- Schaltnetze der Verkehrsampel mit Fußgängertaste nach Moore

(kein FSM-Input) und nach Mealy (incl.FSM-Inputs)

Kombinatorische Schaltnetze entwirft man mit Wahrheitstabelle (truth table), Minimierungsverfahren (minimization) (Methode der Minterme, Quine-McCluskey-Verfahren, KV-Diagramm), Boolescher Gleichung, Gatterschaltplan (nicht bei bei VHDL/Verilog- Entwurf).

8. codiere die FSM der Verkehrsampel mit Fußgängertaste unter Verwendung der erstellten Zustands- und Outputtabellen in 'C' für das rPi (Raspberry Pi)

```
#include <stdio.h>
int main(int argc, char argv[]){
  ...
    (jetzt Du:)
  ...
}
```

9. codiere die FSM der Verkehrsampel mit Fußgängertaste in HDL (VHDL, Verilog).

```
Entity Ampel is PORT(
  Reset:        in      std_logic;
  NotAus:       in      std_logic;
  Alarm:        in      std_logic;
  clock:        in      std_logic;
);
Architecture Core of Ampel is
begin
  -- (jetzt Du:)
end ALU;
```

7.6 Aufgaben Teilnehmer-Beiträge

Listing 1: Aufzug Teilnehmerbeiträge

```
Input Tabelle:
        1a) Knopf 4.OG
        1b) Knopf 3.OG
        1c) Knopf 2.OG
        1d) Knopf 1.OG
        1e) Knopf EG
        1f) Knopf UG
        1g) Tueroeffner
        1h) Tuerschliesser
        1i) Notfallglocke

        2a) 4.OG ruf/runter
        2b) 3.OG ruf/rauf
        2c) 3.OG ruf/runter
        2d) 2.OG ruf/rauf
        2e) 2.OG ruf/runter
        2f) 1.OG ruf/rauf
        2g) 1.OG ruf/runter
        2h) EG ruf/rauf
        2i) EG ruf/runter
        2j) UG ruf/rauf

        3a) Ueberlastungsensor

        4a) 4.OG Tuer offen?
        4b) 4.OG lift anwesend?
        4c) 3.OG Tuer offen?
        4d) 3.OG lift anwesend?
        4e) 2.OG Tuer offen?
        4f) 2.OG lift anwesend?
        4g) 1.OG Tuer offen?
        4h) 1.OG lift anwesend?
        4i) EG Tuer offen?
        4j) EG lift anwesend?
        4k) UG Tuer offen?
        4l) UG lift anwesend?

        5a) Ueberlastung

        6a) Timer

Output Tabelle:
        1a)winde rauf
        1b)winde runter

        2a) 4.OG Pfeil runter
        2b) 4.OG aktuelles Stockwerk
        2c) 3.OG Pfeil runter
        2d) 3.OG Pfeil rauf
        2e) 3.OG aktuelles Stockwerk
        2f) 2.OG Pfeil runter
        2g) 2.OG Pfeil rauf
        2h) 2.OG aktuelles Stockwerk
        2i) 1.OG Pfeil runter
        2j) 1.OG Pfeil rauf
        2k) 1.OG aktuelles Stockwerk
        2l) EG Pfeil runter
        2m) EG Pfeil rauf
        2n) EG aktuelles Stockwerk
        2o) UG Pfeil rauf
        2p) UG aktuelles Stockwerk

        3a) alarm

        4a) ueberlasthorn

        5a) not - aus

Zustaende:
        4.OG geparkt tuer offen
        4.OG geparkt tuer zu
        zwischen 4.OG und 3.OG aufwaerts
        zwischen 4.OG und 3.OG abwaerts
        3.OG geparkt tuer offen
        3.OG geparkt tuer zu
        zwischen 3.OG und 2.OG aufwaerts
        zwischen 3.OG und 2.OG abwaerts
        2.OG geparkt tuer offen
        2.OG geparkt tuer zu
        zwischen 2.OG und 1.OG aufwaerts
        zwischen 2.OG und 1.OG abwaerts
        1.OG geparkt tuer offen
        1.OG geparkt tuer zu
        zwischen 1.OG und   EG aufwaerts
        zwischen 1.OG und   EG abwaerts
          EG geparkt tuer offen
          EG geparkt tuer zu
        zwischen   EG und   UG aufwaerts
        zwischen   EG und   UG abwaerts
          UG geparkt tuer offen
          UG geparkt tuer zu

        gestoppt wegen ueberlastung
        gestoppt wegen not-aus


        Name: Guenther 2\_2
Datum: 20.10.2020

1. Windows Editor

2. Liste Inputs:

   a) -1.Stock
      1.Stock
      2.Stock
      3.Stock
      4.Stock
      Erdgeschoss^...
      Tr ^ffnen^...
      Tr schlie"sen

   b) Not Aus Knopf in der Kabine

   c) Buttons:
      Notfall Knopf (Alarm)
      -1.Stock
      1.Stock
      2.Stock
      3.Stock
      4.Stock
      Erdgeschoss^...
      Tr ^ffnen^...
      Tr schlie"sen

   d) Buttons:
      Rauf
      Runter
```

e), f), g) Sensoren:
Waage

h) Timer:^Ö
ffnen (ca. 5sec) und schlie"sen (ca. 5sec) der ^...Tr

3. Liste Outputs:

Windantrieb: ? (NICHT VERSTANDEN)

Anzeigelampen:
Der Knopf leuchtet wenn man sich in diesem Stockwerk befindet oder
wenn man den Knopf ^...gedrckt hat und zu diesem Stockwerk ^/fhrt.

Alarmton:
Notruf

4. Liste der Zustande:

fahren
stehenbleiben^¶
ffnen
schlie"sen

5. Zustandstabelle:

Zustand	Input
stehenbleiben	Lift steht^¶
ffnen ^...	Tr wird ^¶geffnet
schlie"sen ^...	Tr wird geschlossen
fahren	Lift ^/fhrt

6. Outputtabelle

Zustand	Input
Notruf	Notruftaste wird ^...gedrckt
stehenbleiben	Lift beibt stehen
fahren ^/	fhrt^¶
ffnen	Ankunft
schlie"sen	timer geht an

Dic am 20.10.2020

Atsi 3_2

2. Liste Inputs:

1.Stock
2.Stock
3.Stock
Erdgeschoss^...
Tr ^¶ffnen^...
Tr schlie"sen

Buttons:
Rauf
Runter

Sensoren:
Waage

Timer:
offnen und schlie"sen der ^...Tr

3. Liste Outputs:

Anzeigelampen:
Anzeige in welchen Stock man sich befindet
Anzeige welcher Knopf ^...gedrckt wurde(Stockwerk)

Alarmton:
Notruf

4. Liste der Zustande:

Lift im EG
Lift im 1.Stock
Lift im 2.Stock
Lift im 3.Stock
Button 1.Stock
Button 2.Stock
Button 3.Stock
Button EG
Lift ^/fhrt
Lift steht^...
Tr ^¶ffnen^...
Tr schlie"sen

5. Zustandstabelle:

Zustand	Input
steht	Lift steht
Button 1.Stock ^/	fhrt in 1.Stock
Button 2.Stock ^/	fhrt in 2.Stock
Button 3.Stock ^/	fhrt in 3.Stock
Button EG ^/	fhrt in EG^¶
ffnen ^...	Tr wird ^¶geffnet
schlie"sen ^...	Tr wird geschlossen^/
fhrt	Lift ^/fhrt

6. Outputtabelle

Zustand	Input
steht	Lift steht
Button 1.Stock	Button 1.Stock wird ^...gedrckt
Button 2.Stock	Button 2.Stock wird ^...gedrckt
Button 3.Stock	Button 3.Stock wird ^...gedrckt
Button EG	Button EG wird ^...gedrckt^/
fhrt ^/	fhrt^¶
ffnen	Ankunft
schlie"sen	timer geht an
Notruf	Notruftaste wird ^...gedrckt

Also, da du gesagt hast das wir die Listen das ^/nchste mal machen, habe ich diese hier nicht gemacht.

Der Lift:^/

Oberflchlich: Auf Lift warten => Einsteigen => warten bis er an den Ankunftsort ankommt => Aussteigen

zum Detail: Zum Lift gehen => Auf den Button ^...drcken(Input) => Warten bis er Ankommt(Output) => Warten bis die ^...Tren ^¶ffnen => Warten bis diejenigen im Lift aussteigen => Einsteigen => Bestimmungsort (Stockwerk) ^/whlen(Input)
=> Warten bis der Aufzug es einspeichert => Warten bis die ^...Tren nicht blockiert sind und schliesen(Output) => Warten bis der Lift ^/losfhrt => Warten bis er langsamer wird und Ankommt
=> Warten bis er stehn bleibt => Warten bis sich die ^...Tren ^¶ffnen(Output) => Aussteigen^/

Zustnde: ^...Tr aufmachen=> ^...Tr offen halten=> ^...Tr Blockiert= ^...Tr offen halten, ^...Tr Frei= ^...Tr schliesen=> warten=> Stockwerksensoren=> Losfahren=> Stockwerksensoren=> Anhalten=> warten=> ^...Tr ^¶ffnen
Ausgabe: ^...Tr auf ^... Tr zu Stockwerk: Stockwerk:

I habs hier mal versucht. Die Ausgabe(output) hab i da hingeschrieben, wo es bei den ^/Zustnden passiert.
Crewmate

Kunibert 3_4 3aHEL

Liste der Inputs:

1.Stock
2.Stock
3.Stock
Erdgeschoss^...
Tr ^¶ffnen^...
Tr schliesen

Buttons:
Rauf
Runter

Sensoren:
Waage

Timer:^¶
ffnen und schliesen der ^...Tr

Liste der Outputs:

Anzeigelampen:
Anzeige in welchen Stock man sich befindet

Alarmton:
Notruf

Liste der ^/Zustnde:

Lift im EG
Lift im 1.Stock
Lift im 2.Stock
Lift im 3.Stock
Button 1.Stock
Button 2.Stock
Button 3.Stock
Button EG^/
fhrt
steht^¶
ffnen
schliesen

Zustandstabelle:

Zustand

steht
Button 1.Stock
Button 2.Stock
Button 3.Stock
Button EG^¶
ffnen
schliesen^/
fhrt

Input

Lift steht^/
fhrt in 1.Stock^/
fhrt in 2.Stock^/
fhrt in 3.Stock^/
fhrt in EG^...
Tr wird ^¶geffnet^...
Tr wird ^¶geffnet
Lift ^/fhrt

Outputtabelle:

Zustand

steht
Button 1.Stock
Button 2.Stock
Button 3.Stock
Button EG^/
fhrt^¶
ffnen
schliesen
Notruf

Input

Lift steht
Button 1.Stock wird ^...gedrckt
Button 2.Stock wird ^...gedrckt
Button 3.Stock wird ^...gedrckt
Button EG wird ^...gedrckt^/
fhrt
Ankunft
timer geht an

Notruftaste wird ^...gedrckt

FSM Elevator
Datum: Heute, 17:11:46 CEST
Von: Klampfe 1_3
Hallo Christoph,

anbei mein bisheriges Ergebnis, wohl eher die "amerikanische" Variante.
Ich bin leider noch nicht zur gesamten Tabelle gekommen, arbeite aber fleisig weiter und reiche sie nach!
Vielleicht habe ich das gesamte Projekt auch zu verkopft realisiert, deswegen will ich zuerst noch abwarten,
ob ^...Mitschler (vA Lulatsch) effizientere ^/Einflle als ich haben und mich dann an die komplette Darstellung wagen.

Liebe ^...Grse,
Stanislaus

Design – Draft
Steuerung fuer den Innrain–Aufzug

2. Inputs:

Knoepfe:

a) In den Stockwerken:
* iUGRufen
* iEGRufen
* iERSTRufen
* iZWEITRufen
* iDRITTRufen
* iVIERTRufen

b) In der Kabine:
* iWillUG
* iWillEG
* iWillERST
* iWillZWEIT
* iWillDRITT
* iWillVIERT
* iNOTFALL

Sensoren:

c) In der Kabine:
* iLichtschrankeTuer
* iTuerwiderstand
* iKabTuerZu

d) In den Stockwerken:

Verlangsamung bei Stop bei Aufwaertsfahrt:
* iUGtoEGVerlangsamung //(beiWillEG)
* iEGtoERSTVerl //(beiWillERST)
* iERSTtoZWEITVerl //(usw...)
* iZWEITtoDRITTVerl
* iDRITTtoVIERTVerl

Verlangsamung bei Stop bei Abwaertsfahrt:
* iVIERTtoDRITTVerl
* iDRITTtoZWEITVerl
* iZWEITtoERSTVerl
* iERSTtoEGVerl
* iEGtoUGVerl

WoBinIch?:
* ibinUG
* ibinEG
* ibinERST
* ibinZWEIT
* ibinDRITT
* ibinVIERT

Tueren:
* iTuerwiderstandUG
* iTuerwiderstandEG
* iTuerwiderstandERST
* iTuerwiderstandZWEIT
* iTuerwiderstandDRITT
* iTuerwiderstandVIERT
* iTuerZuUG
* iTuerZuEG
* iTuerZuERST
* iTuerZuZWEIT
* iTuerZuDRITT
* iTuerZuVIERT

e) Seilwindensensoren
* iUeberlast
* iAufwaertsfahrt
* iAbwaertsfahrt

f) Timer:
* iTueroeffnungsdauer 10s
* iAnfahrts– und Tueroeffnungsverzoegerung 2s

3. Outputs
a) Windenantrieb
* oBergaufSchnell
* oBergaufLangsam
* oSTOP
* oBergabSchnell
* oBergabLangsam

b) Anzeigelampen
* oLampeBinUG
* oLampeBinEG
* oLampeBinERST
* oLampeBinZWEIT
* oLampeBinDRITT
* oLampeBinVIERT

* oLampeWillUG
* oLampeWillEG
* oLampeWillERST
* oLampeWillZWEIT
* oLampeWillDRITT
* oLampeWillVIERT

c) Alarmton
* oTon

```
        d) Ueberlasthorn
        * oHorn

        e) Not-Aus Stromabschalter
        * oNOTAUS


        Weiters:
        * oTuerzuKabine
        * oTueraufKabine
        * oTuerAufUG
        * oTuerAufEG
        * oTuerAufERST
        * oTuerAufZWEIT
        * oTuerAufDRITT
        * oTuerAufVIERT
        * oTuerZuUG
        * oTuerZuEG
        * oTuerZuERST
        * oTuerZuZWEIT
        * oTuerZuDRITT
        * oTuerZuVIERT^...


        Fr Data:
        * oWillUG
        * oWillEG
        * oWILLERST
        * oWILLZWEIT
        * oWILLDRITT
        * oWILLVIERT


4.) Liste der Zustaende

Zustand:

        * ZFahreEGAufw
        * ZFahreERSTAufw
        * ZFahreZWEITAufw
        * ZFahreDRITTAufw
        * ZFahreVIERTAufw

        * ZFahreUGAbw
        * ZFahreEGAbw
        * ZFahreERSTAbw
        * ZFahreZWEITAbw
        * ZFahreDRITTAbw

        * ZFahreEGAufwLangs
        * ZFahreERSTAufwLangs
        * ZFahreZWEITAufwLangs
        * ZFahreDRITTAufwLangs
        * ZFahreVIERTAufwLangs

        * ZFahreUGAbwLangs
        * ZFahreEGAbwLangs
        * ZFahreERSTAbwLangs
        * ZFahreZWEITAbwLangs
        * ZFahreDRITTAbwLangs

        * ZBinUGWarte0
        * ZBinUGTuerauf
        * ZBinUGWarteEingabe
        * ZBinUGWarte1          //Speichern der Eingabe in D-FF
        * ZBinUGTuerzu
        * ZBinUGWarte2          //Auslesen der Eingabe von D-FF
                * Daraus ZFahre-Zustand ^/whlen, bei Ankuft Reset auf alle D-FF senden


        * ZBinEGWarte0
        * ZBinEGTuerauf
        * ZBinEGWarteEingabe
        * ZBinEGWarte1          //Speichern der Eingabe in D-FF
        * ZBinEGTuerzu
        * ZBinEGWarte2          //Auslesen der Eingabe von D-FF
                * Daraus ZFahre-Zustand ^/whlen, bei Ankuft Reset auf alle D-FF senden


        * ZBinERSTWarte0
        * ZBinERSTTuerauf
        * ZBinERSTWarteEingabe
        * ZBinERSTWarte1          //Speichern der Eingabe in D-FF
        * ZBinERSTTuerzu
        * ZBinERSTWarte2          //Auslesen der Eingabe von D-FF
                * Daraus ZFahre-Zustand ^/whlen, bei Ankuft Reset auf alle D-FF senden

        * ZBinZWEITWarte0
        * ZBinZWEITTuerauf
        * ZBinZWEITWarteEingabe
        * ZBinZWEITWarte1          //Speichern der Eingabe in D-FF
        * ZBinZWEITTuerzu
        * ZBinZWEITWarte2          //Auslesen der Eingabe von D-FF
                * Daraus ZFahre-Zustand ^/whlen, bei Ankuft Reset auf alle D-FF senden


        * ZBinDRITTWarte0
        * ZBinDRITTTuerauf
        * ZBinDRITTWarteEingabe
        * ZBinDRITTWarte1          //Speichern der Eingabe in D-FF
        * ZBinDRITTTuerzu
        * ZBinDRITTWarte2          //Auslesen der Eingabe von D-FF
                * Daraus ZFahre-Zustand ^/whlen, bei Ankuft Reset auf alle D-FF senden


        * ZBinVIERTWarte0
        * ZBinVIERTTuerauf
        * ZBinVIERTWarteEingabe
        * ZBinVIERTWarte1          //Speichern der Eingabe in D-FF
        * ZBinVIERTTuerzu
        * ZBinVIERTWarte2          //Auslesen der Eingabe von D-FF
                * Daraus ZFahre-Zustand ^/whlen, bei Ankuft Reset auf alle D-FF senden


        * ZNOTFALL
                -> Weitere FSM, ^...fr Anrufe, Fernwartung, /*Jodelautomat*/


        * ZUeberUG
        * ZUeberEG
        * ZUeberERST
        * ZUeberZWEIT
        * ZUeberDRITT
        * ZUeberVIERT




Detaillierung:
        Beipiel
```

```
Lift wartet mit offenen ^...Tren, wird gerufen
        * ZBinVIERTWarteEingabe
                * wenn Ueberlast => ZUeberViert, Ueberlasthorn
                        * wenn keine Ueberlast mehr => ZBinVIERTWarteEingabe
                * iUGRufen, speichern mit oWillEG auf Data-FF
        * ZBinVIERTWarte1
        * ZBinVIERTTuerzu
                * oTuerZuVIERT
                * oTuerzuKabine
        * ZBinVIERTWarte2
        * ZFahreUGAbw
                * ibinDRITT Reset der Licht-D-FF -> Speichern Licht-Data-FF -> oLampeBinDRITT
                * ibinZWEIT Reset der Licht-D-FF -> Speichern Licht-Data-FF -> oLampeBinZWEIT
                * ibinERST  Reset der Licht-D-FF -> Speichern Licht-Data-FF -> oLampeBinERST
                * ibinEG    Reset der Licht-D-FF -> Speichern Licht-Data-FF -> oLampeBinEG
                * iEGtoUGVerl
        * ZFahreUGAbwLangs
                * ibinUG    Reset der Licht-D-FF -> Speichern Licht-Data-FF -> oLampeBinUG
        * ZFahreUGAbwLangs
                * ibinEG
        * ZBinUGWarte0
        * ZBinUGTuerauf
                oTuerAufUG
                oTueraufKabine
        * ZBinUGWarteEingabe
Input:

Ruf-~¶Knpfe in allen Stockwerken^¶
Steuerungsknpfe im Aufzug
Notfallknopf im Aufzug
Sensor(^...Tr offen oder Zu)
Sensor(Maximal Gewicht)

Output:
Alarm-Sirene
Aufzugsmotoren
Anzeige im Aufzug^/

Zustnde:^/
fhrt hoch^/
fhrt runter
steht^...
tr offen^...
tr zu
overweight-Zustand
Feueralarm-Stromausfall




matA 3aHEL 3\_3 | DIC


 Liste der Inputs:

   1.Stock
   2.Stock
   3.Stock
   Erdgeschoss^...
   Tr ~¶ffnen^...
   Tr schliesen

   Buttons:
   Rauf
   Runter

   Sensoren:
   Waage

   Timer:~¶
   ffnen und schliesen der ^...Tr


 Liste der Outputs:


   Anzeigelampen:
   Anzeige in welchen Stock man sich befindet

   Alarmton:
   Notruf


 Liste der ^/Zustnde:

   Lift im EG
   Lift im 1.Stock
   Lift im 2.Stock
   Lift im 3.Stock
   Button 1.Stock
   Button 2.Stock
   Button 3.Stock
   Button EG~/
   fhrt
   steht~¶
   ffnen
   schliesen


 Zustandstabelle:

          Zustand

        steht
        Button 1.Stock
        Button 2.Stock
        Button 3.Stock
        Button EG~¶
        ffnen
        schliesen^/
         fhrt


          Input

        Lift steht^/
        fhrt in 1.Stock^/
        fhrt in 2.Stock^/
        fhrt in 3.Stock^/
        fhrt in EG~...
        Tr wird ~¶geffnet^...
        Tr wird ~¶geffnet
        Lift ~/fhrt


Outputtabelle:

        Zustand
```

steht
Button 1.Stock
Button 2.Stock
Button 3.Stock
Button EG˜/
fhrt˜¶
ffnen
schliesen
Notruf

Input

Lift steht
Button 1.Stock wird ˆ...gedrckt
Button 2.Stock wird ˆ...gedrckt
Button 3.Stock wird ˆ...gedrckt
Button EG wird ˆ...gedrckt˜/
fhrt
Ankunft
timer geht an
Notruftaste wird ˆ...gedrckt

DiC 19.10.2020 – Onkel, 3aHEL 1_4

Meine Ideen ˆ...fr den Entwurf: AUFZUG

1. Liste an Inputs:

a) Buttons in der Kabine:
* 1.OG (Obergeschoss)
* 2.OG
* 3.OG
* 4.OG
* EG (Erdgeschoss)
* KG (Kellergeschoss)
* Notruf–taste (Glockentaste)
* ˆ...Tre ˜¶ffnen
* ˆ...Tre schliesen

b) Buttons in den Stockwerken:
* Hinauf (Ausnahme: 4.OG)
* Hinunter (Ausnahme: KG)

c) Sensoren in der Kabine:
* ˆ...Trsensor (ˆ...Tr ˜¶geffnet/geschlossen)
* Waage (als Schutz gegen ˜Þberbelastung)

d) Sensoren an der Seilwinde:
* Schutz gegen zu hohe Drehzahl

e) Timer:
* Timer zum Schliesen der ˆ...Tren

f) Notaus/ Stromabschalter

2. Liste an Outputs:

a) Windenantrieb: bewegt den Aufzug

b) Anzeigelampen:
* 7–Segmentanzeige (Anzeigen des Stockwerks in der Kabine)
* Lampen in den Stockwerken (Signal: Lift ˜/fhrt rauf oder runter)
* ˜/gewhlte Tasten (Kabine und in den Stockwerken)

c) Alarmton/ Signalton:
* Ankunft im jeweiligen Stockwerk
* Notruftaste ˆ...gedrckt
* Alarm bei ˜Þberlastung

3. Liste an ˜/Zustnden:

* Lift steht im KG
* Lift steht im EG
* Lift steht im 1.OG
* Lift steht im 2.OG
* Lift steht im 3.OG
* Lift steht im 4.OG
* Lift ˜/fhrt
* Button "runter" in einzelnen Stockwerken ˆ...gedrckt
* Button "hinauf" in einzelnen Stockwerken ˆ...gedrckt
* Button "KG" ˆ...gedrckt
* Button "EG" ˆ...gedrckt
* Button "1.OG" ˆ...gedrckt
* Button "2.OG" ˆ...gedrckt
* Button "3.OG" ˆ...gedrckt
* Button "4.OG" ˆ...gedrckt
* Notruftaste ˆ...gedrckt
* Button: "ˆ...Tren schliesen" ˆ...gedrckt
* Button: "ˆ...Tren ˜¶ffnen" ˆ...gedrckt
* Notaus
* ˜Þberbelastung
* Lift bleibt stecken
* Lift zu schnell

4. Zustandstabelle: (Tabelle der ˜/Folgezustnde)

Zustand	Inputs
Lift steht	–%–
Button KG ˆ...gedrckt	Fahrstuhl ˜/fhrt in KG
Button EG ˆ...gedrckt	Fahrstuht ˜/fhrt in EG
Button 1.OG ˆ...gedrckt	Fahrstuhl ˜/fhrt in 1.OG
Button 2.OG ˆ...gedrckt	Fahrstuhl ˜/fhrt in 2.OG
Button 3.OG ˆ...gedrckt	Fahrstuhl ˜/fhrt in 3.OG
Button 4.OG ˆ...gedrckt	Fahrstuhl ˜/fhrt in 4.OG
Button "ˆ...Tren ˜¶ffnen"	ˆ...Tren werden ˜¶geffnet
Button "ˆ...Tren schliesen"	ˆ...Tren werden geschlossen
Button–ausen: "Hinauf"	Fahrstuhl ˜/fhrt zum Kunden
Button–ausen: "Hinab"	Fahrstuhl ˜/fhrt zum Kunden
Notruftaste ˆ...gedrckt	Notruf wird ˜/gewhlt
Notruf ˜/gewhlt	–%–
Notaus	–%–
Lift ˜/fhrt	Ankunft im jeweiligen Stockwerk
Lift angekommen	ˆ...Tren ˜¶ffnen sich, Signalton ˆ...
Tren ˜¶geffnet	ˆ...Tren schliesen nach bestimmter Zeit ˆ...
Tren geschlossen	–%–˜Þ
berbelastung	Fehlersignal, Lift bleibt stehen
Lift zu schnell	Notstop bzw. bremsen

5. Outputtabelle: (Tabelle der Outputs)

Zustand	Inputs

```
_____________________|_____________________________________________
Lift steht           |      -%-
Button KG ^...gedrckt      |      Button KG wird ^...gedrckt
Button EG ^...gedrckt      |      Button EG wird ^...gedrckt
Button 1.OG ^...gedrckt    |      Button 1.OG wird ^...gedrckt
Button 2.OG ^...gedrckt    |      Button 2.OG wird ^...gedrckt
Button 3.OG ^...gedrckt    |      Button 3.OG wird ^...gedrckt
Button 4.OG ^...gedrckt    |      Button 4.OG wird ^...gedrckt
Button "^...Tren ^¶ffnen"  |      Button "^...Tren ^¶ffnen" wird ^...gedrckt
Button "^...Tren schliesen"|      Button "^...Tren schliesen" wird ^...gedrckt
Button-ausen: "Hinauf"     |      Button "Hinauf" wird ausen ^...gedrckt
Button-ausen: "Hinab"      |      Button "Hinab" wird ausen ^...gedrckt
Lift ^/fhrt          |      Stockwerk via Buttons ^/gewhlt
Lift angekommen      |      -%-^...
Tren ^¶geffnet       |      Ankunft Stockwerk, ^...^¶Trffner-Button^...
Tren geschlossen     |      Timersignal, ^...Trschlieser-Button
Notaus               |      ^Þberbelastung
Notruf ^/gewhlt      |      Notruftaste ^...gedrckt^Þ
berbelastung         |      Waage erkennt ^Þbergewicht
Lift zu schnell      |      Sensor am Antrieb

"-%-" ... meint Zustand ^/unverndert


6. Design draft: (Detaillierung)

a) Zustand:     Kunde hat Stockwerk via. Buttons ^/gewhlt

[Stockwerk ^/gewhlt]
        |
        v
[Belastung bzw. Gewicht ok]
        |
        v
[^...Tren werden geschlossen]
        |
        v
[Sensor ^...prft, ob ^...Tren geschlossen]
        |
        v
[Motor bewegt den Aufzug, ^/fhrt los]
        |
        v
[Aufzug ^/fhrt normales Tempo]
        |
        v
[Sensor erkennt Ankunft vom Lift (in den jeweiligen Stockwerken)]
        |
        v
[Lift bremst ab, ^/fhrt langsamer]
        |
        v
[Lift kommt am Wunschort an, bleibt stehen]
        |
        v
[Signalton, optische Signale (Anzeigen)]
        |
        v
[Timer zum ^Öffnen der ^...Tren]
        |
        v
[^...Tren ^¶ffnen sich]
        |
        v
[Kunde kommt an]

b) Zustand:     Lift wird zum Kunden gerufen (Buttons in den Stockwerken ^...gedrckt)

[Button ausen in den Stockwerken ^...gedrckt]
        |
        v
[Belastung bzw. Gewicht ok]
        |
        v
[^...Tren schliesen (wenn noch nicht geschlossen)]
        |
        v
[Sensor ^...prft, ob ^...Tren geschlossen]
        |
        v
[Motor bewegt Aufzug, ^/fhrt zum Kunden]
        |
        v
[Aufzug ^/fhrt normales Tempo]
        |
        v
[Sensor erkennt Ankunft (in den jeweiligen Stockwerken)]
        |
        v
[Lift bremst ab, ^/fhrt langsamer]
        |
        v
[Lift kommt am Wunschort an, bleibt stehen]
        |
        v
[Signalton, optische Signale (Anzeigen)]
        |
        v
[Timer zum ^Öffnen der ^...Tren]
        |
        v
[^...Tren ^¶ffnen sich]
        |
        v
[Lift bereit ^...fr den Kunden, kann ihn betreten]

c) Zustand bzw. Situation:
```

```
-->Kunde ^¶mchte vom EG in den 3.OG,
   Lift befindet sich im 4.OG

(^/vollstndige Simulation)

[Kunde ^/whlt Button ausen im Stockwerk, in dem er sich befindet]
        |
        v
[Belastung bzw. Gewicht ok]
        |
        v
[^...Aufzugtren schliesen (wenn noch nicht geschlossen)]
        |
        v
[Sensor ^...prft, ob ^...Tren geschlossen]
        |
        v
[Motor bewegt den Lift, ^/fhrt los]
        |
        v
[Lift ^/fhrt normales Tempo]
        |
        v
[Sensor erkennt Ankunft (in den jeweiligen Stockwerken)]
        |
        v
[Lift bremst ab, ^/fhrt langsamer]
        |
        v
[Lift kommt beim Kunden an, bleibt stehen]
        |
        v
[Signalton, optische Signale (Anzeigen)]
        |
        v
[Timer zum ^Öffnen der ^...Tren]
        |
        v
[^...Tren ^¶ffnen sich]
        |
        v
[Kunde betritt den Aufzug]
        |
        v
[Kunde ^/whlt Stockwerk (3.OG)]
        |
        v
[Belastung bzw. Gewicht ok]
        |
        v
[^...Tren werden geschlossen]
        |
        v
[Sensor ^...prft, ob ^...Tren geschlossen]
        |
        v
[Motor bewegt den Lift, ^/fhrt los]
        |
        v
[Lift ^/fhrt normales Tempo]
        |
        v
[Sensoren erkennen Ankunft (in den jeweilgen Stockwerken)]
        |
        v
[Lift bremst ab, ^/fhrt langsamer]
        |
        v
[Ankunft am Wunschort, bleibt stehen]
        |
        v
[Signalton, optische Signale (Anzeigen)]
        |
        v
[Timer zum ^Öffnen der ^...Tren]
        |
        v
[^...Tren werden ^¶geffnet]
        |
        v
[Kunde ^/verlsst den Aufzug]
        |
        v
[Timer zum Schliesen der ^...Tren]
        |
        v
[^...Tren geschlossen]

**Fehler und ^...Irrtmer vorbehalten!Inputs:
        a) Knopf 4
                Knopf 3
                Knopf 2
                Knopf 1
                Knopf EG
                Knopf UGþü
                Trffnerþ
                Trschliesser
                Notfall
```

```
        2b) 4 runter
                3  runter
                2  runter
                1  runter
                EG runter
                3  rauf
                2  rauf
                1  rauf
                EG rauf
                UG rauf

        2e)     ueberlastung

        2f)     timer

Output:
        3a)     Winde rauf
                Winde runter

        3b)     4  ruf licht
                3  ruf licht
                2  ruf licht
                1  ruf licht
                EG ruf licht
                UG ruf licht

        3c)     alarmton

        3d)     ueberlastton

        3e)     not stop

Zustandstabelle:
```

```
        Kasimir 2/_1
Dic, 20.10.2020
1. Windows Editor

2. Liste Inputs:

   1.Stock
   2.Stock
   3.Stock
   Erdgeschoss^...
   Tr ^ffnen^...
   Tr schliesen

   Buttons:
   Rauf
   Runter

   Sensoren:
   Waage

   Timer:
   offnen und schliesen der ^...Tr

3. Liste Outputs:

   Windantrieb: ? (keine Ahnung)

   Anzeigelampen:
   Anzeige in welchen Stock man sich befindet

   Alarmton:
   Notruf

4. Liste der Zustande:

   Lift im EG
   Lift im 1.Stock
   Lift im 2.Stock
   Lift im 3.Stock
   Button 1.Stock
```

```
   Button 2.Stock
   Button 3.Stock
   Button EG^/
   fhrt
   steht^¶
   ffnen
   schliesen

5. Zustandstabelle:

      Zustand                      Input

   steht                          Lift steht
   Button 1.Stock ^/               fhrt in 1.Stock
   Button 2.Stock ^/               fhrt in 2.Stock
   Button 3.Stock ^/               fhrt in 3.Stock
   Button EG ^/                    fhrt in EG^¶
   ffnen ^...                       Tr wird ^¶geffnet
   schliesen ^...                    Tr wird geschlossen^/
   fhrt                           Lift ^/fhrt


6. Outputtabelle

     Zustand                      Input

   steht                          Lift steht
   Button 1.Stock                 Button 1.Stock wird ^...gedrckt
   Button 2.Stock                 Button 2.Stock wird ^...gedrckt
   Button 3.Stock                 Button 3.Stock wird ^...gedrckt
   Button EG                      Button EG wird ^...gedrckt^/
   fhrt ^/                        fhrt^¶
   ffnen                         Ankunft
   schliesen                      timer geht an
   Notruf                         Notruftaste wird ^...gedrckt


Name: Kasimir 2\_1


Dic Arbeitsauftrag 20.10.2020 3\_1 3AHEL
 z:...Zustand
 o:...Output
 i:...Input
1. Liste der Inputs
 :Fahrtwunsch
 :Lichschrankenfrei
 :StockwerksensorA
 :Welchen Stock ich hin will
2. Liste der ^/Zustnde
 :idle am Anfnag
 :idle am Ende
 :^...Tr Blockiert
 :^...Tr wird Geschlossen
 :Warten1
 :Langsam oder Schnell fahren
 :i:StockwerksensorB
 :o:anhalten
 :warten2
 :^...Tr wird ^¶geffnet
 :idle
3. Liste der Output
 :Windenantrieb (capstan)
 :Anzeigelampe (display lamps)in welchen Stockwerk wir uns befinden oder der Aufzug
 :Alarmton wenn der Lift stecken geblieben ist^Þ
 :berlasthorn (overload horn) wenn der Lift ^...ber sein maximale ^/Gewichtszulssigkeit ist
 :Not-aus Stromabschalter (emergency circuit breaker) nur in den Ernst ^/fllen benutzen


falscher Tobias 2\_2 23Okt20:
Input Tabelle:
        1a) Knopf 4.OG
        1b) Knopf 3.OG
        1c) Knopf 2.OG
        1d) Knopf 1.OG
        1e) Knopf EG
        1f) Knopf UG
        1g) Tueroeffner
        1h) Tuerschliesser
        1i) Notfallglocke

        2a) 4.OG ruf/runter
        2b) 3.OG ruf/rauf
        2c) 3.OG ruf/runter
        2d) 2.OG ruf/rauf
        2e) 2.OG ruf/runter
        2f) 1.OG ruf/rauf
        2g) 1.OG ruf/runter
        2h) EG ruf/rauf
        2i) EG ruf/runter
        2j) UG ruf/rauf

        3a) Ueberlastungsensor

        4a) 4.OG Tuer offen?
        4b) 4.OG lift anwesend?
        4c) 3.OG Tuer offen?
        4d) 3.OG lift anwesend?
        4e) 2.OG Tuer offen?
        4f) 2.OG lift anwesend?
        4g) 1.OG Tuer offen?
        4h) 1.OG lift anwesend?
        4i) EG Tuer offen?
        4j) EG lift anwesend?
        4k) UG Tuer offen?
        4l) UG lift anwesend?

        5a) Ueberlastung

        6a) Timer

Output Tabelle:
        1a)winde rauf
        1b)winde runter

        2a) 4.OG Pfeil runter
        2b) 4.OG aktuelles Stockwerk
        2c) 3.OG Pfeil runter
        2d) 3.OG Pfeil rauf
        2e) 3.OG aktuelles Stockwerk
        2f) 2.OG Pfeil runter
        2g) 2.OG Pfeil rauf
        2h) 2.OG aktuelles Stockwerk
        2i) 1.OG Pfeil runter
        2j) 1.OG Pfeil rauf
        2k) 1.OG aktuelles Stockwerk
        2l) EG Pfeil runter
        2m) EG Pfeil rauf
        2n) EG aktuelles Stockwerk
        2o) UG Pfeil rauf
        2p) UG aktuelles Stockwerk

        3a) alarm

        4a) ueberlasthorn
```

```
        5a) not - aus

Zustaende:
        4.OG geparkt tuer offen
        4.OG geparkt tuer zu
        zwischen 4.OG und 3.OG aufwaerts
        zwischen 4.OG und 3.OG abwaerts
        3.OG geparkt tuer offen
        3.OG geparkt tuer zu
        zwischen 3.OG und 2.OG aufwaerts
        zwischen 3.OG und 2.OG abwaerts
        2.OG geparkt tuer offen
        2.OG geparkt tuer zu
        zwischen 2.OG und 1.OG aufwaerts
        zwischen 2.OG und 1.OG abwaerts
        1.OG geparkt tuer offen
        1.OG geparkt tuer zu
        zwischen 1.OG und   EG aufwaerts
        zwischen 1.OG und   EG abwaerts
          EG geparkt tuer offen
          EG geparkt tuer zu
        zwischen   EG und   UG aufwaerts
        zwischen   EG und   UG abwaerts
          UG geparkt tuer offen
          UG geparkt tuer zu

        gestoppt wegen ueberlastung
        gestoppt wegen not-aus
```

Anmerkungen

- o *'Motor-Überdrehzahl'*- Sensoren sind unüblich (aber nit falsch!)
  (eingeschaltete Asynchronmaschinen werden vom Drehfeld angetrieben und auch gebremst, ausgeschaltete winden haben eine mechanische Bremsklemme)
  (*'Überlast'*-Sensoren werden als *'Motor-Überstrom-Fühler')* realisiert)
- o ein 'Lift-zu-schnell'- Sensor ist unüblich (aber nit falsch!) und unnötig (s.o.)
  (man könnte von der Zeitdifferenz der Sensorsignale von 'vor-dem-Stock', und 'genau-im-Stock' auf das Tempo schliessen, was einen 'zu-schnell' - Sensor erübrigt)

7.7 Aufgaben Schaltwerk-Simulator

Listing 2: FSM Simulator Teilnehmerbeiträge

```
crewmate 1\_1
import time
Button = int(input())
if Button == 1: #Hier ruft man den Aufzug
    time.sleep(5) #Die ganzen time.sleep sollen des darstellen, das man ja immer warten muss  bis der Lift kimmt oder bis er die ^...Tren ^ffnet usw.
    print("^...Tre ^ffnet") #Hier steigt man ein
    while True:
        Stockwerk=int(input("Zu welchen Stockwerk wollen Sie:"))
        print("Aufzug geht nach",Stockwerk) #Hier sagt der "Aufzugbot" wo man hingeht
        time.sleep(5) #I wollt hier eigentlich no ^...einfgen das wenn die ^...Tr blockiert ist, er no ^/lnger wartet bis er die ^...Tr
        schliest, hab aber um ehrlich zu sein keine Ahnung wie i des machen soll
        print("^...Tre schliesen")
        time.sleep(5) #Hier ^knnt ma vielleicht no ^...einfgen, das er ^/lnger braucht wenn er vom z.b. 1-4 Stockwerk braucht
        print("Aufzug ist beim",Stockwerk,"Stockwerk angekommen")
        time.sleep(2)
        print("^...Tre ^ffnet")
        time.sleep(5)
        print("^...Tre schliest")
        break
#Des ist mein Programm. I denk man ^/htte es besser machen ^knnen aber wie oben ^/erwhnt bin i nit allzu gut im Programmieren und kenn mi daher nit so aus. Aber
    im
grossen und ganzen kommt die Idee ^...rber/*

/*****************************************************************************************************************************************************
 __  __     ______     ______     ______
/\ "-./  \   /\  __ \   /\  ___\   /\  ___\
\ \ \-./\ \  \ \  __ \  \ \ \__ \  \ \  __\
 \ \_\ \ \_\  \ \_\ \_\  \ \_____\  \ \_____\
  \/_/  \/_/   \/_/\/_/   \/_____/   \/_____/
 ______     __  __
/\  == \   /\ \_\ \
\ \  __<   \ \____ \
 \ \_____\  \/\_____\
  \/_____/   \/_____/
 ______     ______     __  __
/\  ___\   /\  ___\   /\ "-./  \
\ \___  \  \ \  __\   \ \ \-./\ \
 \/\_____\  \ \_____\  \ \_\ \ \_\
  \/_____/   \/_____/   \/_/  \/_/
 ______     __  __     ______     ______     ______     __  __
/\  ___\   /\ "-./  \   /\  __ \   /\  ___\   /\  ___\   /\ \_\ \
\ \___  \  \ \ \-./\ \  \ \  __ \  \ \ \____  \ \  __\   \ \____ \
 \/\_____\  \ \_\ \ \_\  \ \_\ \_\  \ \_____\  \ \_____\  \/\_____\
  \/_____/   \/_/  \/_/   \/_/\/_/   \/_____/   \/_____/   \/_____/
1\_6*/
#define play_sound() {}

typedef enum {
    Button_LevelUG,
    Button_LevelEG,
    Button_Level1,
    Button_Level2,
    Button_Level3,
    Button_Level4,
    Button_Door_Close,
    Button_Door_Open,
    Button_Emergency,
    Overweight,
    Timer,
    Call_UG = 0,
    Call_EG = 1,
    Call_1 = 2,
    Call_2 = 3,
    Call_3 = 4,
    Call_4 = 5,
    Level_Current_UG = 0,
    Level_Current_EG = 1,
    Level_Current_1 = 2,
    Level_Current_2 = 3,
    Level_Current_3 = 4,
```

```
    Level_Current_4 = 5
} input;

typedef enum {
    Go_Up,
    Go_Down,
    Arrow_Up,
    Arrow_Down,
    Seven_Segment,
    Speaker
} output;

typedef enum {
    DUG = 01,
    DEG = 11,
    UEG = 12,
    D1 = 21,
    U1 = 22,
    D2 = 31,
    U2 = 32,
    D3 = 41,
    U3 = 42,
    U4 = 52,
    STOPUGZ0 = 03,
    STOPUGZ1 = 04,
    STOPUGO = 05,
    STOPEGZ0 = 13,
    STOPEGZ1 = 14,
    STOPEGO = 15,
    STOP1Z0 = 23,
    STOP1Z1 = 24,
    STOP1O = 25,
    STOP2Z0 = 33,
    STOP2Z1 = 34,
    STOP2O = 35,
    STOP3Z0 = 43,
    STOP3Z1 = 44,
    STOP3O = 45,
    STOP4Z0 = 53,
    STOP4Z1 = 54,
    STOP4O = 55,
    STOP = 0
} state;

void button(int, int);
void call(int, int);
void arrive(int);
void overweight();
void stop();
void timer();
void close_door();
void open_door();

state CURRENT_STATE;


void button(int current_level, int destination_level) {
    call(current_level, destination_level);
}

void call(int current_level, int call_level) {
    int i = CURRENT_STATE % 10;

    if (current_level < call_level) {
        if (i == 3 || i == 4)
            CURRENT_STATE = call_level * 10 + 2;
    } else if (current_level > call_level) {
        if (i == 3 || i == 4)
            CURRENT_STATE = call_level * 10 + 1;
    } else
        if(i == 3 || i == 4)
            CURRENT_STATE += -i + 5;
}

void arrive(int current_level) {
    CURRENT_STATE = current_level * 10 + 3;
    play_sound();
}

void overweight() {
    stop();
    play_sound();
}

void stop() {
    CURRENT_STATE = STOP;
}

void timer() {
    int i = CURRENT_STATE % 10;

    if(i == 3)
        CURRENT_STATE += 1; // - 3 + 4
    if(i == 5)
        CURRENT_STATE -= 1; // - 5 + 4
}

void close_door() {
    int i = CURRENT_STATE % 10;

    if(i == 5)
        CURRENT_STATE -= 1; // - 5 + 4
}

void open_door() {
    int i = CURRENT_STATE % 10;

    if(i == 3 || i == 4)
        CURRENT_STATE += -i + 5;
}

*****************************************************************************************************************************************************

Ubuntuu 2\_5:
leider nix ...
```

8 PLC (SPS)

- Programmable Logic Control (PLC) ist *'speicherprogrammierbare Steuerung* (SPS) auf englisch.
- Diese Systeme durchlaufen eine *Anweisungsliste)* (AWL) in festem Rhythmus (Zykluszeit klassisch:20ms, modern:1ms) immer wieder. So propagiert Input im Lauf der Zeit durch die in AWL definierte logische Gatter-Vernetzung.
- Alternativ kann eine Logikschaltung / ein logisches Verhalten auch in
  - ***FUP*** (Funktionsplan: Schaltungszeichnung mit logischen Gatter-Symbolen)
    oder
  - ***KOP*** (Kontaktplan: Elektriker- Verdrahtungsplan mit Schaltern und Relaiskontakten)

  spezifiziert werden.
- AWL = Anweisungsliste: Programm-Code
  Bei einfachen Programmen, wird der Programmcode direkt in den Organisationsbaustein (zB. OB1) geschrieben. Bei groesseren Programmen steht der Programmcode in den Programmbausteinen PB1, PB2, ... Im Organisationsbaustein steht nur der Aufruf der einzelnen Programmbausteine. Der Programmcode besteht aus einer Folge von Steueranweisungen (U, UN, O, ON, S, ...)
  - U ... "und"
  - O ... "oder"
  - A ... Ausgabe
  - E ... Eingang
  - S ... setzen
  - R ... ruecksetzen
  - M ... Merker (Flag-Bit-Speicher)
  - = ... Zuweisung
  - BE... Baustein-Ende
    usw...

  Bsp.-Programm:

```
UN E 0.0
U(
 O E 0.1
 O A 0.0
)
= A 0.0
BE
```

Bsp.-Programm:

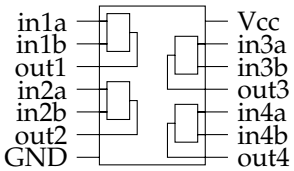
```
U E0.7
S M3.0
U M3.1
UN A0.7
O E0.6
R M3.0
U M3.0
UN M3.1
L KT500.1
SE T1
U T1
= M3.1
U M3.1
ZR Z2
U M3.0
L KZ576
S Z2
U E0.6
R Z2
U Z2
= A0.7
```

http://de.wikipedia.org/wiki/Kategorie:Speicherprogrammierbare_Steuerung
http://de.wikiversity.org/wiki/Kurs:Speicherprogrammierbare_Steuerung

9 integrierte Digitalschaltungen

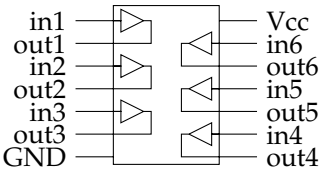
9.1 Gatter

7400 quad NAND und
CD 4011 quad NAND 4 Stk 2-fach-NAND



7402 NOR 4 Stk 2-fach-NOR
7486 quad EXOR 4 Stk 2-fach-XOR
CD 4001 dual NOR 4 Stk 2-fach-NOR

7414 hex inverter 6 Stk Inverter mit Schmitt-Trigger-Eingang (Hysterese)
CD 40106 hex schmitt trigger inverter wie 7414
CD 4049 hex inverter 6 Stk CMOS-Inverter
CD 4069 hex inverter 6 Stk CMOS-Inverter



9.2 Flipflop

7474 dual DFF 2 Stk D-Flipflop mit Set+Reset
CD 4013 dual DFF wie 7474
CD 4027 dual JKFF

9.3 Latches + Register

74373, 74573 8-Bit-Latch
74374, 74574 8-Bit-Register

9.4 Counter

CD 4020 - 14 stage binary counter
CD 4024 - 7 stage binary counter
CD 4040 - 12 stage binary counter
CD 4060 - 14-stage ripple-carry binary counter/divider and oscillator
CD 4029 - 4-Bit Presettable Binary/Decade Up/Down Counter w/parallel in+out

9.5 analog switch

CD 4016 - quad analog switch SPST
CD 4066 - quad analog switch SPST
CD 4051 1x8 analog switch
CD 4052 2x4 analog switch
CD 4053 3x2 analog switch
CD 4067 1x16 analog switch

9.6 div.

CD 4046 PLL
C744046 PLL
74HC9046 PLL
74HCT9046A PLL
CD 4008 adder 4-Bit binary adder
4007
4008
4017-HC4017
4020-24-40
4028-HEF4028B
4029
4063
4070
4076
4093
4095JKFF
4516
74107
74112
74123
74132
74138
7413
744060
74688
74HC109
74HC13
74HC191
74HC193
74hc245
74HC393
74HC93
74HCU04
74LS293

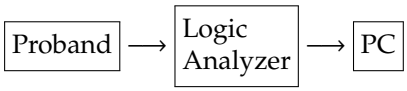
74LS380
74LS90
74LS93

DM74S00-Tk91.pdf
./Comm: MCP2515-CAN-Rl71.pdf
./Eth: enc28j60.pdf
./FTDI: DS-FT232BL.pdf DS-FT245BL.pdf
./I2C: PCF8583-1.pdf
./LCD: HD44780-1.pdf REVERSE-Qk3.txt
./Memory: 2764.pdf 28C16.pdf 4164DRAM-1.pdf 6116-HM6116.pdf 6264.pdf 93LC46-56-66.pdf AT28C16.pdf
./RS232: 16450UART.pdf 16550D-UART.pdf 6402.pdf 82050.pdf 8250A-UART.pdf Max232.pdf
./ShiftReg: 4014.pdf 4015.pdf 4021-cd4021.pdf 74164.pdf 74165.pdf 74166.pdf 74198-sn74198.pdf 74299-2.pdf 7491-ETC.pdf
./Switch: 4053b.pdf 4066.PDF 4067.PDF DG211a.pdf DG306.pdf DG308a.pdf MAX392-a.pdf MAX4541CPA-3.pdf MAX4544CPA.pdf MAX4614anasw.pdf
msp430g2231.pdf
6502
./uC/6502: 6502-RdR4-rockwell
./uP: AR9331IntroToCortex-M3.pdf msp430.PDF
./uP/Mips24k: 24K - LinuxMIPS.html
./uP/NextChip:
./uP/Z80: z80.pdf

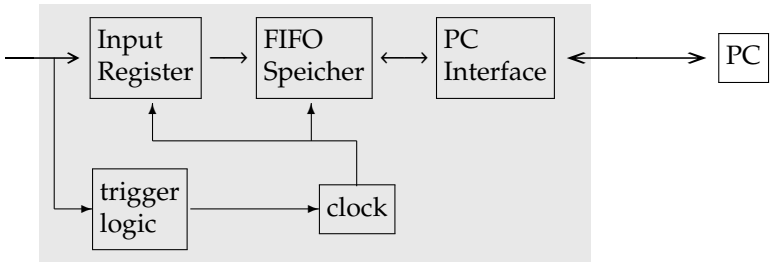
10 Logic Analyzer

- sind Geräte mit (möglichst vielen) parallelen Digitaleingängen (üblicherweise 8, 16, 32),
- deren Logikzustände **gleichzeitig** mit hoher Frequenz in FIFO-Speicher geschrieben (quasi *'abgetastet'*) werden.
- Für das Einschreiben ist komplexe, konfigurierbare Trigger-Logik vorhanden (vgl. *'breakpoints'*
- die FIFO-Inhalte sind von PC auslesbar (zur Analyse und Archivierung)
- Bedienung und Konfiguration erfolgen meist auch vom PC
- Mit dem Logic-Analyzer beobachtet man Digitalsignale von Bussen und Steuerleitungen
- durch die Gleichzeitigkeit der parallelen Signalerfassung sind → *Glitches* auffindbar
- High/Low-Spannung, Signal-Anstiegs/Abfall-Verlauf oder *Spikes* sind *nicht* erfassbar, da der Logic-Analyzer nur '0' und '1' Zustände aufzeichnet (er kann also kein Oszilloskop ersetzen).

Anordnung:



Logic Analyzer:



11 BoundaryScan: JTAG

11.1 JTAG/ BOUNDARY SCAN

JTAG/ Boundary Scan ist das wohl genialste elektrische Testverfahren. Es ist der Sprung vom physikalischen Zugriff auf die Leiterbahnen einer Baugruppe (wie es beim In-Circuit Testverfahren notwendig ist) mit all seinen physischen Grenzen hin zum elektrischen und somit grenzenlosen Zugriff. Dabei kommt das Testverfahren mit gerade einmal vier Steuerleitungen und einer Handvoll wichtiger „Design For Testability" Regeln aus! Spricht man heutzutage von JTAG oder Boundary Scan, bezieht man sich fast immer auf den Standard IEEE1149.1, also den statischen, digitalen Verbindungstest. Dessen Grenzen liegen im analogen Bereich wie auch im digitalen High-Speed Bereich. Doch auch hier erfolgen Weiterentwicklungen mit den beiden Standards IEEE1149.4 und IEEE1149.6.
Ein Boundary Scan Testentwickler muss sich heutzutage nicht mehr zwingend mit jedem einzelnen Detail des Verfahrens beschäftigen, da moderne, auf Bauteilmodellen basierende Tools den Großteil der Arbeiten für ihn übernehmen.
JTAG/ Boundary Scan basiert auf einem weltweit anerkannten Standard des „Institute of Electrical and Electronics Engineers (IEEE)" aus dem Jahre 1990.

Entstehungsgeschichte oder Das Testen einer integrierten Schaltung

Seit es integrierte Schaltungen gibt besteht die Notwendigkeit die Funktion zu überprüfen. Handelt es sich dabei um digitale Schaltungen, gestaltet sich deren Test relativ einfach: Man legt an den Eingängen nacheinander sämtliche möglichen Testvektoren an und vergleicht die Reaktion der Schaltung an den Ausgängen (IST-Ausgangsvektor) mit den erwarteten Mustern (SOLL-Ausgangsvektor). Ergibt der Vergleich keine Abweichung, ist die Schaltung in Ordnung.
Für ein einfaches UND-Gatter mit zwei Eingängen ist die Anzahl der Testvektoren noch überschaubar. Nach Moore und McCluskey errechnet sich diese nach der Formel:

$$Q^{(x+y)}$$

Q = Mindestanzahl der Testvektoren
x = Zahl der Eingänge
y = Zahl der Speicherelemente (bei sequentiellen Schaltungsteilen)

Da ein UND-Gatter gewöhnlich keinerlei Speicherelemente beinhaltet, kommt man auf eine Anzahl von gerade einmal vier notwendigen Testvektoren, was eine leicht zu handhabende Menge darstellt.
Führt man diese Berechnung allerdings für eine Schaltung mit einer angenommenen Anzahl von 25 Eingängen und 50 Speicherelementen durch, wird man sofort erkennen, mit welchem Problem sich die Ingenieure in den Chip Entwicklungen in den 70er Jahren konfrontiert sahen.
1973 hatte Ed Eichelberger bei IBM mit der Erfindung des ersten „Level Sensitive Scan Design (LSSD)" Verfahrens eine bahnbrechende Idee. Hierzu werden die in einem Chip vorhandenen Speicherelemente in Ihrer Funktion erweitert. Sie bekommen vier zusätzliche Anschlüsse: einen Eingang (IN), einen Ausgang (OUT) und zwei Clocks (A und B); Mit diesen zusätzlichen Ressourcen ist es möglich ebenfalls auf die Ein- und Ausgänge der Speicherelemente zuzugreifen. Das im U.S. Patent 3,761,695 beschriebene Verfahren eröffnete also einen zweiten Weg, neben der eigentlichen Funktion der Speicherelemente auf deren Ein- und Ausgänge zuzugreifen. Das war der Durchbruch!
Aufgrund dieser neu geschaffenen Möglichkeit zerfallen die zuvor noch komplexen und untestbaren Chip Designs, bestehend aus einer vielfachen Verschaltung von kombinatorischer mit sequentieller Logik, in Gruppen, welche dann nur noch aus kombinatorischen und somit einfach zu testenden Elementen bestehen.
Erst Anfang der 80er Jahre begann man auch auf Baugruppen-Ebene, die Problematik der „zunehmenden Komplexität der Baugruppen mit immer höheren Packungsdichten" anzugehen. Als eine der ersten beschäftigte sich die 1985 gegründete „Joint European Test Action Group" mit dem Thema. Damals bestand diese Gruppe aus Test-Ingenieuren der großen europäischen Chip Hersteller. 1986 traten ihr weitere Firmen aus Nordamerika bei, und die Gruppe wandelte ihre Bezeichnung in „Joint Test Action Group (JTAG)" um. Die JTAG konstruierte dann ein Verfahren, welches sich stark an das von Ed Eichelberger entwickelte LSSD Verfahren anlehnte. So definiert es ebenfalls Speicherelemente innerhalb eines Chips, welche miteinander in einer Schiebekette verbunden sind. Nur mit dem Unterschied, dass diese Speicherelemente nun zusätzlich an der Peripherie, sprich an der Bauteilgrenze („at boundaries") eingebracht wurden.
Das entwickelte Verfahren wurde deshalb auch Boundary Scan genannt. Standardisiert wurde es im Jahre 1990 vom „Institute of Electrical and Electronics Engineers (IEEE)" als 1149.1 „Standard Test Access Port and Boundary Scan Architecture".
Da die Arbeit der JTAG maßgeblich für den Inhalt des Standards war, hat sich als Synonym für den Standard der Name der Gruppe etabliert.
Der Standard IEEE1149.1 hat unter anderem in den Jahren 1993 und 1994 eine Überarbeitung erfahren und liegt nun im aktuellen Stand von 2001 vor.

11.2 Der Boundary Scan Standard IEEE1149.1

Der Boundary Scan Standard IEEE1149.1 beschreibt den statischen, digitalen Verbindungstest. Spricht man heute von Boundary Scan oder JTAG, so bezieht sich dies immer auf den Standard IEEE1149.1.
Im Standard selbst ist der Aufbau eines Boundary Scan-fähigen Bausteins dargelegt, wie auch die Beschreibungssprache, die „Boundary Scan Description Language (BSDL)", welche die für jeden Baustein einzigartige Boundary Scan Ressourcen offen legt. Für ein besseres Verständnis des Boundary Scan Testverfahrens ist es zunächst notwendig, den inneren Aufbau eines solchen Bausteins zu kennen. Der Standard IEEE1149.1 definiert hierzu vier wesentliche Bestandteile, über die ein Boundary Scan-fähiger Baustein verfügen muss:

- *Test Access Port (TAP)*
- *TAP Controller*
- *Befehlsregister („Instrucion register")*
- *ein oder mehrere Datenregister*

11.3 Test Access Port (TAP)

Das „Test Access Port" stellt die Schnittstelle zwischen der im Baustein befindlichen Boundary Scan Logik und der Außenwelt dar. Es sind drei Eingänge (zzgl. eines optionalen vierten) und ein Ausgang beschrieben. Die Eingänge sind:
- ***TDI** Test Data Input*
- ***TMS** Test Mode Select*
- ***TCK** Test Clock*
- ***/TRST** Test Reset - optional*

Der Ausgang ist:
- ***TDO** Test Data Output*

Bei den beiden Signalen TCK und TMS sowie beim optionalen /TRST Signal handelt es sich um Broadcast Signale, wohingegen das TDI hin zum TDO eine serielle Kette bildet, die sogenannte Scankette oder auch Scanpfad. Auf Baugruppen-Ebene spricht man vom Testbus.
Das Geniale bei diesem Konstrukt ist, dass nie mehr als vier (optional fünf) Signalleitungen benötigt werden, unabhängig davon, wie viele Bausteine in die Scankette geschaltet werden. Im Boundary Scan Baustein sind der „Test Clock", das „Test Mode Select" wie auch der „Test Reset" direkt mit dem „TAP Controller", sprich statisch, verbunden. Die Signale sind einzig und allein verantwortlich für den dessen Zustand. Das bedeutet gleichzeitig auch, dass sich alle Boundary Scan Bausteine einer Scankette immer im gleichen TAP Zustand befinden. Das bedeutet nicht automatisch, dass sich auch alle Bausteine immer im gleichen Betriebsmode/ Befehl befinden müssen. Dieser wird nämlich nicht über den TAP Zustand definiert, sondern über das Datenregister, das für jeden Baustein separat gesetzt wird.
Im Gegensatz zur statischen Zuordnung der zuvor beschriebenen Signale ist die Auswahl des Registers, das zwischen dem „Test Data Input" und dem „Test Data Output" geschaltet wird, flexibel. Dabei wird zunächst im „TAP Controller" entschieden, ob das Befehlsregister oder eines der Datenregister aktiviert wird.
Es wird entsprechend der Graph „Scan IR" oder „Scan DR" beschritten. Im Falle eines Datenregisters basiert die weitere Auswahl auf dem aktuell im Befehlsregister befindlichen Betriebsmode.

11.4 TAP Controller

Der „TAP Controller" ist verantwortlich für die komplette Steuerung der Boundary Scan Logik im Baustein. Das heißt, er ist unter anderem dafür verantwortlich, ob eine Boundary Scan Zelle aktiviert bzw. deaktiviert wird und ob diese messen oder treiben soll.
Herzstück des „TAP Controllers" ist die „TAP state machine". Die darin enthaltenen Zustände haben einen unterschiedlichen Einfluss auf die Steuerung der internen Boundary Scan Logik. Am besten lässt sich dies im Zusammenspiel mit der Boundary Scan Zelle erklären.
Der Wechsel von einem Zustand des „TAP Controllers" in einen anderen erfolgt über die steigende Flanke am „Test Clock". Ein Zustand kann immer auf zwei Wegen verlassen werden. Entscheidend hierfür ist, welchen Pegel das „Test Mode Select" Signal zu diesem Zeitpunkt hat. Als Beispiel kann der „Testlogic Reset (TL Reset)" Zustand nur verlassen werden, wenn das TMS Signal bei der steigenden TCK Flanke gleich 0, also Low ist. Dann wäre der nächste Zustand „Run Test / Idle (RT / Idle)".

11.5 Das Befehlsregister

Das Befehlsregister („Instruction register") entscheidet über den Betriebsmode des Boundary Scan Bausteins, der wiederum Einfluss auf die Steuerung der Boundary Scan Zellen wie auch auf die Auswahl des aktuell in die Scankette (Register zwischen TDI und TDO) geschalteten Datenregisters hat. Der Standard IEEE1149.1 definiert drei zwingend erforderliche Befehle:

- *BYPASS*
- *SAMPLE/PRELOAD*
- *EXTEST*

Zusätzliche Befehle sind erlaubt. Als Beispiel seien an dieser Stelle die häufig anzufindenden Befehle IDCODE und HIGHZ zu nennen.
Jedem Befehl ist ein entsprechender Befehlscode (Bitcode) hinterlegt. Dieser kann von jedem Chip Hersteller frei definiert werden (ausgenommen ist der BYPASS Befehl – er muss sich vollständig aus Einsen zusammensetzen). Auch die Länge des Befehlsregisters ist beliebig wählbar.

11.6 Die Datenregister

In einem Boundary Scan-fähigen Baustein können mehrere Datenregister enthalten sein. Sie dienen dazu, Informationen im Baustein abzulegen oder auch daraus auszulesen. Der Standard IEEE1149.1 beschreibt im Minimum zwei zwingend notwendige Datenregister:
- *bypass*
- *boundary-scan*

Auch hier sind wieder zusätzliche Register möglich, wie das „device identification“ oder umgangssprachlich auch „idcode“ Register.
Das „bypass“ Register stellt dabei die Möglichkeit dar, den Baustein aus einem Verbund von Boundary Scan Bausteinen zu lösen, sprich diesen zu umfahren (“bypass“). Das Register hat dazu eine minimale Länge von nur einem Bit. Der Wert des Bits ist unveränderlich und mit 0 definiert.
Das für ein späteres Testen wesentlich interessantere Datenregister ist jedoch das „boundary-scan“ Register, soweit es die Aneinanderreihung der einzelnen Boundary Scan Zellen darstellt. Da jeder Chip eine andere Anzahl an Boundary Scan Zellen besitzt, ist die Länge dieses Registers variabel.

11.7 Die Boundary Scan Zelle

Die Boundary Scan Zelle ist der Hauptbestandteil des Boundary Scan Testverfahrens. Alle bisher beschriebenen Konstrukte dienen ausnahmslos der korrekten Ansteuerung der einzelnen Boundary Scan Zellen.
Die Boundary Scan Zelle ist die geniale Möglichkeit, den Bauteilpin eines Bausteins gelöst von dessen normaler Funktion zu kontrollieren, d.h. einen bestimmten Pegel zu treiben oder auch zu messen. Zu diesem Zweck befindet sich die Boundary Scan Zelle zwischen der Kernlogik des Bausteins und dessen Peripherie (Ausgangstreiber, Eingangstreiber); Aufgrund der funktionellen Ähnlichkeit zu den physischen Antastnadeln („nails“) des In-Circuit Testverfahren, welche dort den Zugriff auf die einzelnen Testpunkte realisieren, spricht man bei den Boundary Scan Zellen auch von „electronic nails“.

Eine Boundary Scan Zelle kann immer nur EINE der folgenden Funktionen übernehmen:
- *Aktivieren/ deaktivieren des Treibers → Steuerzelle („control“)*
- *Vorgabe des zu treibenden Pegels → Ausgangszelle („output“)*
- *Messen des Pegels → Eingangszelle („input“)*

Damit trotzdem an einem Bauteilpin getrieben und auch gemessen werden kann, befinden sich in aller Regel mehrere (bis zu drei) Boundary Scan Zellen an einem einzigen Bauteilpin. Aus diesem Grund ist es auch nicht verwunderlich, dass es oft mehr Boundary Scan Zellen als Bauteilpins gibt.
Der interne Aufbau einer einzelnen Boundary Scan Zelle kann sich sehr unterschiedlich gestalten. Der Standard IEEE1149.1 beschreibt in der Fassung 2001 allein zehn unterschiedliche Zell Typen (BC1 bis BC10). Eigene Strukturen sind zusätzlich möglich. Der Aufbau ist oft sehr ähnlich.

11.8 Die Boundary Scan Description Language (BSDL)

Jeder Boundary Scan-fähige Baustein hat seine ganz spezielle Boundary Scan Struktur, ohne deren Kenntnis ein Testingenieur bzw. eine Testsoftware nicht in der Lage wäre, sinnvoll mit dem Baustein zu arbeiten. Der Standard IEEE1149.1 schreibt zwar einiges zwingend vor, lässt aber auch genügend Freiräume für Individualität. Die ist auch notwendig, was gerade am Beispiel der Struktur/Anzahl der Boundary Scan Zellen deutlich wird: Ein Baustein mit 20 Anschlusspins verfügt sinnvollerweise über eine geringere Anzahl an Zellen verglichen mit einem Baustein mit 1.500 Pins.
Um diese Individualität zu beschreiben, wurde die „Boundary Scan Description Language (BSDL)“ entwickelt. Sie ist das Verständigungsmedium zwischen Chip-Hersteller (der allein das „Innenleben“ seines Chips kennen kann) und dem Testingenieur (der dieses „Innenleben“ in seinem speziellen Einsatzfall verwenden möchte). Es handelt sich dabei um eine Datei.
Man findet Angaben über:
- *verfügbare Testbussignale (insbesondere Angaben über das Vorhandensein des optionalen /TRST Signals und auch zur maximalen TCK Frequenz, bis zu welcher der Baustein betrieben werden kann)*
- *mögliche „Compliance“ Pins*
- *das Befehlsregister (verfügbare Befehle inkl. deren Bitcode; Länge des Befehlsregisters)*

- *die Datenregister (verfügbare Datenregister inkl. möglicher voreingestellter Werte, z.B. IDCODE des Bausteins)*
- *die Struktur der Boundary Scan Zellen (Anzahl, Typ, Funktion, Zuordnung zum Bauteilpin)*

11.9 Der Standard IEEE1149.4

Der Erfolg des JTAG/ Boundary Scan Standards IEEE1149.1 hat die Beteiligten beflügelt und zu vielen Ideen angeregt, das Testverfahren noch besser, noch grenzenloser zu machen. Aus der Vielzahl der neu entwickelten und zum Teil schon verabschiedeten Standards werden zwei kurz vorgestellt.
Der Durchbruch des Standards IEEE1149.4 würde vermutlich das Aus für das klassische In- Circuit Testverfahren bedeuten, denn hierbei handelt es sich um den „Mixed-Signal" oder auch analogen Verbindungstest.
Das Verfahren ist auch hier sehr einfach gehalten. Zusätzlich zu den vier (oder optional fünf) TAP Signalen werden zwei „Analog Test Access Port (ATAP)" Signale „Analog Test 1 (AT1)" und „Analog Test 2 (AT2)" benötigt. Diese beiden zusätzlichen Pins können unabhängig voneinander per Testbus-Befehl intern auf einen beliebigen Bauteilpin des IEEE1149.4 fähigen Baustein geschaltet werden.
Man könnte auch sagen, dass ein IEEE1149.4 fähiger Baustein über eine interne Relaismatrix verfügt, die per Testbus auf beliebige Pins geschaltet werden kann. Wird nun an das ATAP eine entsprechende externe Messtechnik angeschlossen, hat man vom Prinzip her einen klassischen In-Circuit Tester (mit eingeschränkten Funktionen).

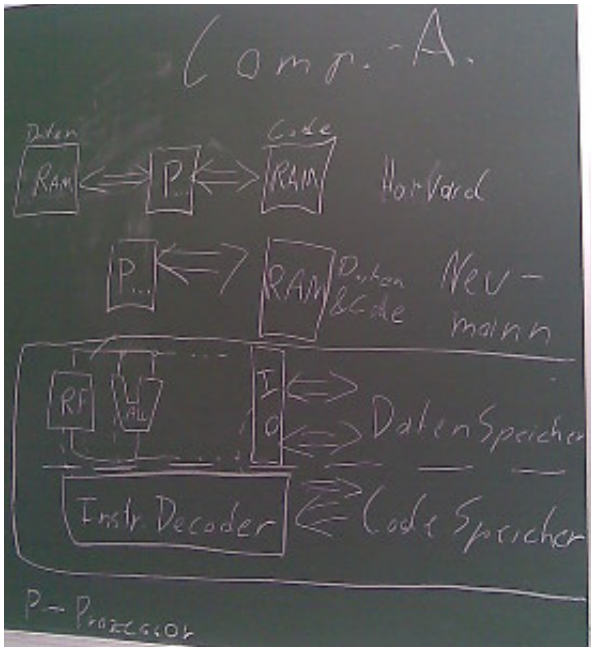
11.10 Der Standard IEEE1149.6

Der Standard IEEE1149.6 ermöglicht das Testen serieller, digitaler High-Speed Verbindungen. So beschreibt er den „Advanced Digital Network" Verbindungstest. Ein großer Vorteil des Standards liegt darin, dass er mit den vorhandenen TAP Signalen auskommt. Er begnügt sich mit wenigen neuen Befehlen, einer etwas erweiterten Boundary Scan Zelle und einem integrierten Testmuster Generator.
Das Prinzip ist auch hier wieder äußerst simpel. Einige Pins des IEEE1149.6 fähigen Bausteins sind mit einem neuen Typ Boundary Scan Zelle verbunden. Diese verfügt im Gegensatz zu den „alten" Zelltypen über einen speziellen Eingang, der mit dem internen Testmuster-Generator verbunden ist. Per Befehl schaltet die Boundary Scan Zelle auf den neuen Eingang um, und der Testmuster-Generator sendet unabhängig vom „Test Clock" Signal das Testmuster an die Boundary Scan Zelle und somit an den Bauteilpin.
Dies war zunächst die Sendeseite. Auf der Empfängerseite wird im gleichen Augenblick begonnen, die Testmuster vom Bauteilpin einzulesen und in einen Buffer zu schreiben. Anschließend erfolgt ein Vergleich mit dem vom Sender geschickten Testmuster und eine Gut/ Schlecht Aussage. Diese wird in einer Boundary Scan Zelle als 0 oder 1 abgelegt und kann vom Testsystem ausgelesen und ausgewertet werden.

(aus de.wikipedia.org, 'Boundary Scan', 'einfrg_bscan.pdf' 07.Dec'12)

Teil III
4-Prozessoren

12 embedded Processor Architectures



Harvard Architektur:
* *instruction word'* unabhängig von Datenbusbreite,
* schneller — kein Datenbus-Engpass

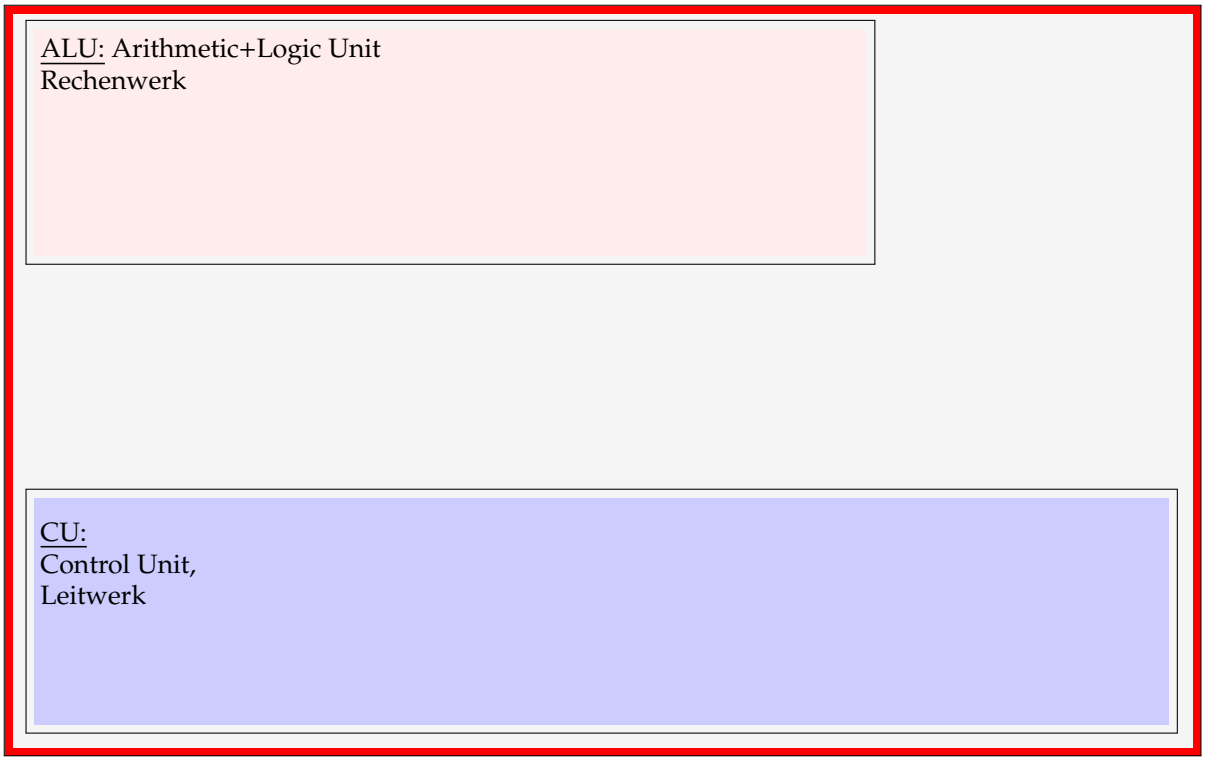
von-Neumann Architektur:
* Weniger externe Busleitungen

(Referat dB 14Sep16)

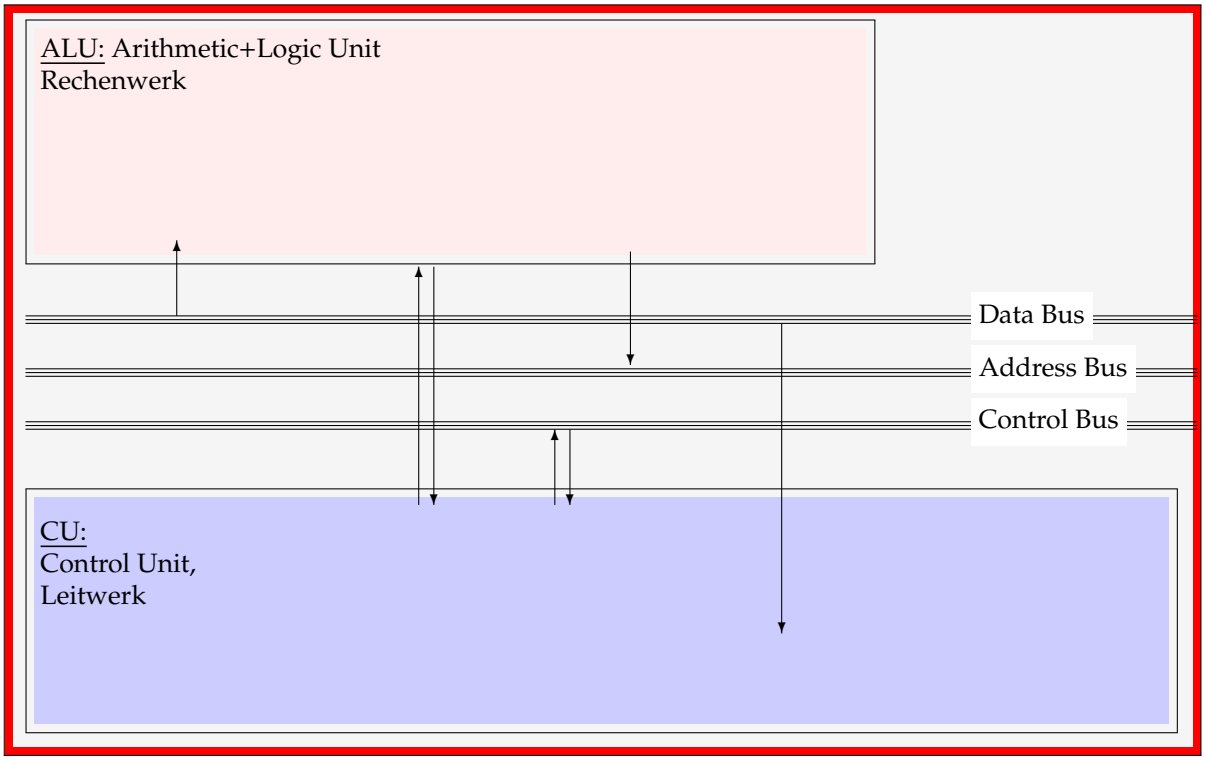
12.1 Du kannsch Prozessoren entwerfen!

12.1.1 Prozessor Blockdiagramm

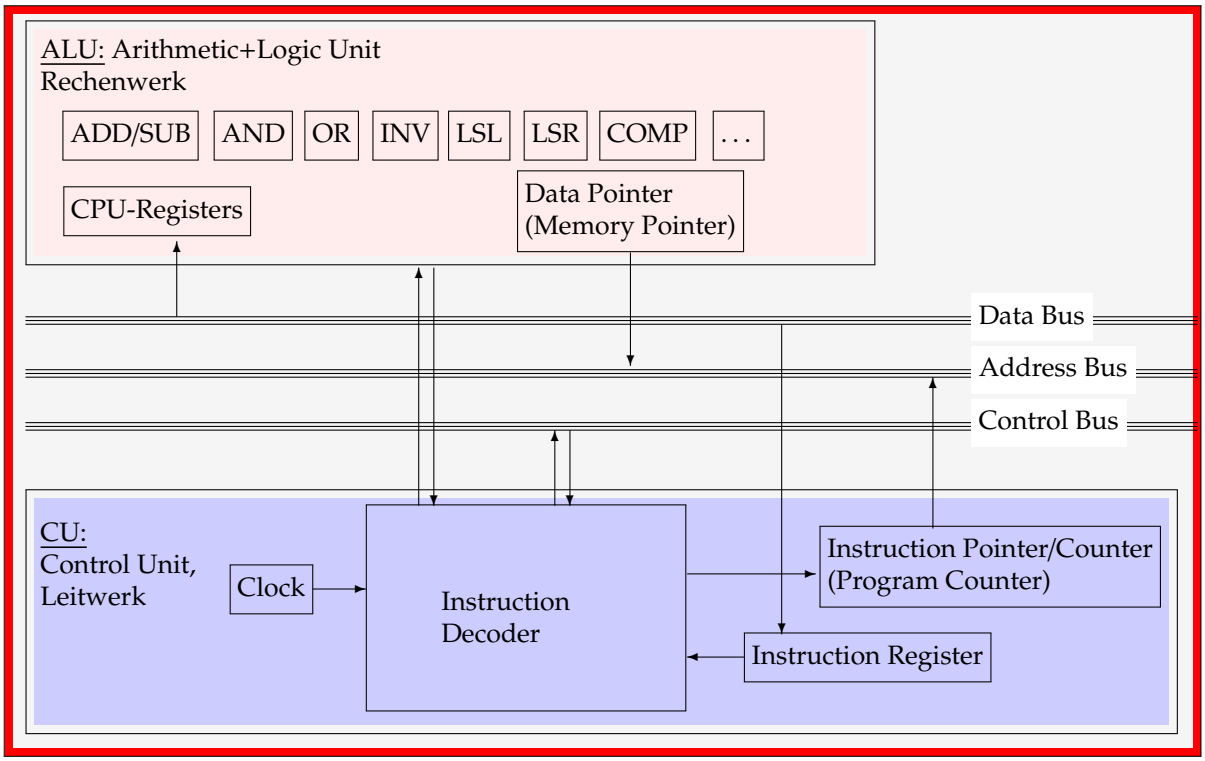
Hauptbestandteile: Rechenwerk (ALU), Leitwerk (CU)



ALU und CU sind untereinander und nach "draussen" verbunden



CU steuert, ALU arbeitet



12.1.3 Digital-Computer-Prozessor in VHDL

Aufgabe Teil1

Achtung: Folgende Aufgabe ist wesentlich **simpler** als es klingt.
(lies weiter unten)

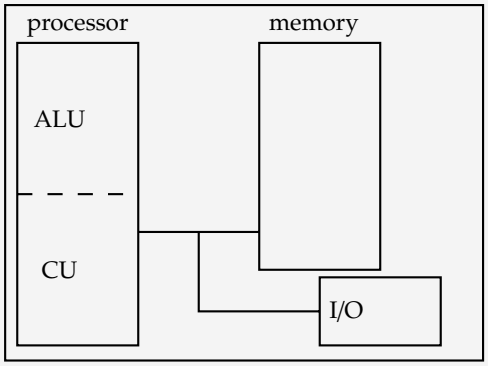
1. schreib eine DigitalProzessor-**ALU** incl. Instruction-Decoder in VHDL-Pseudocode
(mit den wichtigsten arithmetischen und logischen Operationen)
Der Prozessor habe einen 8-Bit-DatenBus und einen 16-Bit-AdressBus
2. schreib für eine DigitalProzessor-**CU** (auch in VHDL-Pseudocode):
   a den Instruction-Pointer
   b das Instruction-Register
   c den von-Neumann-Phasen-Counter

Aufgabe Teil2

im Teil-1 wurden Bestandteile (=Struktur) beschrieben; was diese Teile tun, was mit ihnen geschieht das *'Verhalten' (behaviour)*, fehlt aber noch.
Aus dem Lateinischen, der Sprache der Römer, kennt man das Wort *'Prozess'*, von *'pro-cedere'* - 'pro' ≡ nach vorn, voran und 'cedere' ≡ gehen, schreiten - also *'voranschreiten'*, substantivisch gebraucht ***'Vorgang'***.

3. Ein Vorgehen, Schritt für Schritt nacheinander geschaltet, nennt man auch eine *'Sequenz'*, weshalb man es in eine *'sequentielle Umgebung'* schreibt, in einen ***'process'***.
4. schreib (in VHDL Pseudocode) alles,

– was die **CU** tun muss und
– unter welchen Voraussetzungen (if . . .) sie es tun muss
in einen `'process CU(clk)'`



Com-
pu-
ter

ALU ... Arithmetic + Logic Unit ('Rechenwerk')
CU ..... Control Unit ('Leitwerk')
uP ..... micro processor
uC ..... micro controller
memory ... 'Hauptspeicher', (RAM)
I/O ..... Input+Output Devices

12.1.4 Prozessor nach John vonNeumann

Für einen Digital-Computer-Prozessor braucht man **ALU** und **CU**

Arithmetics and **L**ogics **U**nit - ein 'Rechenwerk' == *'der Taschenrechner'*

ein **'InstructionCode** entscheidet die Rechen-Operation:

```
case (InstructionCode) is
   when( Instruction1 ) => Y<=A+B;
   when( Instruction2 ) => Y<=A-B;
   when( Instruction3 ) => Y<=A+1;
   when( Instruction4 ) => Y<=A-1;
   when( Instruction5 ) => Y<=A and B;
   when( Instruction6 ) => Y<=A or B;
   . . . usw.
```

Diese 'case'-Fallunterscheidung ist ja schon der InstructionDecoder, der zur CU gehört

Control **U**nit - eine automatische Steuerung 'Leitwerk'

Die **ControlUnit** ist wesentlich anspruchsvoller:

1 Instruction **FETCH**: Nächsten InstructionCode aus dem Speicher holen
2 Instruction **DECODE**: Wartezeit für den Durchlauf im InstructionDecoder
3 Data **LOAD**: Daten aus dem Speicher holen
4 **EXECUTE**: Wartezeit für den Durchlauf in der ALU
5 Data **SAVE**: Ergebnis in den Speicher schreiben

} 'von-Neumann'-Zyklus

FETCH Der InstructionPointer (auch 'ProgramCounter') wird auf den AdressBus geschaltet und das -READ Signal auf '0' gezogen. Der Speicher liefert (nach Wartezeit) den Inhalt der angegebenen Adresse, den 'InstructionCode' (auch 'OpCode'), auf dem DatenBus, den wir im InstructionRegister zwischenspeichern. Ohne dieses Register ginge der InstructionCode verloren, sobald der DatenBus anderweitig benutzt wird, und der InstructionDecoder spielte verrrückt.

Bestandteile: **AdressBus, DatenBus, InstructionPointer, -READsignal, InstructionRegister**

DECODE es ist einfach nur zu warten. . .

LOAD Die Datenadresse vom DataPointer wird auf den AdressBus gelegt und das -READ Signal auf '0' gezogen. Der Speicher liefert nun (nach Wartezeit) den Inhalt der angegebenen Adresse auf dem Datenbus, den wir in einem ALU-Register, zB. dem Register'A' ablegen

Bestandteile: **DataPointer, Register'A'**

EXECUTE es ist einfach nur zu warten. . .

SAVE Der DataPointer wird auf den AdressBus geschaltet und das Ergebnisregister auf den Datenbus. Sodann wird das -WRITE Signal auf '0' gezogen (und gewartet).

Bestandteile: **Ergebnisregister, -WRITEsignal**

Da in jeder vonNeumann-Phase anders zu arbeiten ist, muss die Schaltung diese Phase irgendwoher wissen, wozu wir einen eigenen Zähler einbauen.

Bestandteil: **vonNeumannPhaseCounter**

von-Neumann-Rechner haben keine Interrupts, Cache-Speicher, Pipeline, DMA, Branch-Prediction, out-of-order-Execution odgl. Firlefanz

12.1.5 in VHDL

wie schreibt man

→ **'warten':**

vorerst gar nicht

(wir werden Adressen bei fallender clock-Flanke ausgeben und Daten bei steigender Flanke einlesen)

→ **externe Leitungen und Busse:**

```
entity Processor is
    port (
      READ, WRITE : out std_logic;
      clock : in std_logic;
      AddressBus : out std_logic_vector(15 downto 0);
      DataBus : inout std_logic_vector(7 downto 0);
    );
end entity Processor;
```

→ **interne Register und Counter:**

```
architecture ALUundCU of Processor is
  signal
       InstructionPointer,
       DataPointer: std_logic_vector(15 downto 0);
  signal
       InstructionRegister,
       ErgebnisRegister,
       RegisterA: std_logic_vector( 7 downto 0);
  signal
       PhaseCounter: std_logic_vector( 2 downto 0);
begin
  ...
end ALUundCU;
```

```
   /*Digital-Prozessor-Control-Unit:
     ------------------------------ */
ENTITY Prozessor IS
  PORT (
  DBus     : inout  STD_LOGIC_VECTOR(7 downto 0);
  ABus     : out    STD_LOGIC_VECTOR(15 downto 0);
  clk      : in     STD_LOGIC;
  notWrite : out    STD_LOGIC;
  notRead  : out    STD_LOGIC;
  );
END Prozessor;
ARCHITECTURE Innenleben of Prozessor IS
  InstPTR  :signal STD_LOGIC_VECTOR (15 downto 0); --InstructionPointer
  DPointer :signal STD_LOGIC_VECTOR (15 downto 0); --DataPointer
  InstReg  :signal STD_LOGIC_VECTOR ( 7 downto 0); --InstructionRegister
  PhaseCTR :signal STD_LOGIC_VECTOR ( 2 downto 0);  --vonNeumannPhasen1..5
  CpuRegA,
  CpuRegB  :signal STD_LOGIC_VECTOR ( 7 downto 0); --CpuRegisterA+B
  ErgReg   :signal STD_LOGIC_VECTOR ( 7 downto 0); --ErgebnisRegister
  BEGIN
  CU: process(clk) begin
    PhaseCTR <= PhaseCTR+1;
    if (PhaseCTR=1) then --FETCH
      if rising_edge(clk) then
        InstPTR <= InstPTR+1;
        ABus <=  InstPTR;
        notRead <= '0';
      else
        InstReg <= DBus;
        notRead <= '1';
        ABus <= (others => 'z');
      end if;                    --FETCHphase
    elseif (PhaseCTR=2) then     --DECODE
        --(tue gar nichts)
    elseif (PhaseCTR=3) then     --LOAD
      if rising_edge(clk) then
        ABus <= DPointer;
        notRead <= '0';
      else
        CpuRegA <= DBUS
        notRead <= '1';
        ABus <= (others => 'z');
      end if;
    elseif (PhaseCTR=4) then     --EXECUTE
        --(tue gar nichts)
    elseif (PhaseCTR=5) then     --SAVE
      if rising_edge(clk) then
        ABus <= DPointer;
        DBUS <= ErgReg;
        notWrite <= '0';
      else
        notWrite <= '1';
        ABus <= (others => 'z');
        DBus <= (others => 'z');
      end if; --rising_edge(clk)
    end if;  --PhaseCTR
  end process CU;
END Innenleben;
```

funzt wie a Eisenbahnspiel: von-Neumann Zyklus:

Des isch wie *Eisenbahnspielen:*
(Du schiabsch die Bytes aufde Busse ummanonder; ALU, CU und RAM sein je a Bahnhof)
Du hash nia Eisenbahn gspielt?

Shit!

von-Neumann step 1: Instruction FETCH

Die CU schaltet den Program Counter aufn Adressbus (S40, S42), wartet und schaltet den Databus ins Instruction Register durch (S51, S52)

Die next *Instruction (Maschinen-Befehl)* isa Bitkombination, de 'irgendwo' in Memory steaht. A Maschin kann mit 'irgendwo' nix unfongen, des miassma scho digital implementieren – als a Register (a 'Latch') namens *Program Pointer*; und damitse de Instructions der Reih nach ausfiahrt, machma den Pointer als upwards-Counter, dann heisst er **Program Counter** (s. CU unten rechz)

↓ erhöhen des Programmzählers (program counter, instruction pointer)
↓ Programmzählerinhalt auf den Adressbus legen
↓ READ-Signalleitung aktivieren
↓ Wartezeit für Speicherbausteine
↓ neuen Befehlscode (instruction code) vom Datenbus ins Instructionregister einlesen
↓ READ-Signalleitung de-aktivieren
↓ Adressbus freigeben ('Z'-state)

↓

von-Neumann step 2: Instruction DECODE

Mir lassen der Instruction-Decoder-Kombinatorik und den Switches in der ALU einfoch nur a Bissl a Zeit – paar (-zig) Nanosekunden – Summe der Laufzeiten vom Instruction Register bis incl. ALU Switches (S00 - S05)

↓

von-Neumann step 3: Memory Data LOAD

Mir schalten den **Memory Pointer** auf Adressbus (S14, S16), warten und schalten den Databus (S99) ins ALU Register A (Akkumulator) (S30) (oder je nach Befehl und Architektur Register B (S31), R0, R1, ⋯ , R31 odgl.)

Natyrlich muasch vorher die Speicher-Adresse der gewünschten **Variable** in den Memory Pointer einigladen haben (S15, S17) - auf insern Prozessor a extra Instruction vorher ausfiahren - bessere Maschinen kennen des im Befehlswort mitcodieren plus tun.

↓ Data-Pointer (memory pointer) - Inhalt auf den Adressbus legen
↓ READ-Signalleitung aktivieren
↓ Wartezeit für Speicherbausteine
↓ Binärwert vom Datenbus in ein Datenregister (A, B, R0, . . .) einlesen
↓ READ-Signalleitung de-aktivieren
↓ Adressbus freigeben ('Z' state)

↓

von-Neumann step 4: Instruction EXECUTE

Mir *warten* . . . und zwar bis die ALU Ergebnisse stimmen = Summe der Gatter-Laufzeiten von ALU-Registern (zB Akku A) durch die Verarbeitungsschaltungen (zB Adder) bis incl. der Ergebnis-Speicher-Latches.

↓

von-Neumann step 5: Memory Data SAVE

Mir schalten den Memory Pointer (data pointer) (S14, S16) aufn Adressbus und des entsprechende Ergenbis-Latch (zB S00)) aufn Databus (S99) und warten (bis der RAM Speicher die Daten übernommen hat - Zeitbedarf siehe RAM Chip Datenblattl)

↓ DataPointer-Inhalt auf den Adressbus legen
↓ Ergebnisregister-Inhalt auf den Datenbus legen
↓ WRITE-Signalleitung aktivieren
↓ Wartezeit für Speicherbausteine
↓ WRITE-Signalleitung de-aktivieren
↓ Adress- und Datenbus wieder freigeben ('Z' state)

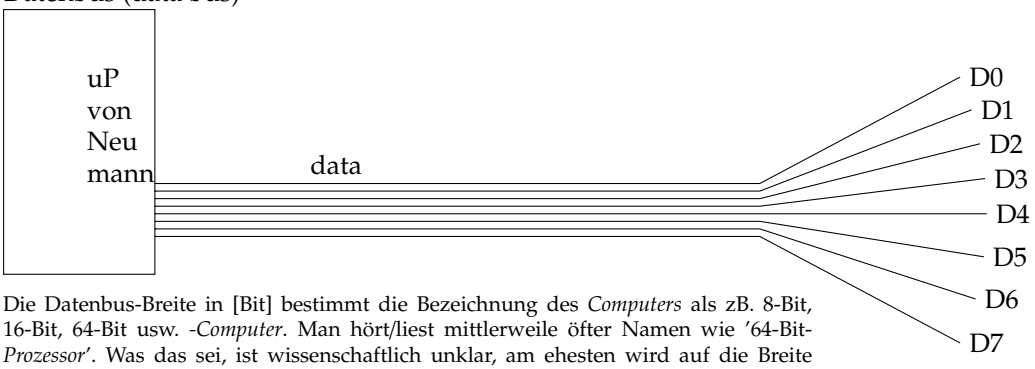
Inser Maschin hat nur 1 Memory Pointer - kanns also nur dorthin speichern, wos die Daten her hat. Wenns woanders hin sollen, miassmas auf mehrere Instructions aufteilen, zB.

'Load MP,X'
'Load A,MP'
'ADD A,B'
'Load MP,Y'
'Store MP,A'

Ez no den Instruction Pointer weiterzählen und wieder va vorn.

Computerbusse - vonNeumann architecture

1) Datenbus (data bus)

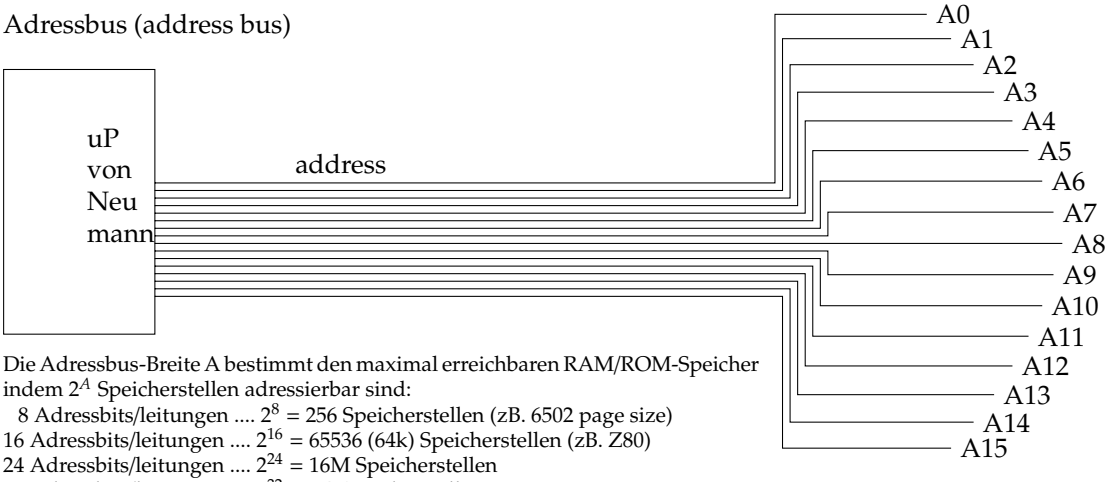


Die Datenbus-Breite in [Bit] bestimmt die Bezeichnung des *Computers* als zB. 8-Bit, 16-Bit, 64-Bit usw. -*Computer*. Man hört/liest mittlerweile öfter Namen wie '64-Bit-*Prozessor*'. Was das sei, ist wissenschaftlich unklar, am ehesten wird auf die Breite von 'general purpose' *Registern* / der ALU geachtet.

Es gibt nämlich Mikroprozessoren wie

- die Motorola 68008 // mit 16-Bit-Alu/Registern, aber 8-Bit-Datenbus
- die Intel 8088 // mit 16-Bit-Alu/Registern, aber 8-Bit-Datenbus
- die NS 32008 // mit 32-Bit-Alu/Registern, aber 8-Bit-Datenbus
- die NS 32016 // mit 32-Bit-Alu/Registern, aber 16-Bit-Datenbus

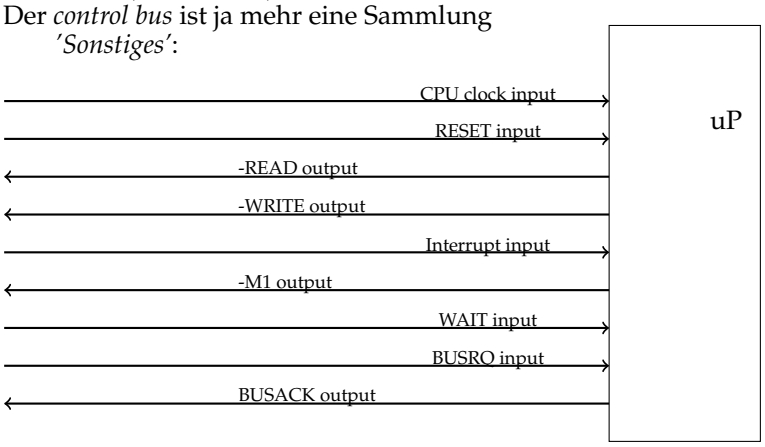
2) Adressbus (address bus)



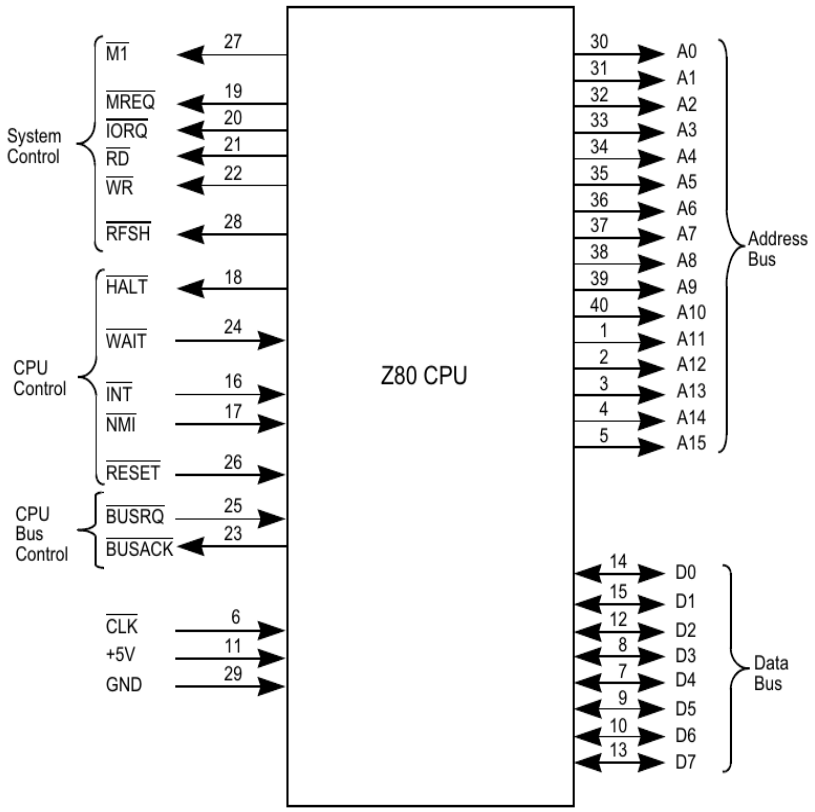
Die Adressbus-Breite A bestimmt den maximal erreichbaren RAM/ROM-Speicher indem 2^A Speicherstellen adressierbar sind:

8 Adressbits/leitungen .... 2^8 = 256 Speicherstellen (zB. 6502 page size)
16 Adressbits/leitungen .... 2^{16} = 65536 (64k) Speicherstellen (zB. Z80)
24 Adressbits/leitungen .... 2^{24} = 16M Speicherstellen
32 Adressbits/leitungen .... 2^{32} = 4G Speicherstellen

3) Steuerbus (control bus)
Der *control bus* ist ja mehr eine Sammlung *'Sonstiges'*:

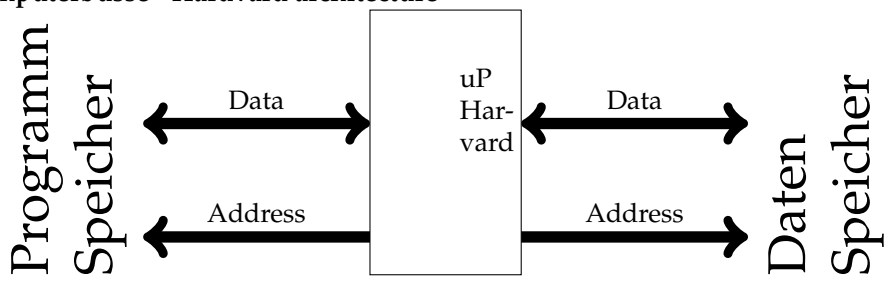


Z80-uP Mikroprozessor



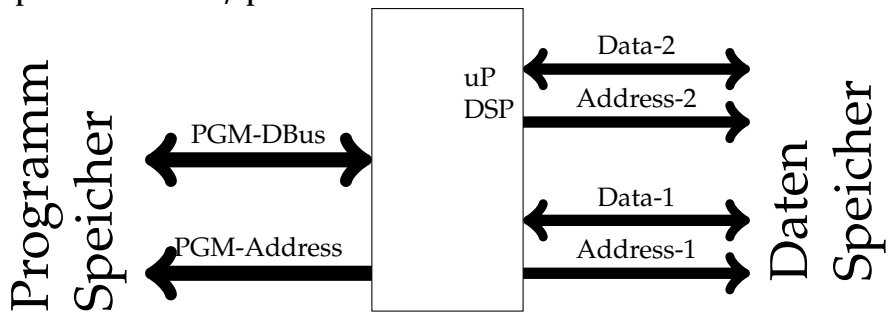
Die Zilog Z80, ein erweiterter Intel-8080 Nachbau, war Ende der 1970-ger Jahre eine extrem erfolgreiche 8-Bit-CPU, die zunächst in sog. Home-Computern eingesetzt, schliesslich vorwiegend in CP/M Rechnern mit FloppyDisks - Vorläufer des MS/DOS - die rasante Personal-Computer Entwicklung mit der heutigen Intel/Microsoft Vormachtstellung begründet haben.

Computerbusse - Hardvard architecture



- Harvard-Maschinen haben getrennte Programm- und Datenspeicher.
- Das unterstützt unterschiedliche Wortbreiten und steigert die Performance indem gleichzeitiger Zugriff auf Programmcode und Daten gelingt.
- zB. AVR-8Bit-Microcontroller (Arduino-Prozessor)
- aber: Es sind auch getrennte Datenbusse
- und Adressbusse notwendig

Computerbusse - DSP/special architectures

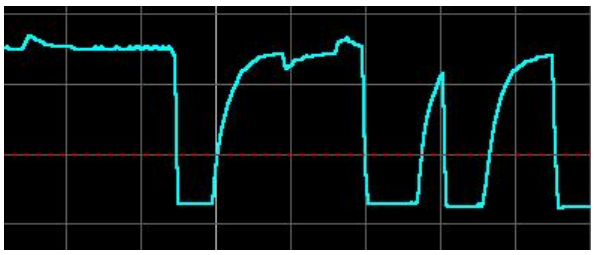


- Zur weiteren Performance-Steigerungen haben einige DSP getrennte RAMs für Samples und Koeffizienten, sodass es auch separate Daten- und Adressbusse braucht.
- Auch gibt es sog. *'dual ported RAM'*, welches zwei unterschiedliche RAM-Zugriffe voll gleichzeitig erlaubt.

Computerbus - Signalformen

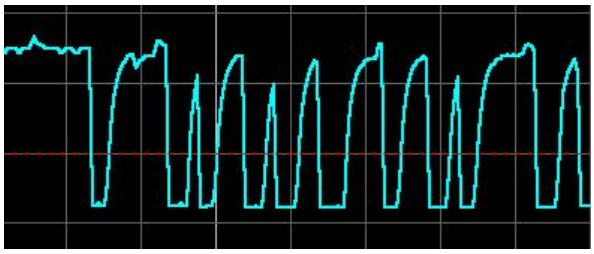
Zur Entgeisterung:

Bussignale sind in Wirklichkeit selten schön *rechteckig* wie im Buch,
sondern eher *"recht dreckig"*:



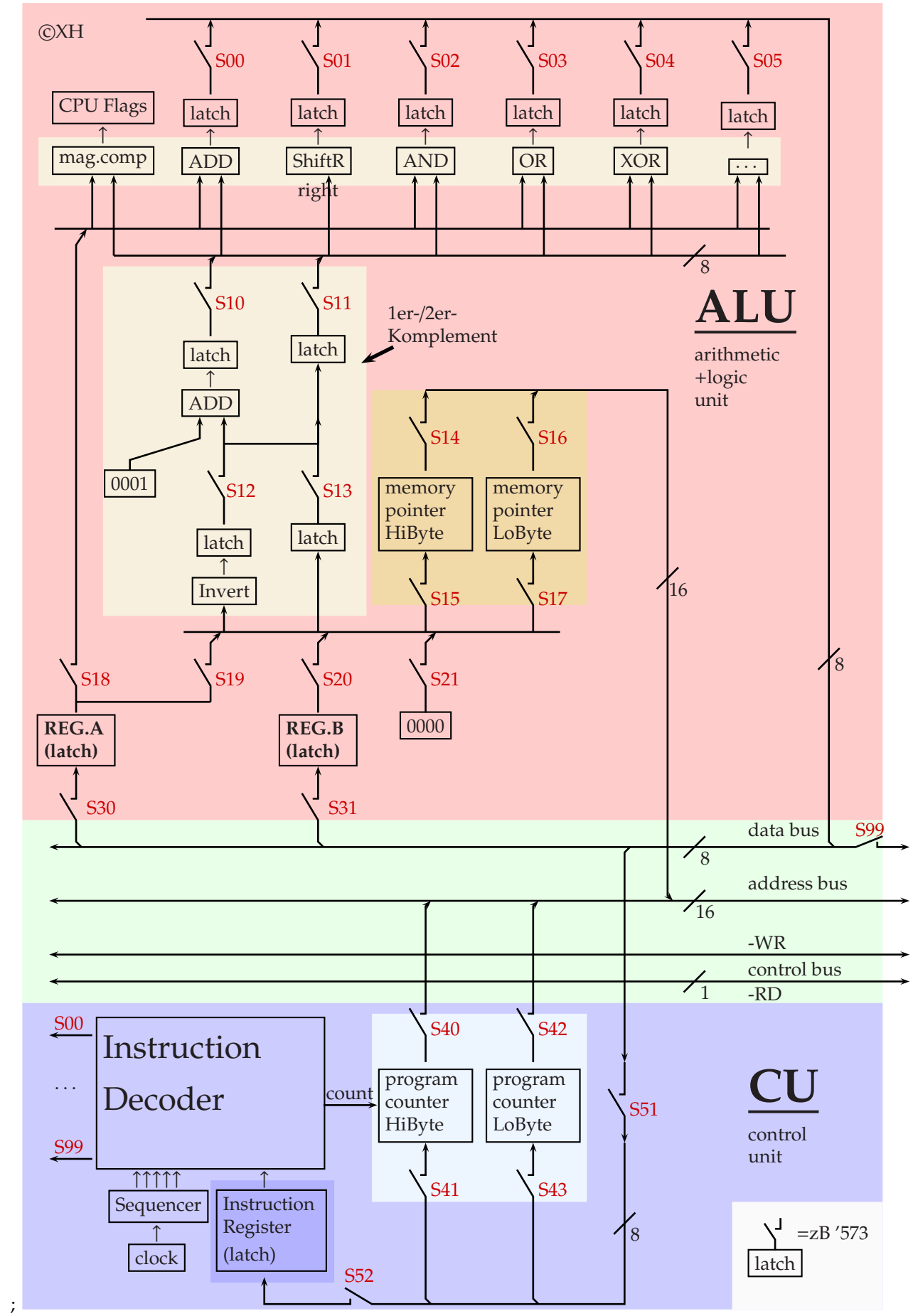
Von vielfach zeitgleichen Schaltvorgängen verursachte Vcc-Schwankungen *'vergiften'* die Signale.

Da die TTL-Spezifikation (s.TTL-Innenschaltung) stark unsymmetrisch ist (I_{OL}=-16mA, I_{OH}=+0,4mA), sind fallende FLanken 40 Mal steiler, während Anstiegsflanken nur müden RC-Anstieg zeigen



Deswegen sind *'eilige'* Signale *'früher' 'logisch negativ'* definiert worden
zB. -READ, -WRITE, -CS (chip select), -IRQ (interrupt)

Wenn kurze Pulse dadurch nicht mehr die volle Amplitude erreichen, wird es kritisch:
Weitere Takt-Steigerung wird zum *'Absturz'* führen.


©XH
S00
S01
S02
S03
S04
S05
CPU Flags
latch
mag.comp
ADD
ShiftR
right
AND
OR
XOR
...
8
ALU
arithmetic
+logic
unit
S10
S11
1er-/2er-
Komplement
ADD
0001
S12
S13
Invert
S14
S16
memory
pointer
HiByte
memory
pointer
LoByte
S15
S17
16
S18
S19
S20
S21
REG.A
(latch)
REG.B
(latch)
0000
S30
S31
data bus
S99
address bus
-WR
control bus
-RD
1
Instruction
Decoder
count
S40
S42
program
counter
HiByte
program
counter
LoByte
S41
S43
S51
CU
control
unit
Sequencer
clock
Instruction
Register
(latch)
S52
=zB '573

;

12.2 AVR-Simulator selbermachen

```
/* File:    avrEmu1.C   XH@24-Feb-2010
 * Version: 0.5, Bipr-2aHW, OOD+OOP
 * compile: g++ avrEmul.C
 * '.C' ist das originale C++ Suffix (Stroustrup),
 * nur konnte Kleinweich weder gross/klein unterscheiden,
 * noch '++' umsetzen.
 */

/*Grob-Design:
 *
 * main():
 *        !
 *        v
 * +------+
 * ^      |
 * |      v
 * |    read next Instruction from progMem
 * |    if Instruction == ADC then call function_ADC()
 * |    if Instruction == ... then call function_ ... ()
 * |    ...
 * |      |
 * |      v
 * +--<---+
 *
 *
 * function_XY..():
 *      modifiziere Register-,FLags-,RAM-,EPROM-,progMEM-Variablen
 *         Datenblatt-konform so, wie es der Controller mit der echten
 *         Hardware tun wuerde.
 *      return();
 *
 ************************************************************************/

/*---ProgrammCode:-------------*/

typedef byte unsigned char;

  // == ATTRIBUTES: Bestandteile als Variablen =======
byte progMem[1024];
byte eeProm [64];
byte sRam [64];
byte Reg [32];
int  carryFlag;
byte Timer;
int  ADConverter [4];
  ...


  /**** METHODEN: Verarbeitungen=Instructions als functions()
   *    == "Emulation" der Prozessor-Instructions
   ************************************************************/

void ADC( int d, int r, int C){
  Reg[d] = Reg[d] + Reg[r] + C;
}

void ADD( int d, int r ){
```

```
  Reg[d] = Reg[d] + Reg[r];
}

void ADIW( int d, int K ){
  Reg[d] = Reg[d] + K;
  carry = Reg[d] / 256;
  Reg[d] = Reg[d] % 256;
  Reg[d+1] = Reg[d+1] + carry;
}

void AND( int d, int r ){
  Reg[d] = Reg[d]  &  Reg[r];
}

void ANDI( int d, int K){
  Reg[d] = Reg[d]  & K;
}

void ASR( int d ){
  Reg[d] = Reg[d] / 2;
}

void EOR( int d, int r){
  Reg[d] = Reg[d]  ^  Reg[r];
}

  ... usw.


        /* da mein PC das uC-Programm nicht im Prog.Flash,
         * sindern in einer Datei stehen hat
         * muss diese Datei zuerst in die Variable 'ProgMem'
         * eingelesen werden:
         ********************************************************/
void uCProgrammEinlesen(char *fn){
  FILE *fp;
  long k;
  fp=fopen(fn,"rb");
  while( !feof(fp) ){
    progMem[ k++ ] = fgetc(fp);
  }
  fclose(fp);
} /* Programmfehler: Dieser Code ueberschreitet die Array-Grenze,
   * wenn die Datei zu gross ist */



        /* das ist jetzt der Simulator-Kern, die Verarbeitungen gem.
         * Instructions-Set Datenblatt:
         ********************************************************/
void BefehleAusfuehren(){
  int BefAdr=0;
  int InstructionCode;

  uCProgrammEinlesen();
  for( ; BefAdr<1023 ; ){
    InstructionCode =
          ProgMem [ BefAdr ] << 8)
```

```
        + ProgMem [ BefAdr+1 ];         //AVR-Prog.Flash is 16Bit-Wort-organisiert

    OpCode = 0xFC00 & InstructionCode;  //\
    d = ( InstructionCode >> 4 ) & 0x001F;      // +- FunktionsCode und
    r = (( InstructionCode >> 4 ) & 0x0010)     ///  Register-Nummern heraustrennen
       +( InstructionCode & 0x000F );

    if( OpCode == 0x1C00 ){ ADC(d,r); } // \
    if( OpCode == 0x0C00 ){ ADD(d,r); } // | Instructions auf-
    if( OpCode == 0x0100 ){ CPC(d,r); } // +-schluesseln und
    if( OpCode == 0x1400 ){ CP(d,r); }  // | Functions aufrufen
    // ...                              // / gem.Grobdesign s.o.

    BefAdr= (BefAdr+2)  // Prog.Mem ist 16-Bit(=2Byte)-Wort-orientiert
  }
}
```

12.3 TN13-Flash-Prommer selbermachen

```
/* TN13pArm.c xh@RfG,RfH,RfI,RfK,RfR,RfT, VcT1
 * AVR AT Tiny13/13A ('TN13') prommer @ linux-ARM(RaspberryPi)
 * compile: 'gcc -o TN13px TN13pArm.c'
 * stop:    <Return> am Ende
 * run:     usage: TN13px  <r(ead|w(rite|e)EEread|b(EEwrite|f(useRead|s|c|x(ecute>   <filename>
 *
 * Verdrahtung RasPi   --mit--  TN13:
 *      -------------          ------------------------color-
 *      Gpio_9  rPi:21         MISO    Dip8Pin: 6      VI
 *      Gpio10  rPi:19         MOSI    Dip8Pin: 5      YE
 *      Gpio11  rPi:23         SCK     Dip8Pin: 7      WH
 *      Gpio17  rPi:11         -       -
 *      Gpio22  rPi:15         Vcc     Dip8Pin: 8      OR
 *      Gpio27  rPi:13         -RESET  Dip8Pin: 1      GY
 *      GND     rPi: 6,14,20   GND     Dip8Pin: 4
 *      GND     rPi: 30,34,39  GND     Dip8Pin: 4      BK
 *      Gpio20  rPi:38         -       -        3      RD
 * ------------------------------------------------------------
 *
/* Grobdesign:
 * Programmierverfahren bitte im Datenblatt ATtiny13A
 * unter 'Memory Programming' - 'Serial Programming'
 * ''Both the Flash and EEPROM memory arrays can be programmed
 * using the serial SPI bus while
 *    ==>> RESET is pulled to GND. <<== !!!
 * ...
 *
 * Details bitte im Code lesen *grausig!* (Xh@VcT)
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <fcntl.h>


  //max(ATtiny13-FlashMem (16-bit-words!)):
#define AT_FLASHWORDS   (512)
#define AT_FLASHBYTES   (AT_FLASHWORDS * 2)
#define AT_PAGELEN_EXP  (4)
#define AT_PAGELEN      (1 << AT_PAGELEN_EXP)
#define AT_EEBYTES      (64)

#define AT_DATA_PWROFF  (0x000)
#define AT_DATA_PWRON   (0x0FF)
#define AT_CTRL_PWROFF  (0x00B | 0x20)
#define AT_CTRL_PWRON   (0x004)
#define AT_FUSE_HIGH    (0x0F3)
#define AT_FUSE_LOW1    (0x0E1)
#define AT_FUSE_LOW2    (0x0E2)
#define AT_FUSE_LOW3    (0x0E3)
#define AT_FUSE_LOW4    (0x0E4)

int             dataword=AT_DATA_PWROFF,ctrlword=AT_CTRL_PWROFF;
unsigned char   codemem[AT_FLASHBYTES];
int             filelen=0;
char            **mainargv;
int     fd_RESET,
        fd_MISO,
```

```
        fd_MOSI,
        fd_SCK,
        fd_XTAL,
        fd_PWR;

#define GPIO_DIR        "/sys/class/gpio"
#define GPIO_EXPORT     "/sys/class/gpio/export"
#define GPIO_UNEXPORT   "/sys/class/gpio/unexport"
#define AT_MISO_PIN     "/sys/class/gpio/gpio9/value"
#define MISO_NR         "9"
#define MISO_DIRECTION  "/sys/class/gpio/gpio9/direction"
int AT_MISO_VAL(){
  int r; char v=0;
  lseek( fd_MISO, 0L, SEEK_SET );
  r=read(fd_MISO, &v, 1);
//  printf("%2X ", v>0x30);
  return((v>0x030)? 1:0);
}
/*int get_AT_MISO_0_1(){
  int r; char v=0;
  lseek( fd_MISO, 0L, SEEK_SET );
  r=read(fd_MISO, &v, 1);
//  printf("%2X ", v>0x30);
  return((v>0x030)? 1:0);
} */
#define AT_MOSI_PIN     "/sys/class/gpio/gpio10/value"
#define MOSI_NR         "10"
#define MOSI_DIRECTION  "/sys/class/gpio/gpio10/direction"

#define AT_SCK_PIN      "/sys/class/gpio/gpio11/value"
#define SCK_NR          "11"
#define SCK_DIRECTION   "/sys/class/gpio/gpio11/direction"

#define AT_RESET_PIN    "/sys/class/gpio/gpio27/value"
#define RESET_NR        "27"
#define RESET_DIRECTION "/sys/class/gpio/gpio27/direction"

#define AT_XTAL1_PIN    "/sys/class/gpio/gpio17/value"
#define XTAL1_NR              "17"
#define XTAL1_DIRECTION "/sys/class/gpio/gpio17/direction"

#define AT_PWR_PIN      "/sys/class/gpio/gpio22/value"
#define PWR_NR          "22"
#define PWR_DIRECTION   "/sys/class/gpio/gpio22/direction"

#define clrPin(a)       write_dev((a), 0)
#define setPin(a)       write_dev((a), 1)
#define RESET           (fd_RESET)
#define MISO            (fd_MISO)
#define MOSI            (fd_MOSI)
#define SCK             (fd_SCK)
#define XTAL            (fd_XTAL)
#define PWR             (fd_PWR)

/*************************************
 * open/close device files
 *************************************/
void psystem(char *s){
  struct timespec ts={0L, 50*1000000L};
  printf("issue [%s]\n", s);
  nanosleep(&ts, NULL);
  system(s);
}
int open_devs(){
  int or;
  psystem("echo \"" PWR_NR "\">" GPIO_EXPORT);
  psystem("echo \"out\" >" PWR_DIRECTION);
  if(0> (fd_PWR  = open( AT_PWR_PIN, O_WRONLY)))   perror("PWR   pin " AT_PWR_PIN);

  psystem("echo \"" RESET_NR "\">" GPIO_EXPORT);
  psystem("echo \"out\" >" RESET_DIRECTION);
  if(0> (fd_RESET= open( AT_RESET_PIN, O_WRONLY))) perror("RESET pin " AT_RESET_PIN);

  if(!strchr("xX", *mainargv[1])){
   psystem("echo \"" MISO_NR "\">" GPIO_EXPORT);
   psystem("echo \"in\" >" MISO_DIRECTION);
   if(0> (fd_MISO = open( AT_MISO_PIN, O_RDONLY)))  perror("MISO  pin " AT_MISO_PIN);

   psystem("echo \"" MOSI_NR "\">" GPIO_EXPORT);
   psystem("echo \"out\" >" MOSI_DIRECTION);
   if(0> (fd_MOSI = open( AT_MOSI_PIN, O_WRONLY)))  perror("MOSI  pin " AT_MOSI_PIN);

   psystem("echo \"" SCK_NR "\">" GPIO_EXPORT);
   psystem("echo \"out\" >" SCK_DIRECTION);
   if(0> (fd_SCK  = open( AT_SCK_PIN, O_WRONLY)))   perror("SCK   pin " AT_SCK_PIN);

   psystem("echo \"" XTAL1_NR "\">" GPIO_EXPORT);
```

```
   psystem("echo \"out\" >" XTAL1_DIRECTION);
   if(0> (fd_XTAL = open( AT_XTAL1_PIN, O_WRONLY)))  perror("XTAL1  pin " AT_XTAL1_PIN);
  }
  or=fd_PWR|fd_RESET|fd_MISO|fd_MOSI|fd_SCK|fd_XTAL;
  printf("opened. %2X\n", or);
  if(0>or){ abort();}


}
int close_devs(){
  if(!strchr("xX", *mainargv[1])){
    close(fd_XTAL);
    close(fd_SCK);
    close(fd_MOSI);
    close(fd_MISO);
    psystem("echo \"" XTAL1_NR "\">" GPIO_UNEXPORT);
    psystem("echo \"" SCK_NR "\">" GPIO_UNEXPORT);
    psystem("echo \"" MOSI_NR "\">" GPIO_UNEXPORT);
    psystem("echo \"" MISO_NR "\">" GPIO_UNEXPORT);
  }
  close(fd_RESET);
  close(fd_PWR);
  psystem("echo \"" RESET_NR "\">" GPIO_UNEXPORT);
  psystem("echo \"" PWR_NR "\">" GPIO_UNEXPORT);
  printf("closed.\n");
}

/*****************************************
 * iopermp
 * (deprecated;
 * hatte auf ia86 mit IOpermissions zu tun...)
 *****************************************/
int iopermp(int m){
 static int iopcnt=0;
 return(
   (m ? iopcnt++ : --iopcnt) ? 0:
     (m? open_devs():close_devs())
 );
}

/*************************************
 * write to AT_XTAL1
 *************************************/
int write_dev(int fd, int v){
  return(
        0> write( fd, v?"1":"0", 1) ?
          perror("write_dev"),0 : 0
        );
}

/*************************************
 * wait n [us]
 *************************************/
void wait_us(unsigned short nus){            //mind n us
  int i;
  struct timespec tw={0L,1L};
  tw.tv_nsec=nus*1000L;
  nanosleep(&tw, NULL);
}

/*************************************
 * wait 3us, 10us 100us, 1ms, 10ms
 *************************************/
void wait3us()  { wait_us(3);   }       //8
void wait10us() { wait_us(13);  }       //23
void wait100us(){ wait_us(133); }       //233
void wait1ms()  { wait_us(1330);}       //2330
void wait10ms() { int i; for(i=0;i<10;i++) wait1ms(); }

/*************************************
 * v_batt an-/abschalten
 *************************************/
void vb_on(){  clrPin(RESET);clrPin(SCK);clrPin(MOSI);setPin(PWR); }
void vb_off(){ clrPin(RESET);clrPin(SCK);clrPin(MOSI);clrPin(XTAL);clrPin(PWR);}


/*************************************
 * send 1 byte to avrXX
 *************************************/
int send1b_avr(int b){
  int           i;
  unsigned int  ctl=0;
  b = b & 0x0FF;
  if(iopermp(1)) perror("ioperm in send1b");
  for(i=0;i<8;i++){
    if(b & 0x080)       setPin(MOSI);
    else                clrPin(MOSI);
```

```
    b = b << 1;
    setPin(SCK);
    wait3us();
    ctl = (ctl << 1) | (AT_MISO_VAL() & 0x01);
    clrPin(SCK);
    wait3us();
  }
  iopermp(0);
  return(ctl);
}

/*************************************
 * leave_programming_mode AVR
 *************************************/
void leave_programming_mode(){
        //programming is done with -RESET==Low
 wait100us();
 setPin(SCK);
 setPin(MOSI);
 setPin(XTAL);
 clrPin(RESET);
 wait10ms();
 vb_off();
}


/*************************************
 * reset AVR
 *************************************/
void reset_avr(){
        //programming is done with -RESET==Low
        //"give -RESET a positive pulse of atleast 2 clock cycles while SCK==0"
  int   i;
  struct timespec tw={0L,10L*1000000L};
  vb_on();
  for(i=0;i<100;i++) wait1ms();
  clrPin(SCK);
  wait100us();  clrPin(RESET);
  nanosleep(&tw,NULL);  setPin(RESET);
  nanosleep(&tw,NULL);  clrPin(RESET);
//  nanosleep(&tw,NULL);        setPin(SCK);    //outb(AT_SCK_1,   AT_SCK_REG);
  wait10ms();
}


/*************************************
 * send "programming enable" instruction
 *      rv < 0 === ERROR
 *************************************/
int pgmenable_avr(){
  int    rv=-1,i;
  for(i=0;(rv<0) && (i<10);i++){
    reset_avr();
    printf("pgmenable_avr: %2X\t", send1b_avr(0x0AC));
    printf("%2X\t", send1b_avr(0x053));
    printf("%2X\t", rv= (send1b_avr(0x0FE)==0x053 ? 0 : -1));
    printf("%2X\n", send1b_avr(0x0FE));
  }
  if(rv<0) printf("pgm enable ERROR\n");
  return( rv );
}


/*************************************
 * send "chip erase" instruction
 *************************************/
int chiperase_avr(){
  send1b_avr(0x0AC);
  send1b_avr(0x08F);
  send1b_avr(0x0FE);
  send1b_avr(0x0FE);
  wait10ms(); wait10ms();
  return( 0 );
}


/*************************************
 * send "prog fuse bits" instruction
 *************************************/
int prog_fusebits(int n){
  //1010-1100 1010-1000 0000-0000 AT_FUSE_HIGH
  //1010-1100 1010-0000 0000-0000 AT_FUSE_LOW
  send1b_avr(0x0AC);
  send1b_avr(0x0A8);
  send1b_avr(0x0FE);
  send1b_avr(AT_FUSE_HIGH);
  wait10ms(); wait10ms();
```

```
  send1b_avr(0x0AC);
  send1b_avr(0x0A0);
  send1b_avr(0x0FE);
  send1b_avr(n);
  wait10ms(); wait10ms();
  return( 0 );
}


/*************************************
 * program 1 word to ATtiny13V
 *************************************/
void burn1w_avr(int adr,unsigned int w){
  // [...] "data low byte must be loadad before" [...]
  if(w != 0x0FFFF){
    send1b_avr(0x040);
    send1b_avr(0x00F);
    send1b_avr(0x00F & adr);
    send1b_avr(w & 0x0FF);
    wait10us();

    send1b_avr(0x048);
    send1b_avr(0x00F);
    send1b_avr(0x00F & adr);
    send1b_avr((w>>8) & 0x0FF);
    wait10us();
  }
}


/*************************************
 * read 1 word from AVR
 *************************************/
int read1w_avr(int adr){
  int   rv=0;
  send1b_avr(0x028);
  send1b_avr((adr>>8) & 0x0F);
  send1b_avr( adr & 0x0FF);
  rv= (send1b_avr(0x0FE) << 8);
  wait10us();
  send1b_avr(0x020);
  send1b_avr((adr>>8) & 0x0F);
  send1b_avr( adr & 0x0FF);
  rv=(rv | (send1b_avr(0x0FE) & 0x0FF));
  wait10us();
  return(rv);
}


/*************************************
 * read 1 byte EEPROM from AVR
 *************************************/
unsigned int readEE_avr(int adr){
  unsigned int  rv=0;
  send1b_avr(0x0A0);
  send1b_avr(0x000);
  send1b_avr( adr & 0x03F);
  rv= (send1b_avr(0x055));
  wait10us();
  return(rv);
}


/*************************************
 * burn 1 byte EEPROM to AVR
 *************************************/
int burnEE_avr(int adr, unsigned int data){
  int    i;
  if(readEE_avr(adr) != data){
    send1b_avr(0x0C0);
    send1b_avr(0x000);
    send1b_avr( adr & 0x03F);
    send1b_avr( data & 0x0FF );
    for(i=0;i<15;i++) wait1ms();
    for(i=0;i<1000;i++){
      if(readEE_avr(adr) == data) return(0);
      wait1ms();
    }
    wait10us();
    return(-1);
  }else  return(0);
}


/*************************************
 * read calibration byte from AVR
```

```
 ***************************************/
unsigned long readcalibration_avr(){
  unsigned long    rv=0;
  if(pgmenable_avr() < 0) return(-1);
  send1b_avr(0x038);
  send1b_avr(0x0FE);
  send1b_avr(0x003);
  rv= ((send1b_avr(0x0FE) & 0x0FFL) << 24L);
  send1b_avr(0x038);
  send1b_avr(0x0FE);
  send1b_avr(0x002);
  rv= rv | ((send1b_avr(0x0FE) & 0x0FFL) << 16L);
  send1b_avr(0x038);
  send1b_avr(0x0FE);
  send1b_avr(0x001);
  rv= rv | ((send1b_avr(0x0FE) & 0x0FFL) << 8L);
  send1b_avr(0x038);
  send1b_avr(0x0FE);
  send1b_avr(0x000);
  rv= rv | ((send1b_avr(0x0FE) & 0x0FFL ) );
  wait10us();
  return(rv);
}


/***************************************
 * read fuse bits word from AVR
 ***************************************/
int readfusew_avr(){
  int   rv=0;
  if(pgmenable_avr() < 0) return(-1);

  send1b_avr(0x058);
  send1b_avr(0x08);
  send1b_avr(0x0FE);
  rv= (send1b_avr(0x0FE) << 8);
  wait10us();
  send1b_avr(0x050);
  send1b_avr(0x00);
  send1b_avr(0x0FE);
  rv=(rv | (send1b_avr(0x0FE) & 0x0FF));
  wait10us();
  return(rv);
}


/***************************************
 * read signature bytes from AVR
 ***************************************/
unsigned long readsig_avr(){
  unsigned long rv=0;
  if(pgmenable_avr() < 0) return(-1);

  send1b_avr(0x030);
  send1b_avr(0x0FE);
  send1b_avr(0x07F);
  rv= ((send1b_avr(0x0FE) & 0x0FFL) << 24L);
  wait10us();
  send1b_avr(0x030);
  send1b_avr(0x0FE);
  send1b_avr(0x0FE);
  rv= rv | ((send1b_avr(0x0FE) & 0x0FFL) << 16L);
  wait10us();
  send1b_avr(0x030);
  send1b_avr(0x0FE);
  send1b_avr(0x0FD);
  rv= rv | ((send1b_avr(0x0FE) & 0x0FFL) << 8L);
  wait10us();
  send1b_avr(0x030);
  send1b_avr(0x0FE);
  send1b_avr(0x0FC);
  rv= rv | (send1b_avr(0x0FE) & 0x0FFL);
  wait10us();
  return(rv);
}


/***************************************
 * uC-code einlesen
 ***************************************/
void loadcode(char *fname){
  FILE  *fp;
  int   i;
  for(i=0;i<AT_FLASHBYTES;i++) codemem[i]=0x0FF;
  if(NULL!=(fp=fopen(fname,"rb"))){
    filelen=fread(codemem,1,AT_FLASHBYTES,fp);
    fclose(fp);
```

```
  }
}


/*************************************
 * read EE and store to file
 *************************************/
int AT_EE_read(char *fn){
  FILE  *fp;
  unsigned int   rv=0,adr,dat;
  if(NULL==(fp=fopen(fn,"wb"))){
    printf("cant open file /%s/ for writing AVR EEPROM contents",fn);
  }else{
    printf("reading ATtiny13 EEPROM and writing to /%s/...",fn);
    if(pgmenable_avr() < 0){
      printf("\ncant enter programming mode.\n");
      rv=-1;
    }else{
      for(adr=0;adr<AT_EEBYTES;adr++){
        codemem[adr]= 0x0FF & readEE_avr(adr);
      }
     filelen=fwrite(codemem,1,AT_EEBYTES,fp);
     printf("done\n");
   }
   fclose(fp);
  }
  return(rv);
}


/*************************************
 * program EEPROM AVR
 *************************************/
int progEE_avr(){
  int    adr;
  if(pgmenable_avr() < 0){
    printf("cant enter programming mode -");
    return(-1);
  }
  for(adr=0;adr<AT_EEBYTES;adr++)
    if(burnEE_avr(adr,codemem[adr])) return(-1);
  return(0);
}


/*************************************
 * prom EE from file
 *************************************/
void AT_EE_prom(char *fn){
  printf("loading from file /%s/...",fn);
  loadcode(fn);
  printf("file len=%d\n",filelen);
  printf("%02X %02X %02X %02X   %02X %02X %02X %02X   %02X %02X %02X %02X ...\n",
        codemem[0],codemem[1],codemem[2],codemem[3],codemem[4] ,codemem[5],
        codemem[6],codemem[7],codemem[8],codemem[9],codemem[10],codemem[11]);
  if(progEE_avr()){
    printf("EEPROM programming Error.\n");
  }
}


/*************************************
 * write flash memory page to ATtiny13V
 *************************************/
void writeflashpage_avr(int adr){
  send1b_avr(0x04C);
  send1b_avr((adr >> 8) & 0x001);
  send1b_avr((adr | (AT_PAGELEN-1)) & 0x0FF);
  send1b_avr(0x0FF);
  wait10ms();
}


/*************************************
 * program ATtiny13V flash
 *************************************/
int checkPage4FF(int pg){
  int   i;
  for(i=(pg*AT_PAGELEN);i<((pg+1)*(AT_PAGELEN));i++){       //words!
    if(codemem[i+i]!=0x0FF) return(0);
    if(codemem[i+i+1]!=0x0FF) return(0);
  }
  return(1);
}

/*************************************
 * program ATtiny13V flash
```

```
 ************************************/
int burnPage(int pageno, int *adr, int n){
  int j;
  for(j=0;(j<(AT_PAGELEN)) && (*adr<n);j++){
    burn1w_avr(*adr,(codemem[*adr+*adr+1] << 8) + codemem[*adr+*adr]);
    (*adr)++;
  }
  wait10ms();
}

/************************************
 * program ATtiny13V flash
 ************************************/
int progflash_avr(){
  int   adr,j,pageno,n=filelen/2;
  if(pgmenable_avr() < 0) return(-1);
  chiperase_avr();
  for(j=0;j<20;j++) wait10ms();
  for(adr=0;adr<n;){
    pageno=(adr)>>(AT_PAGELEN_EXP);
    printf("p%d",pageno);
    if(!checkPage4FF(pageno)){
      burnPage(pageno,&adr,n);
      writeflashpage_avr(adr-1);
      printf("W,");
    }else{
      adr+=(AT_PAGELEN);
      printf("*,");
    } fflush(stdout);
  }
  return(0);
}


/************************************
 * read ATtiny13V flash
 ************************************/
int readflash_avr(){
  int           adr,j,pageno,n=AT_FLASHWORDS;
  unsigned int  rv;
  if(pgmenable_avr() < 0) return(-1);
  for(adr=0;adr<n;){
    pageno=(adr)>>4;
    printf("p%d",pageno);
    for(j=0;(j<16) && (adr<n);j++){
      rv=read1w_avr(adr);
      codemem[adr+adr+1]=(rv>>8) & 0x0FF;
      codemem[adr+adr]=rv & 0x0FF;
      adr++;
    }
    printf("+"); fflush(stdout);
  }
  return(0);
}


/************************************
 * prom flash from file
 ************************************/
void AT_prom(char *fn){
 printf("loading from file /%s/...",fn);
 loadcode(fn);
 printf("file len=%d\n",filelen);
 printf("%02X %02X %02X %02X   %02X %02X %02X %02X   %02X %02X %02X %02X ...\n",
        codemem[0],codemem[1],codemem[2],codemem[3],codemem[4], codemem[5],
        codemem[6],codemem[7],codemem[8],codemem[9],codemem[10],codemem[11]);
 progflash_avr();
}

/************************************
 * read flash and store to file
 ************************************/
void AT_read(char *fn){
 FILE   *fp;
 if(NULL!=(fp=fopen(fn,"wb"))){
  printf("reading AVR and writing to /%s/...",fn);
  if(readflash_avr()<0)
        printf("\ncant enter programming mode.\n");
  else{
        filelen=fwrite(codemem,1,AT_FLASHBYTES,fp);
        printf("done\n");
  }
  fclose(fp);
 }else{
  printf("cant open file /%s/ for writing ATtiny13 contents",fn);
 }
}
```

```
//C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//C+++   M A I N       Procedure
//C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
main(int argc, char *argv[]){
  int           i,l;
  char          s1[300];
  FILE          *fp;
  struct timespec t3s={3L,1L};

  mainargv=argv;//global
  if(argc<3){
    printf("usage: TN13px  <r(ead|w(rite|e)EEread|b(EEwrite|f(useRead|s|c|x(ecute>  <filename>\n");
    abort();
  }

  iopermp(1);
  vb_on();

  switch(*argv[1]){
  case 'R':
  case 'r':     AT_read(argv[2]);
                break;
  case 'W':
  case 'w':     AT_prom(argv[2]);
                break;
  case 'E':
  case 'e':     AT_EE_read(argv[2]);
                break;
  case 'B':
  case 'b':     AT_EE_prom(argv[2]);
                break;
  case 'F':
  case 'f':     printf("fuse bits word = /%04X/\n",readfusew_avr());
                break;
  case 'S':
  case 's':     printf("signature bytes = /%08lX/\n",readsig_avr());
                break;
  case 'C':
  case 'c':     printf("calibration byte = /%08lX/\n",readcalibration_avr());
                break;
  case 'X':
  case 'x':
                printf("reseting TN13...");
                clrPin(RESET);
                wait10ms();
                setPin(RESET);
                for(i=0;i<30;i++) wait10ms();
                printf("TN13 running...");
                break;

  //for testing only:
  case '1':     for(;;){
                 wait100us();
                 iopermp(1);
                  setPin(PWR);
                  setPin(RESET);
                  setPin(XTAL);
                  setPin(SCK);
                  setPin(MOSI);
                 wait100us();
                 nanosleep(&t3s,NULL);
                  clrPin(RESET);
                  clrPin(XTAL);
                  clrPin(SCK);
                  clrPin(MOSI);
                  clrPin(PWR);
                 iopermp(0);
                 putchar('.');
//               if(kbhit()) break; ... aus MSDOS Zeiten :-))
                 if('q'==getchar()) break;
                }
                break;
  case '2': for(;;){ pgmenable_avr();if(getchar()) break;} break;
  case '3': for(;;){ AT_read(argv[2]);if(getchar()) break;} break;
  case '4':     loadcode(argv[2]);
                if(NULL!=(fp=fopen("CODEMEM","wb"))){
                 filelen=fwrite(codemem,1,AT_FLASHBYTES,fp);
                 fclose(fp);
                }
                break;
   case '5':    if(pgmenable_avr() >= 0){
                 prog_fusebits(AT_FUSE_LOW1);
                }
                break;
   case '6':    if(pgmenable_avr() >= 0){
```

```
                prog_fusebits(AT_FUSE_LOW2);
               }
               break;
   case '7':   if(pgmenable_avr() >= 0){
                prog_fusebits(AT_FUSE_LOW3);
               }
               break;
   case '8':   if(pgmenable_avr() >= 0){
                prog_fusebits(AT_FUSE_LOW4);
               }
               break;

  default:     printf("unknown command /%s/\n",argv[1]);
               break;
  }

  printf(">");getchar();


l_ende:
  leave_programming_mode();
  vb_off();
  iopermp(0);
  return (0);
}

/* rPi pin assignment f. ATN13:
 * Funktion     GPIO    PfostensteckerPin
 * --------------------------------------
 * +3V3                 1,17
 * +5V0                 2,4
 * GND                  6,9,14,20,25,30,34,39
 * SDA                  3
 * SCL                  5
 * GND                  9
 * AT_XTAL1_PIN gpio17  Pin11
 * AT_RESET_PIN gpio27  Pin13
 * AT_PWR_PIN   gpio22  Pin15
 * +3V3                 Pin17
 * AT_MOSI_PIN  gpio10  Pin19
 * AT_MISO_PIN  gpio9   Pin21
 * AT_SCK_PIN   gpio11  Pin23
 * GND                  25
 *              GPIO05  29
 *              GPIO06  31
 *              GPIO13  33
 *              GPIO19  35
 *              GPIO26  37
 *              Gpio20  38
 * GND                  39
 */
```

12.4 PSOC5LP

12.4.1 Sehrgutaufgabe:

Inputsignal → ADC → FIRfilter → DAC → Output

12.4.2 Di-26Apr'22:

lbr Simon, Simon, Patrick,
Danke vielmals fyr Dei Geduld und Deine durchaus sinnvollen Versuche mit dem pSocken und mirXH;
Hinweis:

- 'BLINKLED' is ka echtes Device, sondern a 'dummy' – do siehgma Nix.
- fyr die realen LEDs auf dem Print ziehtma so einen 'Digital-I/O Port' in den Schaltplan und gibt ihm einen Namen (Doppelklick und eintragen, zB. 'PORT1')
- in dem Window mit den Pinzuordnungen (IC-Outline) waehlt man einen Pin lt. LED-Platinenbeschriftung (zB. '12.2')
- Im C-Programmcode schreibt man dann
  `PORT1_Write( 0 | 1 )`
  bzw.
  `rv = PORT1_Read()`
- das Blinkprogramm waere dann etwa:

```
for(;;){ //forever-loop
   PORT1_Write(1);
   CyDelay(500);
   PORT1_Write(0);
   CyDelay(500);
}
```

12.4.3 Monday 25Apr'22:

Hi chum(p),
we (you) will start using se PSOC. Beeing said that your Klasse didn't use PSOC ever b4 (=before) gained some XH-shocking, so it will b suitable 2 begin ganz unten at level '00'. Maybe u didnt even install the 'PSOC Creator' developement tool so far. Coming Monday therefore we will

↪ discuss the first steps
↪ let Aze complain of 'C'
↪ check se own PC
↪ ggf. download und install se 'PSOC Creator'
↪ fire it up and configure
↪ check se PSOC5LP, given se oppurtunity dass u did it mitbring

(if u refuse collaborating/participating in or sorg dafür, that u cannot mitmach, u simply will pull se next worse Mitarbeitsnote and out)

12.4.4 what we brauch first:

Für die ersten Schritte brauchen wir

★ se Windows-PC
(das Cypress-pSOC developement tool 'psoc Creator'[2] lauft nur auf Wintox)

Zum Installieren/Downloaden des psoc-Creator[3] ist das pSOC5LP-Modul nicht unbedingt erforderlich; auch `C-Programmcode eintippen` und `syntax check` gehen ohne.

12.4.5 getting started

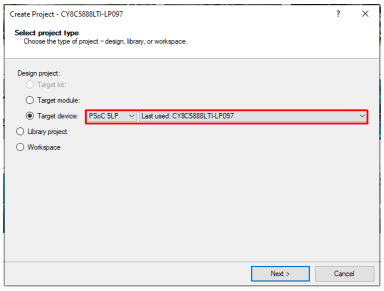
Allerdings wird empfohlen, bereits unmittelbar nach Eröffnen eines *new Project* die (beabsichtigte) pSOC-interne

↪ `Hardware in den (leeren) Schaltplan` hinein zu *ziehen*,
da sonst deren API-Module
↪ im Programmcodeeditor `nicht angezeigt` werden und
↪ `nicht ansprechbar` sind

Erst beim Upload des fertig compilierten und gelinkten `Binary` sind Anschluss und exakte Auswahl des PSOC5LP-Moduls erforderlich.

[2] PSoC Creator is the second generation software IDE to design debug and program the PSoC 3 / 4 / 5 devices. The development IDE is combined with an easy to use graphical design editor to form a powerful hardware/software co-design environment.

[3] Download PSoC Creator from www.cypress.com/psoccreator, or install from a kit CD. For assistance, call Cypress Support at 1-800-541-4736 and select 8. For features, system requirements, and installation notes, refer to the Release Notes available at: www.cypress.com/go/creator/releasenotes



siehe auch *PSOC Creator Qick Start Guide* , zB.: https://www.cypress.com/file/195271/download

12.4.6 What language does PSoC creator use?

Cypress Creator IDE is using **C Language** as standard. There are some tricks to have C++ compiled. Achtung: C++: Objektorientierte und Script-Systeme sind **nicht realtime-fähig!** Durch die ständige Objekt-Erzeug und -Löscherei (bes. der verborgenen, automatisch verwalteten (namenlosen) Objekte) wird der *Garbage Collector* unkontrollierbar involviert, sodass weder Speicherbedarf noch Laufzeitverhalten vorhersagbar in Griff zu kriegen sind; C++, Java, Python, HTML udgl. Kinderkram sind also Weicheierprogrammiersprachen für Webseiten und GUI-Farbenspiele, aber für technische hard-realtime-Automatisierungsanwendungen unbrauchbar. Bsp.: Derart programmierte mehrpolige Elektromotorantriebe schlagen mit ihren Steuersignal-Jitter-verursachten Torsionsschwingungen sämtliche Lager und Getriebe rasch zu Schrott — verwende für so einen Zweck den kleinstmöglichen uC (ohne 'Multitasking' oder OS!), den unmittelbar DU SELBST programmierst (und nicht irgendein Entwicklungssystem oder 'Framework')

12.5 PSOC5LP Code-Beispiele

12.5.1 Beispiel 'LedBlink'

```
void main(){
    for( ; ; ){         /* <-- so schreibt man 'ewige Schleifen' in 'C'*/

       LED_Write( ~LED_Read() );

        CyDelay(500);                       // delay in [ms]
    }
}
```

das Creatorzeug hängt '_Write' und '_Read' dran

Den Namen 'LED' vergibst Du in der Creatorschaltung.
Verwende einen 'Digital-Pin'.
Ordne ihn einem PSOC5LP-Anschluss, zB. '12.2', zu.

12.5.2 Beispiel '2Led'

```
int main(void){
    Led1_Write(0);
    Led2_Write(0);
    for(;;){
        Led1_Write(1);
        CyDelay(500);           Cy... 'Cypress'Zuig */
        Led1_Write(0);
        Led2_Write(1);
        CyDelay(500);
        Led2_Write(0);
    }
}
```

12.5.3 Beispiel 'Button Read'

```
int main(void){
  for (;;){
    if (BUTTON_Read() == 0){    //Button pressed
        ein: LED1_Write(1);
    else
        aus: LED1_Write(0);
  }
}
```

12.5.4 Beispiel 'FSM'

```
/* XH: der Code is ziemlicher C-Murks!
   -ka Entwurf
   -ka Beschreibung
   -wirre Variablennamen
   -von der konfigurierten Hardware weiss der Autor offenbar selber nix
   Zustaende benennt man Q... (Q1, Q2, Q3...)
   Schalter (auch Tast-Schalter) benennt man S... (S1, S2, S3...)
 */
int iZustand= 0;
int iTasterWurdeGedrueckt= 0;
void SelectFsmLed(){
  switch(iZustand){
        case(0): //nach Einschalten
          Led2_Write(1);
          Led3_Write(0);
          Led4_Write(0);
          iZustand= 10;
          break;
        case(10):
          if (Taster_Read()== 0){          //Wenn Taster gedrückt => Taster_Read = 0
            Led2_Write(1);
            Led3_Write(0);
            Led4_Write(0);
            iZustand= 20;
            iTasterWurdeGedrueckt= 1;
          }
          break;
        case(20):
          if (Taster_Read()== 0){
            Led2_Write(0);
            Led3_Write(1);
            Led4_Write(0);
            iZustand= 30;
            iTasterWurdeGedrueckt= 1;
          }
          break;
        case(30):
          if (Taster_Read()== 0){
            Led2_Write(0);
            Led3_Write(0);
            Led4_Write(1);
            iZustand= 10;
            iTasterWurdeGedrueckt= 1;
          }
          break;
  }
}
int main(void){
  for(;;){
    SelectFsmLed();
    if(iTasterWurdeGedrueckt==1){       //Delay, dass bei einem Tastendruck nicht mehrere Leds hintereinander angeher
        CyDelay(250);
    }
    iTasterWurdeGedrueckt= 0;
  }
}
```

```
/* Bereinigung:
 * asynchronous (no clock) FiniteStateMachine */
Led2_Write(1); Led3_Write(0); Led4_Write(0);
int Qn= 10;
for(;;){
  while(Taster_Read() != 0){    //key pressed == 0
  CyDelay(250);                 //entprellen == warten
  switch( Qn ){
  case(10):
        Led2_Write(1); Led3_Write(0); Led4_Write(0);
        Qn=20;
        break;
  case(20):
        Led2_Write(0); Led3_Write(1); Led4_Write(0);
        Qn=30;
        break;
  case(30):
        Led2_Write(0); Led3_Write(0); Led4_Write(1);
        Qn=10;
        break;
  }
}
```

12.5.5 Beispiel 'Timer-Interrupt'

```
  /* hier fehlt total die Beschreibung der Hardwarekonfiguration */

CY_ISR(TimerTC_isr){                    //'ISR'=InterruptServiceRoutine
  intr_timer_ReadStatusRegister();      //clear interrupt_request
  INTR_LED_Write(1);                    //set_handler_running_LED
  MAIN_LED_Write(0);                    //clear_main_running_LED
  CyDelay (1000);                       //synchronous tasks_dummy
  INTR_LED_Write(0);                    //clear_handler_running_LED
}

int main (void){
  CYGlobalIntEnable;
  intr_timer_Start();                   //4s-periodic-interrupt
  timer_isr_StartEx(TimerTC_isr);       //set vector_adress for timer_isr
  for(;;){                              /*(LEDs wern in ISR gschaltet)*/
    MAIN_LED_Write(1);
    CyDelay(2);                         /*XH: Multitasking ==> Wartezustaende einbauen!*/
  }
  return 0;
}
```

12.5.6 Beispiel 'LED mit Poti und PWM'

```
// Aufgabe für Herrn. Wechner (eine LED mit Potentiometer und PWM Signal steuern.  23.05.2022

/* Beschreibung zum Aufbau:
 1) Zuerst wird Psoc_Creator gestartet.
 2) Dann wird ein neues Workspace unter dem Namen LED_Potentiometer erstellt.
 3) Im TopDesgin wird
        ein Digital Input,
        ein DigitalOuput,
        ein Analog-Digital_Converter,
        ein Puls-Weiten-Modulator,
        eine 1,2 MHz Colck und
        ein Digital_low plaziert.
 4) Den DigitalInput nenne ich Potentiometer.
 5) Man muss Drive mode Resistive pull up/down einstellen.
 6) Den Digital Output nenne ich LED.
 7) Bei dem wird als Drive Mode Strong drive eingestellt werden.
 8) Beim ADC wird nichts eingestellt und
 9) bei der PWM wird die PWM Mode auf One Output eingestellt.
10) Der Poti und die Colck werden mit dem ADC verbunden.
11) Die Clock und der digital-low werden in die PWM geleitet.
12) Von der PWM wird dann in die LED gefahren.
13) In der cydwr Datei müssen noch die Pins vergeben werden.
14) Der Poti ist auf dem Pin P3(0) gelegt und
15) die LED wird auf den Pin P12(2) gelegt.
16) Danch muss der Potentiometer auf der Platine noch mit einem Kabel zum Pin P3(0) verbunden werden und
17) der VCC des Potis wird zum 3,3V Quelle verbunden und
18) der GND wird zu einem aktiven GND verbunden.
19) Danach wird alles mit dem Programm der darunter angegeben ist geregelt.
*/

#include "project.h"
int main(void){
    CyGlobalIntEnable;
    ADC_Start();                // AnalogDigitalWandler wird gestartet
    PWM_Start();                // Pulsweiten Modulation wird gestartet
    ADC_StartConvert();         // Die konvnetierung der Analogen Werte wird gestartet
    for(;;){//endless loop
       PWM_WriteCompare(ADC_GetResult8()); // Die Ergebnise des ADW werden als 8 bit in das PWM geschickt
    }
}//main

// Der Screenshoot vom TopDesign ist in dem PSOC_Bilder ordner gespeichert.
```

12.5.7 Beispiel 'LED mit Taster und PWM'

```
// Autor: Popeye
// Project bei Herrn Wechne, es war auf eine LED mit eiem Taster zu steuern(ein, aus, und Helligkeit dimmen) 30.05.2022
/* Beschreibung zum Aufbau:
 1) Zuerst wird Psoc_Creator gestartet.
 2) Dann wird ein neues Workspace unter dem Namen LED_Taster erstellt.
 3) Im TopDesgin wird
        ein Digital Input,
```

```
        ein DigitalOuput,
        ein digital-low,
        eine 1,2HMhz Clock und
        ein PWM plaziert.
 4) Den DigitalInput nenne ich Button.
 5) Man muss bei ihm die HW connection ausschalten und
 6) als Drive mode Resistive pull up/down einstellen.
 7) Den Digital Output nenne ich LED.
 8) Bei ihm wird als Drive Mode muss Strong drive eingestellt werden.
 9) Bei der PWM wird der PWM Mode auf One Output gestellt.
10) Die Clock und der digital-low werden mit der PWM verbunden und
11) der ausgang wird dann zur LED geführt.
12) In der cydwr Datei müssen noch die Pins vergeben werden.
13) Der Button ist auf dem Pin P3(0) gelegt und
14) die LED wird auf den Pin P2(4) gelegt.
15) Danch muss der Button auf der Platine noch mit einem Kabel zum Pin P3(0) verbunden werden.
16) Danach wird alles mit dem Programm der darunter angegeben ist geregelt.
*/

#include "project.h"
int main(void){
        XH: Der Code wirkt planlos, Kommentare geistlos und erklaeren keine Funktionaliaet
        Variablennamen irrefuehrend (zB.'a'), Tastenentprellung muehselig, insgesamt topfig
    CyGlobalIntEnable; // Aktivieren der globale Interrupts wozu?
    uint8_t a, up; // Dieses Array speichert die Daten, die in einem 128-Bit-Register eines externen Peripheriegeräts.
        da ist kein Array!
    PWM_Start();  // Startet den PWM Baustein
    PWM_Enable(); // PWM wird aktiviert
        was unterscheidet 'starten' und 'aktivieren'?
    a = 0; 'a' wie 'brightness'?        // Variable einen bestimmten Wert zuweisen, a ist die Helligkeit der LED
    PWM_WriteCompare(a);   // a wird in PWM geschickt
    for(;;){
        if ((Button_Read() == 0) && (a==0)){                // Falls Button gedrückt und a = 0
            CyDelay(200);                                   // Delay für 200ns
            if (Button_Read() == 1){                        // Falls Button losgelassen wird,
                a = 255;                                    // wird a der Wert 255 zugewisen (LED auf max. Helligkeit)
                PWM_WriteCompare(a);                        // a in PWM gegeben
            }
            while ((Button_Read() == 0) && (a<255)){        // Während Button gedrückt und a größer 255
                a++;                                        // wächst der a Wert (LED wird heller)
                up = 1;                                     // der Variable up wird der Wert 1 zugewiesen
                PWM_WriteCompare(a);                        // a in PWM gegeben
                CyDelay(20);                                // Delay für 20ns -- iXH bezweifle die 20 Nanosekunden
            }
            CyDelay(100);                                   // Delay für 100ns
        }
        if ((Button_Read() == 0) && (a==255)){              // Falls Button gedrückt und a = 255
            CyDelay(200);                                   // Delay für 200ns
            if (Button_Read() == 1){                        // Falls Button losgelassen wird,
                a = 0;                                      // wird a der Wert 0 zugewisen (LED aus)
                PWM_WriteCompare(a);                        // a in PWM gegeben
            }
            while ((Button_Read() == 0) && (a>0)){          // Während Button losgelassen und a größer 0
                a--;                                        // wird der a Wert weniger (Helligkeit geht runter)
                up = 0;                                     // der Variable up wird der Wert 0 zugewiesen
                PWM_WriteCompare(a);                        // a in PWM gegeben
                CyDelay(20);                                // Delay für 20ns
            }
            CyDelay(100);                                   // Delay für 100ns
        }
        if ((Button_Read() == 0) && 0<a && a<255){          // Falls Button gedrückt und a größer 0 aber kleiner 255
            CyDelay(200);                                   // Delay für 200ns
            if (Button_Read() == 1){                        // Falls Button losgelassen wird,
                a = 0;                                      // wird der Variable a der Wert 0 zugewisen
                PWM_WriteCompare(a);                        // a in PWM gegeben
            }
            if (up == 1){                                   // Falls up = 1 ist
                while ((Button_Read()==0) &&(a>0)){         // Während Button gedrückt und a größer 0,
                    a--;                                    // wird a weniger (Helligkeit geht runter)
                    up = 1;                                 // up wird der Wert 1 gegeben
                    PWM_WriteCompare(a);                    // a in PWM gegeben
                    CyDelay(20);                            // Delay für 20ns
                }
            }
            if (up == 0){                                   // Falls up = 0 ist
                while ((Button_Read()==0) &&(a<255)){       // Während Button gedrückt und a kleiner 255,
                    a++;                                    // wächst a (Es wird heller)
                    up = 0;                                 // up wird der Wert 0 gegeben
                    PWM_WriteCompare(a);                    // a in PWM gegeben
                    CyDelay(20);                            // Delay für 20ns
                }
            }
        }//if
    }//for
}//main
```

```
        XH: Da die gesonderte Tastenentprellung hier wertlos ist,
        schlage iXH folgende Vereinfachung obigen Code-Haufens vor:
    p=0;        //Tastverhaeltnisse nennt man 'p' und nicht 'a'
    up=1;       //beginne mit steigendem (UPwards) Tastverhaeltnis
    for(;;){
      while(Button_Read()==0){
        if(p<1) up=1; else if(p>254) up=0;
        (up)? p++:p--;
        PWM_WriteCompare(p);
          /*von der Fernseh-Technik wissen wir, dass nur Bewegungen unter 25 Bildern/s
            einzeln wahrgenommen werden, also reduzieren wir die Bildfolgerate unter 25/s */
        CyDelay(45);
      }
    }//for
```

12.5.8 Beispiel 'Reihenleuchte' (?)

```
REIHENLEUCHTE: es ist ein ->'Lauflicht'
Erstes Projekt sehr simpel, eine Reihenleuchte.
Ziel ist es die LEDs 0 - 7 der REIHE nach leuchten lassen.
Es soll immer nur eine LED gleichzeitig leuchten(also wenn es von LED 0 zu LED 1 springt
soll sich davor die LED 0 ausschalten).
Wenn es dann die LED 7(die "Letzte" LED) dann soll es "abprallen" und die andere Richtung weiter Leuchten.
Dann sobald es auf dem anderen Ende, der LED 0 ankommt soll es wieder "abprallen".
Dann muss ich beim TopDesign etwas aufbauen.
Das geht ganz schnell weil ich brauche einfach nur 8 digital Output pins.
Die benenne ich dann LED 0 - LED 7 und muss bei den Konfigurationen jeweils
die HW Connection ausschalten und den Digital Output ausschalten.
Dann muss ich die Pins zuweisen, dafür muss ich genau auf den PSOC schauen,
weil die LEDs einzeln beschriftet sind.
Und dann kann ich schon anfangen zu Coden. ->''codieren''

#include <project.h>
void setLED(int LED,int value){ //setLED(schaut welche LED angesprochen ist, entweder 0 oder 1 um die LED ein/aus zuschalten)
    if (LED == 0){              //alle LEDs abfragen und einen Wert zwischen
                                [0;7] zu geben, diese werden dann in der setLED "angeschaut"
        LED0_Write(value);
    }
    else if (LED == 1){
        LED1_Write(value);
    }
    else if (LED == 2){
        LED2_Write(value);
    }
    else if (LED == 3){
        LED3_Write(value);
    }
    else if (LED == 4){
        LED4_Write(value);
    }
    else if (LED == 5){
        LED5_Write(value);
    }
    else if (LED == 6){
        LED6_Write(value);
    }
    else if (LED == 7){
        LED7_Write(value);
    }
    else if (LED == -1){        //für spezial Fall unten
        for (int i = 0;i < 8;i++){
            setLED(i, value);   // 'i' ersetzt LED,
                Oh! Die Function is ja rekursiv!
        }
    }
}
int main(void){
    int step = -1;               // 'step' um hochzuzählen
    for(int i = 0;;i+=step){    // 'i+=step' zählt hoch OR runter falls 'step' negativ
        setLED(-1, 0);          // spezial Fall bei '-1' werden ALLE LEDs auf '0' gesetzt
        setLED(i, 1);           // wenn 'i' gleich 'LED' dann wird auf 1 gesetzt
        if (i == 7||i == 0){    // || = lazy OR
            step *= -1;
        }
        CyDelay(500);
    }
}
```

```
  /* kuerzer: */
void setLED(int LED, int value) {
  switch (LED) {
        case 0: LED1_Write(value); break;
        case 1: LED2_Write(value); break;
        case 2: LED3_Write(value); break;
        case 3: LED4_Write(value); break;
        case 4: LED5_Write(value); break;
        case 5: LED6_Write(value); break;
        case 6: LED7_Write(value); break;
        case 7: LED8_Write(value); break;
        case -1:LED1_Write(0);LED2_Write(0);LED3_Write(0);LED4_Write(0);
                LED5_Write(0);LED6_Write(0);LED7_Write(0);LED8_Write(0);
                break;
  }
}
int main(void){
    int ctr=0,updn = 1;         //counter + Zaehlrichtung
    for(ctr=0;;ctr+=updn){      //count up(1) || down(-1)
        setLED(-1, 0);          //alle LED loeschen
        setLED(ctr,1);
        if (ctr == 7 || ctr == 0){
            updn= -updn;
        }
        CyDelay(500);
    }
}
```

```
  /* noch kuerzer, aber 'doof': */
#define delay 500
int main(void){
  for(;;){
        LED1_Write(1);CyDelay(delay);LED1_Write(0);
        LED2_Write(1);CyDelay(delay);LED2_Write(0);
        LED3_Write(1);CyDelay(delay);LED3_Write(0);
        LED4_Write(1);CyDelay(delay);LED4_Write(0);
        LED5_Write(1);CyDelay(delay);LED5_Write(0);
        LED6_Write(1);CyDelay(delay);LED6_Write(0);
        LED7_Write(1);CyDelay(delay);LED7_Write(0);
        LED8_Write(1);CyDelay(delay);LED8_Write(0);
  }
}
```

12.5.9 Beispiel 'Würfel'

```
Brauch:
        1xDigital Input Pin (für Taster)
        7xDigital Output pins (für die LEDs)
Beim Button:
        Digital input: enabled
        HW Connection: disabled
        Digital output: disabled
Bei den LEDs:
        Digital input: disabled
        Digital Output: enabled
        HW Connection: disabled
Beim Psoc selber muss ich den 'Würfel' anschließen.
Da muss ich schauen welche LED bei welchem Pin ist:
LED 0: P2.7
LED 1: P1.2
LED 2: P1.3
LED 3: P1.4
LED 4: P1.5
LED 5: P1.6
LED 6: P1.7
Taster: P2.6
        Der Pin für die LED 2[P1.3] ist bereits von dem PSOC belegt,
        d.h ich weiche auf einer der anderen LEDs um! Nämlich dem P12.3 Pin!
```

12.5.10 Beispiel 'UART-Receive-Only-Bsp2'

```
/* ========================================
 * Autor:Rocky Balboa
 * Datum:23.05.2022
 * Titel:UART_Receive_Only_Beispie2
 * ========================================
*/
#include <project.h>
```

```
int main(){
 uint8 ch;                    //received character
 CyGlobalIntEnable;           /* Enable global interrupts. */
        /* Place your initialization/startup code here (e.g. MyInst_Start()) */
 UART_1_Start();
 for(;;){                       /* Place your application code here. */
   ch = UART_1_GetChar();
    if (ch != 0u) {             //Character received
     Control_Reg_1_Write(ch);   //display ASCII code at LED7...0
   }
 }
}

//Bei diesem Übungsbeispiel werden 8bit Werte (Bytes) seriell empfangen und parallel
//an den LED0 ... LED7 ausgegeben. /*
1) Es ist ein neues PSoC Creator Projekt anzulegen;
2) benötigt wird die UART-Einheit, ein Control-Register sowie digitale I/O-Pins
3) Der TX-Pin wird permanent mit log. 1 verbunden; damit wird verhindert,
   dass das PCseitige Terminalprogramm irgendwelche Störungen als empfangene Daten interpretiert.
   (XH: aso?
        RS232/V.24 kennt ka 'log.1' sondern
        MARK und SPACE, also 'aktiv +12V' und 'Ruhe -12V'.
        Weder MARK noch SPACE unterdruecken 'irgendwelche' Stoerungen!)
4) Im Polling-Verfahren wird nach neuen, empfangenen Daten geprüft.
   Dies erfolgt durch Aufruf der Funktion
        UART_1_GetChar()
   Wurde kein neues Zeichen empfangen, so ist der Rückgabewert 0x00 - ansonsten wird
   das empfangene Zeichen zurückgegeben
   (XH: auch '0x00' ist ein gueltiger Datenwert und
        beginnt mit einem 'Startbit'
        '0x00' != 'Nichts' !!! )
5) Das Projekt ist zu kompilieren und auf das PSoC 5LP Edu Board zu übertragen.
6) Auf der PC Seite ist das Terminalprogramm zu öffenen
   und die Verbindung mit dem PSoC 5LP Edu Board herzustellen.
7) Wird nun ein Zeichen gesendet,
   so wird das Datum an den LEDs 7...0 ausgegeben (i.d. Regel der ASCII Code).
*/
```

12.5.11 Beispiel 'UART-Transmit-Only-Bsp1'

```
/* ========================================
 * Autor:Ruebezahl
 * Datum:23.05.2022
 * Titel: UART_Transmitt_Only_Beispiel1
 * ========================================
*/
#include <project.h>
int main(){
 uint8 i=0x21;          // wir beginnen mit dem Zeichen !
 CyGlobalIntEnable;     /* Enable global interrupts. */
                        /* Place your initialization/startup code here (e.g. MyInst_Start()) */
 UART_1_Start();
  for(;;){              /* Place your application code here. */
   UART_1_PutChar(i);   // Ausgabe von i via UART
   i++;
   if (i > 0x7D) {      // nach Ausgabe von } ist wieder mit ! zu beginnen
     i='!';             // wir können auch den Compiler das ! in 0x21
                        // umrechnen lassen
   }
   CyDelay(250);          // kurze Pause bevor nächstes Zeich übertragen wird
 }
}

//Im Schematic wurde ein UART_1 Baustein verwendet
//Das Projekt kann nun compiliert und übertragen werden.
//Auf dem PC ist ein serielles Terminalprogramm zu starten (HyperTerm, TeraTerm, Putty, ...
//oder z.B. HTerm).
  Der COM-Port der USB-UART Brücke ist auszuwählen und vor dem
//ersten Connect sollte geprüft werden, ob die restlichen Schnittstellenparameter korrekt
//eingestellt sind; diese müssen mit der Konfiguration der UART-Einheit am PsoC
//übereinstimmen - d.h.
        Baudrate: 115200
        8 Datenbits, 1 Stopp-Bit, keine Parität
//Danach sollte einer erfolgreichen Verbindung nichts mehr im Wege stehen... hat funktioniert!!
```

12.5.12 Konfus: Beispiel 'USB-UART-Bridge' +main.c

```
aus https://blog.artwolf.in/a?ID=b820cfa8-c118-4d00-8b2d-f9d932779725

In this article, it shows the procedure to use the USB-UART-Brige
  in PSoC 5LP Prototyping Kit (CY8CKIT-059 ).
  This procedure is picked up the contents of which are described in the "KitProg User Guide" of Cypress.
  "Note that use the USB-UART-Bridge in PSoC4 is PSoC 5LP version of".
Things necessary
  CY8CKIT-059
  PSoC Creator
Try to make the sample program
  Creating a project
  Create an empty project, puts the UART component.
UART settings
  Double-click the UART component, and set the communication speed.
Build
  Once, to build the project, to generate the necessary source.
Pin settings
  Open the project name .cydwr, and set the pin assignment. Leave the set as shown in FIG. (Important).
  Name  Port    Pin
  RX_1  P12[6]  20
  TX_1  P12[7]  21
main.c sample program of
   For example, you try to write a program such as the following.
int main(){
    CyGlobalIntEnable;/* Enable global interrupts. */
    UART_1_Start();
    UART_1_PutString("USB-UART");
    UART_1_PutCRLF(1);
    for(;;){
        uint8 ch = UART_1_GetChar();
        switch(ch){
        case 0:
           /* Nothing */
            break;
        case '?':
            UART_1_PutString("Sample Program. ");
            UART_1_PutCRLF(1);
            break;
        case '!':
            CySoftwareReset();
            break;
        }
    }
}

Build & writing
  Build and writing to the board.
To connect with terminal software
  TeraTerm try to connect in a terminal software such as.
  To connect to the serial port. Select the port that is displayed as "KitProg".
  And the setting of the communication speed.
Execution
  You can see the execution. "!" Reset of the board is performed in the key,
  it will display "USB-UART". "?" Key "Sample Program." And it will be output.
reference
  It indicates a well-use likely to function in the UART component. XXX is the component name.

Function Name                                   Description
---------------------------------------------------------------
void XXX_Start(void)                            to start the communication of the UART
uint8 XXX_GetChar(void)                         will receive a single character
uint16 XXX_GetByte(void)                        will receive a 1 byte
void XXX_PutChar(uint8 txDataByte)              to send a single character
void XXX_PutString(const char8 string[])        to send a string
void XXX_PutCRLF(uint8 txDataByte)              will send a newline character
void XXX_Sleep(void)                            and in sleep mode
void XXX_Wakeup(void)                           will return from sleep mode
```

12.5.13 Beispiel 'ADC': ADCmuijnz.c

```
/***************
 *Title:        ADC
 *Author:       Miezekatze
 *Date:         28.04.2028
 *Description:  AD-Messung ohne internen Analog-Digital-Wander
 *Dokumentation:Mein Code ist ja selbsterklaerend!
 ****************************************************************/
float tau = 0.001;    //Tau=R4*C2=100*10*10**(-6)=0.001 Idiot!
float C2 = 0.00001;   //10uF Kondensator
```

```
float dzeit, zeit1, zeit2;
float widerstand;
int zaehler1 = 0;
int zaehler2 = 0;
char cbuf [32];

/* Entladung C2 wird über R4 entladen */
void Entladung(void){
    P1_0_SetDriveMode(PIN_DM_DIG_HIZ);
    P3_1_SetDriveMode(PIN_DM_DIG_HIZ);
    P3_2_SetDriveMode(PIN_DM_DIG_HIZ);
    P3_3_SetDriveMode(PIN_DM_STRONG);
    P3_3_Write(0);
    CyDelay(5*tau); //Mindestzeit für das Entladen
}

/* Messphase 1 (nur R3) C2 wird NUR über R3 aufgeladen */
void Ladephase1(void){
    P1_0_SetDriveMode(PIN_DM_DIG_HIZ);
    P3_1_SetDriveMode(PIN_DM_DIG_HIZ);
    P3_2_SetDriveMode(PIN_DM_STRONG);
    P3_3_SetDriveMode(PIN_DM_DIG_HIZ);
    Timer_Start();
    P3_2_Write(1);
    while(P1_0_Read() == 0){
        if (uebertrag_Read() == 1){
            zaehler1++;
        }
    }
    zeit1=zaehler1*0.001;
    P3_2_Write(0);
}

/*Messphase 2 (R3+TS1) C2 wird über R3 + TS1 (Temperaturwiderstand) aufgeladen*/
void Ladephase2(void){
        P1_0_SetDriveMode(PIN_DM_DIG_HIZ);
        P3_1_SetDriveMode(PIN_DM_STRONG);
        P3_2_SetDriveMode(PIN_DM_DIG_HIZ);
        P3_3_SetDriveMode(PIN_DM_DIG_HIZ);
    Timer_Start();
    P3_1_Write(1);
    while(P1_0_Read() == 0){
        if (uebertrag_Read() == 1){
            zaehler2++;
        }
    }
    zeit2=zaehler2*0.001;
    P3_1_Write(0);
}

/*Endzustand --> alle Pins werden auf High-Z geschaltet*/
void Endzustand(void){
        P1_0_SetDriveMode(PIN_DM_DIG_HIZ);
        P3_1_SetDriveMode(PIN_DM_DIG_HIZ);
        P3_2_SetDriveMode(PIN_DM_DIG_HIZ);
        P3_3_SetDriveMode(PIN_DM_DIG_HIZ);
}
/*Widerstandsberechnung
 *aus den ermittelten Zeiten kann der Widerstand berechnet werden*/
void Berechnung(void){
    widerstand = (zeit2-zeit1) / (5*C2);
    sprintf(cbuf, "Widerstand betraegt: %fOhm", widerstand);
    UART_PutString(cbuf);
}

int main(void){
    CyGlobalIntEnable; /*Enable global interrupts.*/
    UART_Start();
    for(;;){
        Entladung();
        Ladephase1();   //Messphase 1 (R3)
        Entladung();
        Ladephase2();   //Messphase 2 (R3 + TS1)
        Berechnung();   //Widerstandsberechnung
        Entladung();
        Endzustand();   //Endzustand --> alle auf High-Impedance
        CyDelay(5000);  //Kurzer Delay, dass im Terminal nicht die ganze Zeit ausgegeben wird
    }
}
```

12.6 PSOC5LP miese Dokumentation

Die Dokumentation der PSOC-Lösungen zerfällt in die Kategorien

a) völlig undokumentiert
b) unverständlich, nicht nachvollziehbar*)
c) lückenhaft, nicht nachvollziehbar*)
d) brauchbar

*) 'nachvollziehbar' heisst, jede Person kann es nachmachen, ohne spezifische Detail-Vorkenntnisse; Lücken oder falsche Bezeichnungen verhindern das

Deshalb hier eine kleine Nachhilfe zur Planung:

Technische Dokumente sind je nach Entstehung
Plan ↔vorher
Protokoll↔gleichzeitig
Bericht ↔danach

Die Aussage "ich fertige zuerst das Produkt – wenn es funktioniert, erstelle ich die Planung"

⇒ heißt auch
⇒ "ich kann nicht planen"
⇒ "ich gehöre in die Fachschule und nicht in die Höhere"
⇒ "ich bin Entwicklungs-unfähig"
weil man dazu das Ziel vorher kennen muss

⇒ "ich kann kein einer Kundschaft weder Preis noch Lieferzeit nennen"
weil ich es nicht planen kann

⇒ "ich kann kein Team zusammenstellen"
weil ich nicht weiss, was die können sollen

⇒ "ich tauge nicht für Management +Organisaion"
weil man dazu wissen müsste, was man wann von wem braucht

⇒ "ich kann keine Karriere machen"
weil man dabei planen und einteilen müsste

Das Gegenargument "ich muss erst schauen, obs geht"
⇒ ist nicht vertretbar:
→ dann ist eben eine Vorstudie vorzuschalten

12.6.1 Doku 'communication.h'

```
 * TX is a Register that is directly linked to the TX pins
 * LEDs is a Register that is directly linked to the LEDs
 * SOCCOM_DEBUG is a define that enables the debug output
 * SOCCOM_1_DELAY is a define that sets the delay between the 2 flips of a 1-Bit
 * SOCCOM_BIT_DELAY is a define that sets the overall delay between 2 bits
 *
 * Author: Chesley Sullenberger
 * v1: 22-05-30
*/
#pragma once
// include the generated header file to access TX and LEDs
#include "project.h"
```

12.6.2 Doku 'main.c'

```
/* The main (debug) code for the communication
 * Author: Schwarzenegger */
// enables the debug output
#define SOCCOM_DEBUG 1
// include the auto-generated header file
#include "project.h"
// include the communication-library
#include "communication.h"
// include the standard-library
#include <stdio.h>
// define a debug string
char text[] = "Lorem Ipsum dolor sit amet.";
// the entry point of the program
int main(void) {
```

12.6.3 Doku 'UART1.c'

```
//Im Schematic wurde ein UART_1 Baustein verwendet
//Das Projekt kann nun compiliert und übertragen werden.
//Auf dem PC ist ein serielles Terminalprogramm zu starten (HyperTerm, TeraTerm, Putty, ...
//oder z.B. HTerm).
  Der COM-Port der USB-UART Brücke ist auszuwählen und vor dem
//ersten Connect sollte geprüft werden, ob die restlichen Schnittstellenparameter korrekt
//eingestellt sind; diese müssen mit der Konfiguration der UART-Einheit am PsoC
//übereinstimmen - d.h.
        Baudrate: 115200
        8 Datenbits, 1 Stopp-Bit, keine Parität
//Danach sollte einer erfolgreichen Verbindung nichts mehr im Wege stehen... hat funktioniert!!
```

12.6.4 Doku 'CypressDemo.c'

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include "project.h"
#include "stdlib.h"
```

12.6.5 Doku '.c'

```

```

12.6.6 Doku 'SAR-ADC.c'

```
/*--------------------------------------*
| Author: Neil Armstrong               |
| Datum: 23.05.2022                    |
| ADC ist SAR ADC                      |
| LEDs ist ein ControlRegister,        |
| welches die LEDs ansteuert           |
*--------------------------------------*/
```

12.6.7 Doku '????'

```
/* ========================================
 * LED up/down "Trimmer"
 *
 * im TopDesign:
 *  2 Buttons Definieren (wichtig Drivemode: Resistive pull up/down und Hardware Connection ausschalten).
 *  8 LEDs an ein ControlRegister (Name "LED") hängen.
 * Alle Pins im Pins Tab der Hardware entsprechend definieren.
 *
 * Mit FileZilla hochgeladen
 * ========================================
*/
```

12.6.8 Doku 'neuse projekt'

```
/* neuse Projekt erstellen
    8 DIgital outputs in die schematic ziehen
    Outputs den Pins zuweisen
    Programmcode schreiben
    Psoc anstecken und programier
    Fertig */
```

12.6.9 Beispiel 'Lauflicht'

```
/* ========================================
Lauflicht 1
Winnetou, 4ahel
 * ========================================
*/
/*
1. PsoC Creator 4.4 öffnen
2. Links oben auf "File" danach auf "New" und einen neuen Projekt anlegen.
3. PsoC anschließen
Danach muss im Worspace Explorer das Fenster "TopDesign.cysch" geöffnet werden.
Danach muss rechts am Bildschirm beim "Component Catalog" unter Ports und Pins
der Digital Output Pin [v2.20] auf die Fläche hineingezogen werden.
(In unserem Fall muss es sieben Output Pins gegeben sein.)
Diese werden dann als LED_1, LED_2, LED_3 .... umbenannt.
Später muss im Workspace Explorer der Unterpunkt Pins geöffnet werden und
anschließend muss man dem jeweiligen virtuellen Pin ein realen Pin zuweisen.
```

12.6.10 Doku 'doppelklicken'

```
vom 'FALKE' wurde mir einmal als Gebrauchsanweisung das einzige Wort
      doppelklicken
abgegeben. (wobei das Doppelklicken gar nicht funktioniert hat)
```

12.6.11 Doku 'x'

```

```

12.6.12 Doku 'x'

```

```

12.7 PSOC5LP Bspe Dokumentation

12.7.1 Beispiel 'Gebrauchsanweisung'

```
/* ========================================================================
 * PROGRAM DESCRIPTION:
 * When initiated by tying P0[7] LOW, the program is shortly (9ms) supplying
 * a constant current of exactly 9mA (or constant CURRENT8U) to its analogue
 * output on P3[0] to which the anode of an an LED shall be connected before-
 * hand. (Max. voltage 5V)
 * After said settling time of 1ms, the program measures the needed voltage
 * to feed 1mA with a resolution of 12b and outputs the measured voltage in
 * uV via UART serial connection to the terminal. This voltage is the thresh-
 * old voltage

 * BUTTONS: The button inputs are tied high by a Pull-Up-Resistor
 * by the PSoCs internals. Pressing a button grounds the input. A
 * pressed button ties the input to logical low.
 * PIN CONFIGURATION: The pin selection can be viewed and changed
 * with the Pin-Configuration in the Workspace. Standard pin con-
 * figuration:
 *              button_1                -   P0[7]
 *              button_2                -   P0[6]
 *              button_3                -   P0[5]
 *              OUTPUT VDAC (to res)    -   P3[2]
 *              INPUT   ADC (from led)  -   P3[0]
 *
 * Buttons need to be wired up, too! They are not hard-wired by
 * the board. Pressed buttons always short to ground! Thus inter-
 * nal Pull-Ups are used.

 * M A N U A L:
 * Push button 1 to increase the output voltage of the DAC, keep
 * it pushed to incrementally increase the voltage further. Push
 * button 2 to decrease the DAC voltage.
 * Use button 3 to get the LED voltage and print it via Serial
 * to your terminal emulator.
 * ========================================================================
 */
```

12.7.2 Beispiel 'COM console'

```
/* ============================================================
 * G E N E R A L     I N F O
 * CONNECTING VIA SERIAL: Use Putty or another Terminal Emulator
 * to connect via serial to the KitProg's COM-Port at 9600Bd.
 * You can get the port-number in Device-Manager (Geraete-Manager)
 * at "Ports" (COM and LPT) ("Anschluesse" (COM und LPT).
 * MANUAL:
 *  1. Connect your PSoC 5LP via the KitProg-Programmer (Big USB A on the left)
 *  2. Flash the program onto your PSoC using CTRL+F5 in PSoC Creator
 *  3. Open 'Device Manager' (Geraetemanager) on your PC
 *  4. Search for category 'Ports' (COM & LPT) (Anschluesse (COM & LPT))
 *  5. Find the COM-number of KitProg USB-UART
 *  6. Disconnect your Device as to not accidentally short something when using...
 *  7. ...M2M-Jumper-Wires to connect:
 *  7.1  P0[7] to 'button 1'
 *  7.2  P3[0] to the anode (+) of the LED that gets tested
 *  7.3  GND or your PSoC to GND of your LED-Board
 *  8. Reconnect your Device
 *  9. Open 'PuTTY' or another terminal emulator on your PC
 * 10. Establish a serial connection via the KitProg's COM-Port (see 5.)
 * 10.1  9600 BAUD, 8 Data Bits, 1 Stop Bit, No Parity, XON/XOFF;
 * 11. Use 'button 1' to trigger a measurement
 * 13. You should now see a serial output in uV as well as a binary 8b-output via
 *     the developtment board's LEDs which show 255 divisions of 5V. Change the
 *     jumper at P3[0] to the Anode of the next LED and repeat your measurement by
 *     pressing 'button 1' again.
 */
```

12.7.3 Beispiel 'Anschlusstabelle'

```
/*  NAME    |   TYPE    |   PIN
    --------+-----------+--------
    BUT     |   input   |   15[0]
    LED_1   |   output  |   2[7]
    LED_2   |   output  |   1[7]
```

```
    LED_3   |   output  |   12[2]
    LED_4   |   output  |   1[4]
    LED_5   |   output  |   1[5]
    LED_6   |   output  |   1[6]
*/
```

12.7.4 Beispiel 'tuerkisch deutsch'

```
/* Beschreibung zum Aufbau:
 1) Zuerst wird Psoc_Creator gestartet.
 2) Dann wird ein neues Workspace unter dem Namen LED_Potentiometer erstellt.
 3) Im TopDesgin wird
        ein Digital Input,
        ein DigitalOuput,
        ein Analog-Digital_Converter,
        ein Puls-Weiten-Modulator,
        eine 1,2 MHz Clock und
        ein Digital_low plaziert.
 4) Den DigitalInput nenne ich Potentiometer.
 5) Man muss Drive mode Resistive pull up/down einstellen.
 6) Den Digital Output nenne ich LED.
 7) Bei dem wird als Drive Mode Strong drive eingestellt werden.
 8) Beim ADC wird nichts eingestellt und
 9) bei der PWM wird die PWM Mode auf One Output eingestellt.
     Ohne Output wird aber nit viel herauskommen!
10) Der Poti und die Colck werden mit dem ADC verbunden.
11) Die Clock und der digital-low werden in die PWM geleitet.
12) Von der PWM wird dann in die LED gefahren.
13) In der cydwr Datei müssen noch die Pins vergeben werden.
14) Der Poti ist auf dem Pin P3(0) gelegt und
15) die LED wird auf den Pin P12(2) gelegt.
16) Danch muss der Potentiometer auf der Platine noch mit einem Kabel zum Pin P3(0) verbunden werden und
17) der VCC des Potis wird zum 3,3V Quelle verbunden und
18) der GND wird zu einem aktiven GND verbunden.
19) Danach wird alles mit dem Programm der darunter angegeben ist geregelt.
*/
-> wie kann ein Potentiometer ein DIGITAL input sein ??
-> Das Deutsch ist hier leider sehr sehr mangelhaft, sinnstoerend und untolerierbar.
   Von 'musterguеltig' kann keine Rede sein.
```

12.7.5 Beispiel 'mit Properties'

```
Brauch:
        1xDigital Input Pin (für Taster)
        7xDigital Output pins (für die LEDs)
Beim Button:
        Digital input: enabled
        HW Connection: disabled
        Digital output: disabled
Bei den LEDs:
        Digital input: disabled
        Digital Output: enabled
        HW Connection: disabled
Beim Psoc selber muss ich den 'Würfel' anschließen.
```

12.7.6 Beispiel 'receive only'

```
//Bei diesem Übungsbeispiel werden 8bit Werte (Bytes) seriell empfangen und parallel
//an den LED0 ... LED7 ausgegeben. /*
1) Es ist ein neues PSoC Creator Projekt anzulegen;
2) benötigt wird die UART-Einheit, ein Control-Register sowie digitale I/O-Pins
   deren Einstellungen(Properties) ich hier nicht angebe,
   weil mir die Brauchbarkeit der Doku scheissegal ist.
3) Der TX-Pin wird permanent mit log. 1 verbunden. Der RS232-Ruhepegel 'Space'
   ist zwar -12V, die invertierenden RS232 line-driver-ICs fehlen dem PSOC aber,
   sodass wir ein logisches '1' als Ruhesignal auszugeben haben.
4) Im Polling-Verfahren wird nach neuen, empfangenen Daten geprüft.
   Dies erfolgt durch Aufruf der Funktion
        UART_1_GetChar()
   Wurde kein neues Zeichen empfangen, so ist der Rückgabewert 0x00 - ansonsten wird
   das empfangene Zeichen zurückgegeben, dh. eine echt empfangene ASCII-Null
   mit Start-, Stop- und Paritybit
   ist nicht von 'Nichts' unterscheidbar?
5) Das Projekt ist zu kompilieren und auf das PSoC 5LP Edu Board zu übertragen.
6) Auf der PC Seite ist das Terminalprogramm zu öffenen
```

```
   und die Verbindung mit dem PSoC 5LP Edu Board herzustellen.
   Bei Verbindung ueber RS232-Leitungen waere bitte zu beachten, dass diese
   +12V/-12V Spannungen fuehren, womit sie die 5V/3.3V-Logik-ICs zerstoeren!!
   (kaum ein PC hat noch eine RS232/V.24 Schnittstelle)
   Bei Usb-emulierter 'virtual COM port' Verbindung ist das nicht der Fall.
   Von welchem der Autor hier spricht, laesst er leider unklar.
7) Wird nun ein Zeichen gesendet,
   so wird das Datum (=Einzahl von 'Daten')
   an den LEDs 7...0 als Binaerzahl angezeigt (der ASCII Code).
*/
```

12.8 PSOC5LP

Aus dem Inhaltsverzeichnis des 'reference manual'
(→ RefMan.pdf)
erschließen sich die Features

- Section B: CPU System
  - ⊕ CortexTM-M3 Microcontroller
  - ⊕ PSoC 5LP Cache Controller
  - ⊕ PHUB and DMAC
  - ⊕ Interrupt Controller
- Section C:Memory
  - ⊕ Nonvolatile Latch
  - ⊕ SRAM
  - ⊕ Flash Program Memory
  - ⊕ EEPROM
  - ⊕ EMIF
  - ⊕ Memory Map
- Section D:System Wide Resources
  - ⊕ Clocking System
  - ⊕ Power Supply and Monitoring
  - ⊕ Low-Power Modes
  - ⊕ Watchdog Timer
  - ⊕ Reset
  - ⊕ I/O System
  - ⊕ Flash, Configuration Protection
- Section E:Digital System
  - ⊕ Universal Digital Blocks (UDBs)
  - ⊕ UDB Array and Digital System Interconnect
  - ⊕ Controller Area Network (CAN)
  - ⊕ USB
  - ⊕ Timer, Counter, and PWM
  - ⊕ I2C
  - ⊕ Digital Filter Block (DFB)
- Section F:Analog System
  - ⊕ Switched Capacitor/Continuous Time
  - ⊕ Analog Routing
  - ⊕ Comparators
  - ⊕ Opamp
  - ⊕ LCD Direct Drive
  - ⊕ CapSense
  - ⊕ Temperature Sensor
  - ⊕ Digital-to-Analog Converter
  - ⊕ Precision Reference
  - ⊕ Delta Sigma Converter
  - ⊕ Successive Approximation Register ADC
- Section G: Program and Debug
  - ⊕ Test Controller
  - ⊕ Cortex-M3 Debug and Trace
  - ⊕ Nonvolatile Memory Programming

Teil IV
VHDL

13 VHDL VHSIC hardware definition language

VHSIC = *very high speed integrated circuit*
(wurde mehrfach geändert)

13.1 VHDL Einführung

13.1.1 VHDL Grundgerüst

VHDL - "Very High Speed Integrated Circuit Hardware Description Language"
(die Deutung änderte sich mehrfach)

- HDL (hardware description language HDL) ist kompakter als Schaltpläne
- ist leichter kopierbar (StrgC + StrgV) (als Schaltpläne)
- lässt sich automatisch überprüfen
- lässt sich simulieren
- lässt sich ”synthetisieren” (in eine FPGA-Innen-Schaltung umwandeln)
- ist vorteilhaft bei Prozessordesign
- VHDL steht in Konkurrenz zu "Verilog"
- wir verwenden das Produkt ’Quartus II’ (Nachfolger ’Quartus Prime’)
  und das ’DE0’ Development Board

```
-- (this is a VHDL comment)
/*
    this is a block comment (VHDL-2008)
*/
-- VHDL is case-insensitive!

library IEEE;
use IEEE.std_logic_1164.all;

entity SchaltungsName is
  port (
        ...
  );
end entity SchaltungsName;

architecture Verarbeitung of SchaltungsName is
begin
        ...
end architecture Verarbeitung;
```

das Grundgerüst: Du hast
a) einen Header (so was wie diese ’#include’ in ’C’) → das schreiben wir einfach jedes Mal so ab

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

b) eine Modul-Beschreibung (’Entity’) mit Ein- und Ausgängen:

```
ENTITY <Modulname> IS PORT(
  ...
);
END <Modulname>;
```

und
c) die Schaltungsbeschreibung ('Architecture'):

```
ARCHITECTURE <Schaltungsname> OF <Modulname> IS
BEGIN
  ...
END <Schaltungsname>;
```

→
das Ganze schaut dann so aus:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY <Modulname> IS PORT(
  ...
);
END <Modulname>;

ARCHITECTURE <Schaltungsname> OF Modulname IS
BEGIN
  ...
END <Schaltungsname>;
```

13.1.2 simples Beispiel

```
library IEEE;
use IEEE.std_logic_1164.all;

entity SchaltungsName is
  port (
    Eingang : in   std_logic;
    Ausgang : out  std_logic;
    BusOut  : out  std_logic_vector(7 downto 0); --'vector'==Leitungs-Buendel
    BusIn   : out  std_logic_vector(7 downto 0);
  );
end entity SchaltungsName;

architecture Verarbeitung of SchaltunngsName is
begin
  Ausgang <= not Eingang;
  BusOut  <= not BusIn when(Eingang) --invertieren
             else BusIn;             --kopieren
end architecture Verarbeitung;
```

13.1.3 Fachsprache

wie immer ist händische Mitschrift zu führen

- **Synthese** ::= zusammensetzen, *"erzeugen"*
- *kann* ***synthetisiert*** *werden* ::= lässt sich in eine Schaltung umsetzen
- **Analyse** ::= zerlegen, *"schauen, was drin ist"*
- **Simulation** ::= gleichwertige Ergebnisse liefernde Software;
  Simulatoren *ignorieren das Zeitverhalten*, sind meist viel langsamer, aber vorteilhaft für Prognose und Untersuchungen

- "nur für Simulation" ::= lässt sich **nicht!** als Schaltung realisieren
- **Emulator** ::= nachahmende Ersatzhardware,
  Emulatoren zeigen auch das *originale Zeitverhalten*, sind daher **ersatzweise** nutzbar
- VHDL wird **nicht!** von einem Computer ausgeführt!
  → nicht alle Programmiersprachen erzeugen ausführbaren Computer-Code.
  - HTML erzeugt → Webseiten
  - LaTeX erzeugt → PDF-Dokumente (damit schreibi inser U-PDF)
  - SPICE erzeugt → Simulationsdiagramme
  - VHDL erzeugt **Hardware**
    daraus werden dann ICs gefertigt oder FPGAs programmiert (ge'flash't)
    'flashen' ist nicht englisch, es stammt von der Speicher-Technologie 'Flash-EEPROM'
    die Elektronikabteilung hat eine handvoll FPGA-EntwicklungsModule, wo man mit dem Utility 'Quartus' VHDL-Programme hineinflashen und testen kann

  → es gibt zwar (was iXH Enk aus Verwirrungsgefahr no nit zeigen will) auch Variable und Schleifen in VHDL, die führen aber nicht dazu, dass etwas mehrfach ausgeführt, sondern dieselbe Hardware mehrfach erzeugt wird.
- "nur für Simulation" ::= lässt sich *nicht* als Schaltung realisieren
- "signal" ::= erzeugt schaltungsinterne Verbindungsleitungen

Aufgabe: Schreib in Deine Handmitschrift in VHDL

a) ein UND-Gatter
b) ein NOR-Gatter
c) einen Inverter
d) eine Prozessor-ALU, die '+', '-', +1, -1, AND und OR kann

und maile mirXH dann das Foto der Mitschrift

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY dingsda IS PORT(
  A1,A2,
  A3    : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
  B     : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
  sel0,
  sel1  : IN  STD_LOGIC;
);
END dingsda;

ARCHITECTURE Funktion OF dingsda IS
  SIGNAL vonAnachB: STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
  vonAnachB <= A2 when (sel0 = '1') else A1;
  B         <= A3 when (sel1 = '1') else vonAnachB;
END Funktion;
```

dieses "A2 when(...) else A1;"
erzeugt einen **"Selector"** = **"Multiplexer"**,
der - abhängig von der Bedingung im "when(...)" - entweder das Bündel A2 oder A1 auswählt ('selektiert')

* Aufgabe: Schreib einen 2:1 Multiplexer mit 8-Bit-Inputs A und B, 8-Bit-Output Y und einem Auswahl-Eingang "s", der bei s=0 das A und bei s=1 das B an Y ausgibt
* Abgabetermin: Stundenende

13.1.4 VHDL signal

wie immer ist händische Mitschrift zu führen

"signal" ::= erzeugt schaltungsinterne Verbindungsleitungen

```
ARCHITECTURE Funktion OF dingsda IS
  SIGNAL   vonAnachB : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
  ...
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY dingsda IS PORT(
  A1,A2 : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
  A3    : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
  B     : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
  sel0  : IN  STD_LOGIC;
```

```
  sel1  : IN  STD_LOGIC;
);
END dingsda;
```

Initialisierung ≡ Anfangswert:

```
ARCHITECTURE Funktion OF dingsda IS
  SIGNAL   vonAnachB : STD_LOGIC_VECTOR(7 DOWNTO 0) := (others => '0');
                                                     --Initialisierung alle '0'
BEGIN
  vonAnachB <= A2 when (sel0 = '1') else A1;
  B         <= A3 when (sel1 = '1') else vonAnachB;
END Funktion;
```

Wenn 'signal' als Bitspeicher (Flipflop) verwendet wird, ist die Initialisierung oft erforderlich

13.1.5 Selector-Multiplexer-Decoder

wie immer ist händische Mitschrift zu führen

→ gib die Funktion der Schaltung als Wahrheitstabelle an
(da stehen dann beim 'Y' a, b und c statt 0 und 1 drin)
vgl. mit Baustein '74HC138'

```
begin
  -- r, s, t   ... Auwahl-Inputs (std_logic)
  -- a, b, c   ... Daten -Inputs (std_logic)
  rst <= (r s t);
  with rst select
    Y <=
        a when ''001'',
        b when ''101'',
        c when ''110'';
end;
```

13.1.6 if case when with

wie immer ist händische Mitschrift zu führen

Schreib in Deine Handmitschrift:
- Ist ein Prozessor eine FSM (Schrittschaltwerk)? (Argumente auflisten)
- Ist ein Prozessor ein *synchrones* oder ein *asynchrones* 'Zeug' ?
- Wieviele Zustände hätte ein Prozessor ?
- Wieviele Zustände hätte ein Computer ?
- schlag (zB. im Internet) die VHDL-Operatoren "srl", "sra" und "ror" nach - was tun sie?
- schreib ein Schaltnetz, welches aus den
  8-Bit-Inputs A und B
  ↓
  die Outputs
  'LT' .... bei A < B
  'GT' .... bei A > B
  'EQ' .... bei A==B
  auf '1' schaltet, in VHDL

und maile mirXH dann das Foto der Mitschrift mit Lösungen

Fallunterscheidungen

with/select:

```
with <auswahl-vector> select
ergebnis <= <Verkn&uuml;pfung_1> when <auswahlwert_1> ,
            <Verkn&uuml;pfung_2> when <auswahlwert_2> ,
            ...
            <Verkn&uuml;pfung_n> when others;
```

when/else:

```
<ergebnis> <=  <Verkn"upfung_1> when <Bedingung_1>
           else <Verkn"upfung_2> when <Bedingung_2>
                ...
           else <Verkn"upfung_n>;
```

if/then: only in sequential environment! (lernen wir erst)

```
if    <Bedingung_1> then <Sequentielle Anweisungen 1>;
elsif <Bedingung_2> then <Sequentielle Anweisungen 2>;
elsif <Bedingung_3> then <Sequentielle Anweisungen 3>;
      ...
else  <Sequentielle Anweisungen n>;
endif;
```

case/is: only in sequential environment!

```
case <testsignal> is
     when <Wert_1> => <Sequentielle Anweisungen 1>;
     when <Wert_2> => <Sequentielle Anweisungen 2>;
          ...
     when others => <Sequentielle Anweisungen n>;
end case;
```

13.1.7 'Elevator' Steuerg. (6)

```
architecture ...
begin
  schliessen <= '1'
        when  timer1   ='1'
          AND timer2   ='0'
          AND zustand  ="stop"
          AND LBarrier ='1'
          AND openButt ='0'
        else '0';
  LangsamfahrtAuf <= '1'
        when  TuerZu   ='1'
          AND timer2   ='1'
          AND Richtung ="steigen"
        else '0';
end;
```

13.1.8 in, out

wie immer ist händische Mitschrift zu führen

Ein - und Ausgänge:
Es gibt allerlei 'Datentypen', wir verwenden vorerst aber nur 'STD_LOGIC'.
einzelne Ein- und Ausgangsleitungen erzeugt man mit zB.:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
  x   : IN   STD_LOGIC;
  y   : OUT  STD_LOGIC;
);
END reg8;
```

ganze Leitungs-Bündel erzeugt man mit 'vector':
(hier ein 8-faches Bündel mit den Leitungen Nr.0 bis Nr.7)
→ es ist *ratsam*, die Reihenfolge abwärts anzugeben, sonst verdreht es Dir später bei Registern und Addierern die Bits

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
  d   : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
  q   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
);
END reg8;
```

Mit 'in', 'out' (und 'inout') gibt man die Daten-Richtung an:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
  x   :  IN   STD_LOGIC;
  y   :  OUT  STD_LOGIC;
);
END reg8;
```

13.1.9 Ampel, RSFF, 1-Bit-Latch, DFF

wie immer ist händische Mitschrift zu führen

was ist das:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY dingsda IS PORT(
  set,reset : IN    STD_LOGIC;
  Q,notQ    : INOUT STD_LOGIC;
);
END dingsda;

ARCHITECTURE Funktion OF dingsda IS
BEGIN
  Q    <= reset NOR notQ;
  notQ <= set   NOR Q;
END Funktion;

        -- was ist neu in diesem VHDL-Programm?
        -- wir haben dieses 'Ding' auch mit NAND statt NOR kennengelernt -
              beschreibe es in VHDL
        -- beizeiten sendest Du mir bitte wieder Mitschriftfotografien
```

was ist neu in diesem VHDL-Programm?

wir haben dieses 'Ding' auch mit NAND statt NOR kennengelernt - beschreibe es in VHDL!

ganze Prozessoren oder Speicher so zu beschreiben, gäbe schauderhaft umständlichen VHDL-Code
dafür hat man etwas Besseres/Einfacheres erfunden:
den VHDL-**'Process'**

Ein Latch (1):

- leider gibt es keinen Befehl "bau da jetzt ein DFF hinein" (oder ein Latch)
- man kann das VHDL aber unsichtbar dazu **zwingen**, Latches einzubauen, indem es anstatt eines neuen Wertes den alten nehmen muss
- diesen "alten" hat es aber nur, wenn er **gespeichert** wurde.
- was sind Latches? und Register? Was ist der Unterschied?

```
Q <= D when (Enable) else Q;
```

- **'others'** ::= *alle nicht genannten*
- **'all'** ::= *alle Möglichkeiten*

Ein Latch (2)

```
process(all)
begin
    Q <= D when (Enable);
end process;
```

Ein DFF ('D'-Flipflop) (3)

```
DFF : process(all) is
begin
  if (RST) then
    Q <= '0';
  elsif rising_edge(CLK) then
    Q <= D;
  end if;
end process DFF;
```

auch ein DFF (4)

```
DFF : Q <= '0' when (RST)  else D when rising_edge(clk);
```

13.1.10 8-Bit-Latch, Register

wie immer ist händische Mitschrift zu führen

```
-- 8-Bit-Latch:
ENTITY Modul IS
  PORT
  (
  Din : IN STD_LOGIC_VECTOR(7 downto 0);
  ena : IN STD_LOGIC;
  Q   : OUT STD_LOGIC_VECTOR(7 downto 0);
  );
END Modul;
ARCHITECTURE DLatch8 OF Modul IS
BEGIN
  latch : PROCESS (clk, Din)
  BEGIN

    IF (ena = '1') THEN -- wenn clk == '0'
      Q <= Din;         -- kriegt das Q keinen Wert,
    END IF;             -- also nimmts den alten

  END PROCESS latch;

END DLatch8;

--If clk equals '1', the value Din is assigned to Q.
--If clk equals '0', the circuit maintains its previous state, creating a latch.
```

* die DFF-Übung wurde wunderbar gemacht!
* ganze Prozessoren oder Speicher so zu beschreiben, gäbe schauderhaft umständlichen VHDL-Code
* dafür hat man etwas Besseres/Einfacheres erfunden: den *VHDL-'Process'*

```
-- 8-Bit-Register, steigende clk-Flanke:
ENTITY Modul IS
  PORT
  (
  Din : IN STD_LOGIC_VECTOR(7 downto 0);
  clk : IN STD_LOGIC;
  Q   : OUT STD_LOGIC_VECTOR(7 downto 0);
  );
END Modul;
ARCHITECTURE Dreg8 OF Modul IS
BEGIN
  register : PROCESS (clk)
  BEGIN

    IF rising_edge(clk) THEN    -- ausser bei steigender clk-Flanke
      Q <= Din;                 -- kriegt das Q keinen Wert,
    END IF;                     -- also nimmts den alten

  END PROCESS register;
END Dreg8;
```

```
        --leider gibt es keinen Befehl "bau da jetzt ein DFF hinein" (oder ein Latch)
        --man kann das VHDL aber unsichtbar dazu zwingen, Latches einzubauen,
          indem es anstatt eines neuen Wertes den alten nehmen muss
        --diesen "alten" hat es aber nur, wenn er gespeichert wurde.
        --was sind Latches? und Register? Was ist der Unterschied?

        Aufgabe: Schreib ein (flankengetriggertes) DFF als VHDL-Process
        Abgabe Deadline: Samstag Abend

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
  Din : IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
  Q   : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
  clr : IN  STD_LOGIC;
  clk : IN  STD_LOGIC;
);
END reg8;

ARCHITECTURE working OF reg8 IS
BEGIN
  process(clk, clr)
  begin
   if clr = '1' then
        Q <= x''00000000'';
   elsif rising_edge(clk) then
        Q <= Din;
   end if;
 end process;
END working;
```

oder mit negativer Flanke:

```
-- 8-Bit-Register, fallende clk-Flanke:
ENTITY Modul IS
  PORT
  (
  Din : IN STD_LOGIC_VECTOR(7 downto 0);
  clk : IN STD_LOGIC;
  Q   : OUT STD_LOGIC_VECTOR(7 downto 0);
  );
END Module;
ARCHITECTURE Dreg8 OF Modul IS
BEGIN
  register : PROCESS (clk, Din)
  BEGIN
    IF falling_edge(clk) THEN
      Q <= Din;
    END IF;
  END PROCESS register;
END Dreg8;
```

13.1.11 VHDL-'process'

wie immer ist händische Mitschrift zu führen

primitiv-Ampel:
schreib eine ganz primitive rot-gelb-grün-Ampel ohne irgendwelche Fussgänger oder Blinkphasen **in VHDL**.
Du wirst merken, dass Du SPEICHER-Elemente (DFF) für die Zustände brauchst.

- tu so, als könne man *Bits in Verbindungsleitungen speichern*
  (was natürlich in Wirklichkeit nicht geht)
- innere Verbindungsleitungen legt man mit
  signal (wie bei 'port')
- Syntax: **signal** Name **: typ**;

```
zB.:
 architecture Schaltung1 of Modul is
  signal vonAnachB: std_logic;
  signal vonCnachD: std_logic_vector( 7 downto 0);
 begin
  ...
 end;
```

- Web-Recherche:
  mache Dich zur Eignung der FPGA-Programme "Quartus IIünd "Quartus Prime"für unsere Zwecke schlau (wir sollten Digitalschaltungen erfassen und simulieren sowie das Ältera DE0Ëntwicklungskit programmieren können) zB. auf https://en.wikipedia.org/wiki/Intel_Quartus_Prime,
  https://www.google.com/search?q=Quartus+II
  und https://www.google.com/search?q=Quartus%20Prime.
  Fertige dabei ein "Recherche-Protokoll" (wo, was, wie nützlich...) an (Handmitschrift)
  (ganz professionell wäre gleich eine Nutzwertanalyse)
- beizeiten sendest Du mir bitte wieder Mitschriftfotografien

Der *'sequential'* **process**

- der 'process' wurde **für Programmierer** erfunden
- es wird **Hardware** erzeugt, die *Ähnliches* leistet.
  (*) das Zeitverhalten ist nicht das einer von-Neumann-Programmsoftware

```
<proc-name> ':' 'process'  <sensitivity-list>  'begin'

    -- hier koennen lokale Signale
       oder Variablen deklariert werden

begin

    -- hier ist ein 'sequential environment'
    ...

end <proc-name>;
```

<proc-name>	ist frei wählbar (bis auf die 'reserved words' wie *'entity'*, *'port'*, *'process'* ...)
<sensitivity-list>	führt alles an, worauf ein 'process' *sensibel* reagiert, dh. einen *'process'-Durchlauf* auslöst.

ein VHDL-Compiler erzeugt die Hardware so,
als würde der 'process' immer wieder durchlaufen
(Funktionen hintereinander in Serie geschaltet):

zum Beispiel:

```
count_up: process (clk, reset) begin

 +-------+
 |       |
 ^       v
 |   if (reset = '1') then
 |     count <= (others=>'0'); <----------> (others=>'0')
 ^   elsif (rising_edge(clk)) then          =alle anderen
 |     if (enable = '1') then               nichtgenannten
 |       count <= count + 1;                Einzelleitungen
 ^     end if;
 |   end if;
 |       |
 +-------+

end count_up;
```

jetzt kannst Du in VHDL schon
→ Register
→ Latches
→ Counter
→ Multiplexer und Decoder
→ kominatorische Schaltungen wie Prozessor-ALU
bauen!

Es wird Zeit, dass Du einen ganzen Prozessor baust!

dazu brauchsch den → *'von-Neumann-Zyklus'* **p.80**

13.1.12 nomal von vorn

Aufgabe: Schreib (in Dei Mitschrift) VHDL mit
- 8-Bit-Inputs A, B
- 8-Bit-Output Y
- einem 1-Bit-Input 'AddSub'
- wenn AddSub='0' gebe Y die Summe A+B aus,
  wenn AddSub='1' die Differenz A-B
- mache die Übung einmal mit 'with/select' und einmal mit 'when/else'
- primitiv-Ampel:
  schreib eine ganz primitive rot-gelb-grün-Ampel ohne irgendwelche Fussgänger- oder Blinkphasen **in VHDL**.
- Du wirst merken, dass Du SPEICHER-Elemente (DFF) für die Zustände brauchst.
- tu so, als könne man *Bits in Verbindungsleitungen speichern*
  (was natürlich in Wirklichkeit nicht geht)
- *innere Verbindungsleitungen* legt man mit
  signal (wie bei 'port')

Syntax: **signal** <Name> **: typ;**
zB.:

```
architecture Schaltung1 of Modul is
        ...
  signal   vonAnachB :  std_logic;
  signal   vonCnachD :  std_logic_vector( 7 downto 0);
begin
  ...
end;
```

13.1.13 primitiv-ALU OPs + Phasen

wie immer ist händische Mitschrift zu führen

```
with OPCODE select
  ERGEBNIS <=\\
      ERGEBNIS when ''1111 1111'' --NOP
      A+A      when ''0000 0100'', --ADD A,A u. SLA A
      A+B      when ''0000 0101'', --ADD A,B
      A-A      when ''0000 1100'', --SUB A,A
      A-B      when ''0000 1101'', --SUB A,B
      A+1      when ''0000 0000'', --INC A
      B+1      when ''0000 0001'', --INC B
      A-1      when ''0000 1000'', --DEC A
      B-1      when ''0000 1001'', --DEC B
      (-)A     when ''0001 1000'', --INV A
      (-)B     when ''0001 1001'', --INV B
      A AND B  when ''0011 0101'', --AND A,B
      A NAND B when ''0011 1101'',
      A OR B   when ''0011 0001'', --OR A,B
      A NOR B  when ''0011 1001'',
      A XOR B  when ''0011 0011'', --XOR A,B
      A XNOR B when ''0011 1011'',
      NOT A    when ''0010 1010'', --NOT A
      NOT B    when ''0010 1011'', --NOT B
      '1111 1111' when others;

      ALU - Arithmetics + Logics Unit ('Rechenwerk'):

        Ueberlege, welche 'operations' Dein Prozessor haben soll, wie zB.

               operation:             'mnemonic'      binaercode:
               'nichts'               NOP             1111 1111
               'A+1'                  INC A           0000 0000
               'B+1'                  INC B           0000 0001
               'A-1'                  DEC A           0000 1000
               'B-1'                  DEC B           0000 1001
               'A+B'                  ADD A,B         0000 0100
               'A-B'                  SUB A,B         0000 1100
               (-)A ( == not(A-1))    INV A           0001 1000
               (-)B ( == not(B-1))    INV B           0001 1001
               'not A'                CPL A           0010 1000
               'not B'                CPL B           0010 1001
               'A and B'              AND A,B         0011 0000
               'A or B'               OR  A,B         0011 1000
              usw.

              -- Gib den 'operations' binaere Nummern, die dann in VHDL mit
                 'if'/'case'/'when-else'/'with-select' unterschieden werden koennen.
              -- so eine Tabelle nennt man das 'instruction-set'
              -- die Zuordnung des Binaercode hat ein "System"
              -- 'Mnemonics' sind textuelle Instruktions-Namen

      CU - Control Unit ('Leitwerk')
              steuert die ALU und den gesamten Ablauf s. ==> 'von-Neumann-Zyklus'.
              Daraus ergibt sich der Bedarf an zusaetzlichen Registern
              zur Zwischenspeicherung binaerer Codenummern und Datenwerte:
        in der CU:
              -- von-Neumann-Phasen-Zaehler
              -- program counter ( = instruction ointer)
              -- instruction register
              -- Fallunterscheidung: von-Neumann-Phase-1, s. ==> 'von-Neumann-Zyklus'
              -- Fallunterscheidung: von-Neumann-Phase-2, s. ==> 'von-Neumann-Zyklus'
              -- Fallunterscheidung: von-Neumann-Phase-3, s. ==> 'von-Neumann-Zyklus'
              -- Fallunterscheidung: von-Neumann-Phase-4, s. ==> 'von-Neumann-Zyklus'
              -- Fallunterscheidung: von-Neumann-Phase-5, s. ==> 'von-Neumann-Zyklus'
        in der ALU:
              data pointer
              Rechen-Register zB.:
                      ALU-Register A
                      ALU-Register B
                      (oder eine ganze 'register-bank' R0, R1, R2 ...)
              ALU-Ergebnis-Register s. auch ==> 'von-Neumann-Zyklus'

      Abgabe: Foto der Handmitschrift mit Deinem bisherigen Prozessor-Entwurf in VHDL
```

13.1.14 ALU, CU, FETCH

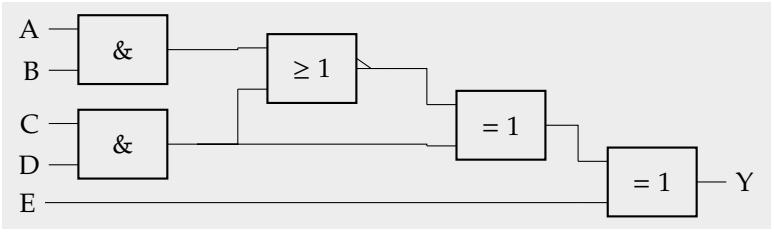
Fr-11Dez20:
wie immer ist haendische Mitschrift zu fuehren

```
ENTITY Prozessor IS
  PORT (
  Databus  : inout  STD_LOGIC_VECTOR(7 downto 0);
  Adressbus: out    STD_LOGIC_VECTOR(7 downto 0);
  clk      : in     STD_LOGIC;
  notWrite : out    STD_LOGIC;
  notRead  : out    STD_LOGIC;
  );
END Prozessor;
ARCHITECTURE Innenleben of Prozessor IS BEGIN
  ...
  ALU:process(clk) begin
  ...
  end ALU;
  CU: process(clk) begin
  ...
  end CU;
END Innenleben;
END Prozessor;


        Ergaenze zum bisherigen Prozessor-VHDL:
        Umsetzung der von-Neumann-Phase-1 nach VHDL s. 'von-Neumann-Zyklus':

        Instruction FETCH:
          --erhoehen des 16-Bit-Programmzaehlers (program counter, instruction pointer)
          --Programmzaehlerinhalt auf den 16-Bit-Adressbus legen,
                solange clock=="Hi"
          --READ-Signalleitung aktivieren (active low)
          --Wartezeit fuer RAM-Speicherbausteine
          --neuen Befehlscode (instruction code) vom 8-Bit-Datenbus
                ins 8-Bit-InstructionRegister einlesen
          --READ-Signalleitung de-aktivieren
          --Adressbus wieder freigeben (=Verbindung mit program counter trennen),
                sobald clock="Low"

        Abgabe: Handmitschrift-Foto Deines heute erweiterten Prozessor-Entwurfs in VHDL
        Abgabe-Termin: glei!
```

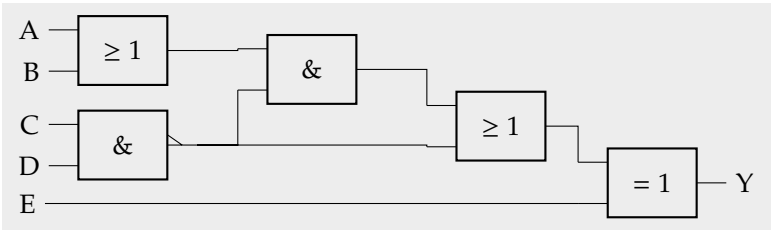
bitte fang damit schon einmal an!
Vielleicht beginnst Du bei der ALU, des hast schon einmal gemacht:
Überlege, welche 'operations' Dein Prozessor haben soll, wie zB. 'A+B', 'A-B', 'A+1', 'A-1', (-)A, 'not A', 'A and B', 'A or B' usw.
Gib den Operations binäre Nummern, die dann in VHDL mit 'if', 'when', 'select' udgl. unterschieden werden können.

13.1.15 VHDL-Übungsaufgaben

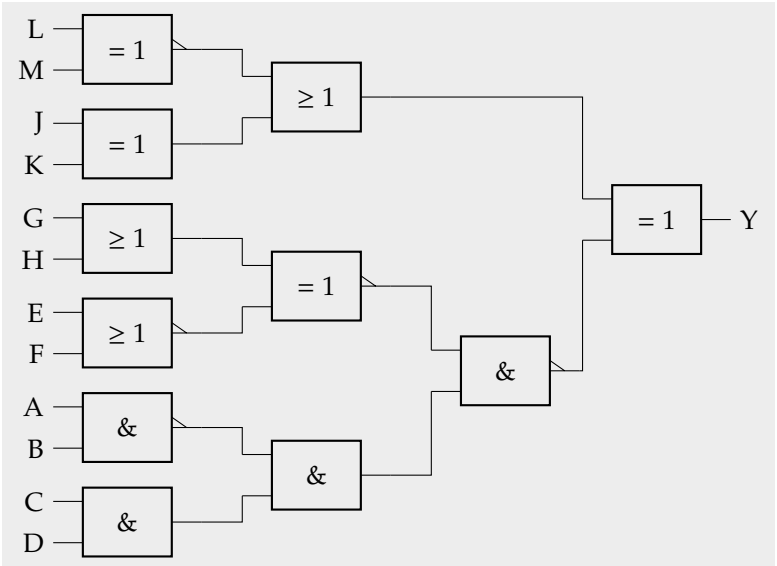
↪ schreibe folgende Gatterschaltungen auf VHDL um:
↪ vereinfache folgende Gatterschaltungen:



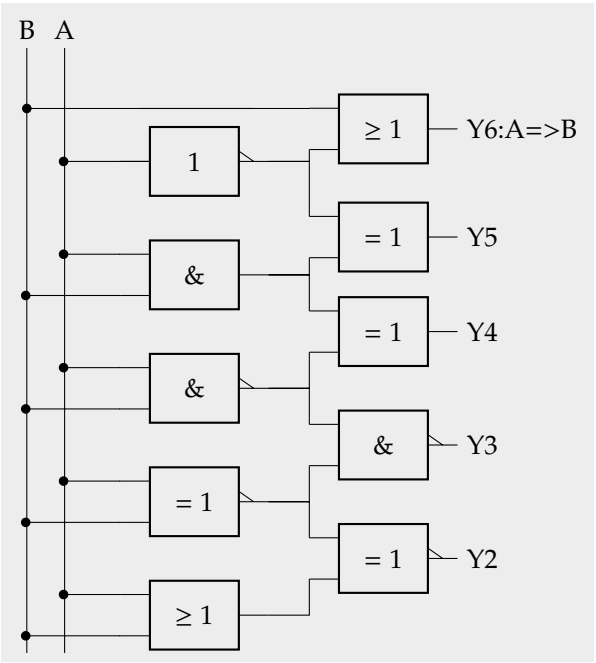
Bsp.Lösung:
Y <= (((A and B) nor (C and D))
xor (C nand D)) xor E



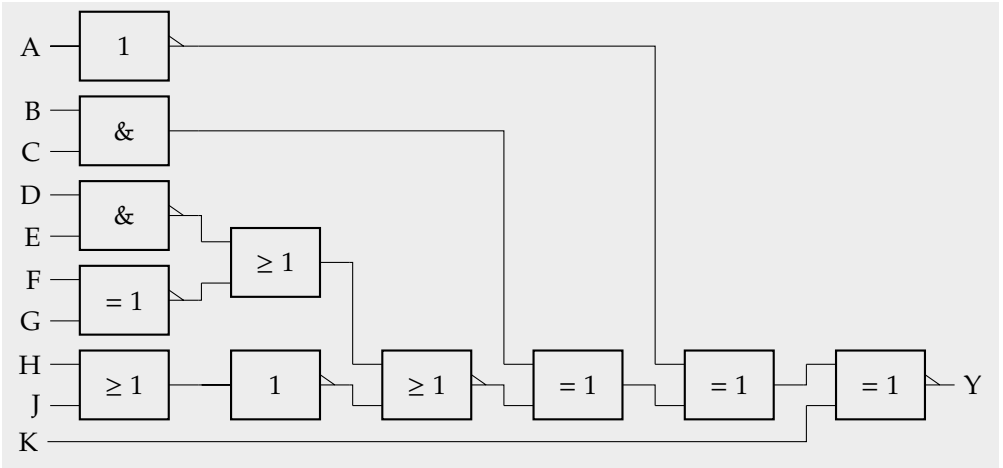
Bsp.Lösung:
Y <= (((A or B) and (C nand D))
or (C nand D)) xor E



Bsp.Lösung:
Y <= (((L xnor M) or (J xor K))
xor
(((G or H) xnor (E nor F))
nand
((A nand B) and (C and D))));



Bsp.Lösung:
Y6 <= B or not A;
Y5 <= not A xor (A and B);
Y4 <= (A and B) xor not(A and B);
Y3 <= not(not(A and B) and not(A xor B));
Y2 <= not(not(A xor B) xor (A or B));



Y = not xor((not A xor ((B and C) xor (((D nand E) or (F xnor G)) nor (not(H or J))))) xor K);

13.1.16 Truth Table (5)

→ gib die Funktion der Schaltung als **Wahrheitstabelle** an

- 'realisiere folgende Wahrheitstabelle in VHDL'

a b c	Y
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	0

```
begin
  rst <= (r s t);
  with rst select
    Y <=
        '1' when ''001'',
        '1' when ''101'',
        '1' when ''110'',
```

```
        '0' when others;
end;
```

oder (sequential):

```
architecture ... begin
  process (a,b,c) begin
    abc <= (a b c);
    case abc is
      when ''001'' => Y <= '1';
      when ''101'' => Y <= '1';
      when ''110'' => Y <= '1';
      when others => Y <= '0';
    end case;
  end process;
end;
```

13.1.17 Kombinatorik

↪ realisiere in VHDL:
Wenn timer1 und nicht timer2 und Zustand==stop aber Lichtschranken nicht blockiert und TorAuf-Taste nicht gedrückt, dann aktiviere Türschließmotoren

↪ realisiere in VHDL:
Wenn TuerZu und timer2 und Richtung='steigen', dann aktiviere LangsamfahrtAuf

```
Bsp.concurrent:

architecture ...
begin
schliessen <= '1'
   when timer1 = '1'
   AND timer2='0'
   AND zustand="stop"
   AND LBarrier='1'
   AND openButton='0'
   else '0';
LangsamfahrtAuf <= '1'
   when TuerZu = '1'
   AND timer2 = '1'
   AND Richtung="steigen"
   else '0';
end;
```

```
sequential:

architecture ...
begin
process(timer1,timer2,zustand,LBarrier,openButton)
begin
IF timer1 = '1'
   AND timer2='0'
   AND zustand="stop"
   AND LBarrier='1'
   AND openButton='0'
THEN schliessen <= '1'
ELSE schliessen <= '0'
END IF;
end process;
process(timer2,TuerZu,Richtung) begin
IF TuerZu = '1'
   AND timer2 = '1'
   AND Richtung="steigen"
THEN LangsamfahrtAuf <= '1'
ELSE LangsamfahrtAuf <= '0'
END IF;
end process;
end;
```

13.1.18 Latch, Register

↪ beschreibe ein 8-Bit-Latch in VHDL
↪ beschreibe ein 8-Bit-Register in VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
   d : IN   STD_LOGIC_VECTOR(7 DOWNTO 0);
   q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   clr : IN STD_LOGIC;
   clk : IN STD_LOGIC;
);
END reg8;

ARCHITECTURE working OF reg8 IS
BEGIN
   process(clk, clr)
   begin
     if clr = '1' then
       q <= x"00000000";
     elsif rising_edge(clk) then
       q <= d;
     end if;
   end process;
END working;
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
   d : IN   STD_LOGIC_VECTOR(7 DOWNTO 0);
   o : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   LE : IN STD_LOGIC;
   nOE : IN STD_LOGIC;
);
END reg8;

ARCHITECTURE working OF reg8 IS
   signal q : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
   process(LE,nOE)
   begin
     if LE = '1' then
       q <= d;
     end if;
     if nOE = '0' then
       o <= q;
     else
       o <= "ZZZZZZZZ";
     end if;
   end process;
END working;
```

13.1.19 counter

↪ realisiere up-counter und down-counter in VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ctr8 IS PORT(
   e : IN   STD_LOGIC_VECTOR(7 DOWNTO 0);
   q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   clr : IN STD_LOGIC;
   clk : IN STD_LOGIC;
);
END ctr8;

ARCHITECTURE working OF ctr8 IS
BEGIN
   process(clk, clr, e)
   begin
     if clr = '1' then
       q <= x"00000000";
     elsif falling_edge(clk) then
       q <= e + "1";
     end if;
   end process;
END working;
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ctr8 IS PORT(
   e : IN   STD_LOGIC_VECTOR(7 DOWNTO 0);
   q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   clr : IN STD_LOGIC;
   clk : IN STD_LOGIC;
);
END ctr8;

ARCHITECTURE working OF ctr8 IS
BEGIN
   process(clk, clr, e)
   begin
     if clr = '1' then
       q <= x"00000000";
     elsif rising_edge(clk) then
       q <= e - "1";
     end if;
   end process;
END working;
```

13.1.20 Bus+Register

18Dez20

Übung: Schreib folgende Prozessor-Bestandteile in VHDL:

↪ 16-Bit-Datenbus
↪ 24-Bit-Adressbus
↪ *'clk'*-Eingang
↪ *'notRead'*-Ausgang
↪ *'notWrite'*-Ausgang

↪ 16-Bit-ALU-Register 'A'
↪ 16-Bit-ALU-Register 'B'
↪ 24-Bit-ALU-Register 'DataPointer'

Musterlsg.:

```
ENTITY CPU16 IS PORT(
   DBUS : INOUT   STD_LOGIC_VECTOR(15 DOWNTO 0);
   ABUS : OUT STD_LOGIC_VECTOR(23 DOWNTO 0);
   clk  : IN STD_LOGIC;
   notRead : OUT STD_LOGIC;
   notWrite : OUT STD_LOGIC;
); END CPU16;
ARCHITECTURE working OF CPU16 IS
   SIGNAL A,
   B : STD_LOGIC_VECTOR (15 downto 0);
   SIGNAL DPTR : STD_LOGIC_VECTOR (23 downto 0);
```

13.1.21 RAM-Zugriffe

22Dez20

es muss eine 'Instruction' geben (die uns noch fehlt), die ein Byte aus einer RAM-Speicher-Stelle holt ('RAM read') und ins CPU-Register 'A' **liest**/kopiert.

Die RAM-Speicher-Adresse dafür stehe im ALU-'DataPointer' Register

a) Bitte beschreibe die Prozessor-'Instruction' LOAD A,RAM gem. → ***'von-Neumann-Zyklus'*** **p.80**

b) bitte auch eine fürs ALU-Register 'B'

c) bitte auch eine 'Instruction', die aus ALU-Register 'A' ins RAM zurück **speichert**, wird man brauchen: 'STORE A,RAM'

d) und bitte ebenso für das ALU-Register 'B'

Musterlsg.:

```
RAMREAD: process(clk) BEGIN
  if (PhaseCTR=3)
  AND InstReg(7 downto 2)="0100 000" THEN
    if rising_edge(clk) THEN
      ABUS <= DPTR;     --DataPointer auf Abus
      notRead <= '0';   --READ signal
    END if;
      -- Wartezeit für Speicherbausteine:
      -- (== warten auf fallende clk-Flanke)
    if falling_edge(clk) then
       --Datenregister vom Datenbus lesen
      case InstReg is
        when "0100 0000" => A <= DBUS;
        when "0100 0001" => B <= DBUS;
      END case;
      notRead <= '1';    -- release READ
      ABUS <= others => 'Z'; -- Adressbus freigeben
    END if;
  END if;
END RAMREAD;

RAMWRITE: process(clk) BEGIN
  if (PhaseCTR=3)
  AND InstReg(7 downto 2)="0110 00" THEN
    if rising_edge(clk) THEN
      ABUS <= DPTR;     --DataPointer auf Abus
      case InstReg is
        when "0110 0000" => DBUS <= A;
        when "0110 0001" => DBUS <= B;
        when "0110 0010" => DBUS <= ErgReg;
      END case;
      notWrite <= '0';   --WRITE signal
    END if;
      -- Wartezeit für Speicherbausteine:
      -- (== warten auf fallende clk-Flanke)
    if falling_edge(clk) then
      notWrite <= '1';    -- release WRITE
      ABUS <= others => 'Z'; -- Adressbus freigeben
      DBUS <= others => 'Z'; -- Datenbus freigeben
    END if;
  END if;
END RAMWRITE;
```

13.1.22 Phasen 2-5

wie immer ist händische Mitschrift zu führen

(mach vielleicht an Teil am Mo und einen am Di)

leider werdiXH wegen Maturaklassen-Unterrichts in der Anstalt am Dialog-Unterricht mit Dir gehindert, sodass Du das alleine im Arbeitsauftrag machen musst.

Es sind **ganz großartige Lösungen** der Aufgabe(Arbeitsauftrag) vom Fr-11Dez'20 eingesandt worden!

Ihr seids jo schun richtige CPU-Designer !!!

Teifl, Mander, Eys seids guat!
Nach 2 Monat schu Prozessor entwerfen ischa Hammer!

(leider von nur 6 ganz tollen Teilnehmern)

möchtegern-'Ingenieure', die nur wissen, wann sie *nicht* arbeiten wollen,
aber nicht einmal die Hälfte beherrschen,
braucht niemand :-((

Ergänze zum bisherigen Prozessor-VHDL (Bissel modifiziert und ergaenzt):

→ Umsetzung der von-Neumann-Phase-2 ins VHDL (s. ==> 'von-Neumann-Zyklus'):

2 Instruction DECODE
diese Wartezeit für die kombinatorische Decodier-Logik im 'Leitwerk' (control unit CU) besteht in der Wirklichkeit aus unumgänglichen Gatterlaufzeiten. Wenn diese zu knapp bemessen werden (zB. zu hohe 'clock' - Frequenz), dann wird schon weitergerechnet, bevor die Steuersignale/Ergebnisse stimmen, und die ganze 'Mühle' 'stürzt ab' (das erlebt man immer wieder beim 'Overclocking')

Eine der Abgaben sagt *'das RAM ist eh so schnell, dass man es nicht berücksichtigen muss'*
- da ist u.U. was dran, nur 'clocken' die Hersteller ihre PCs bis genau an diese Grenze

→ Umsetzung der von-Neumann-Phase-3 ins VHDL (s. ==> 'von-Neumann-Zyklus'):

3 Data LOAD
↓ DataPointer-Inhalt auf den Adressbus legen
↓ READ-Signalleitung aktivieren
↓ Wartezeit für Speicherbausteine
↓ Binärwert vom Datenbus in ein Datenregister einlesen
↓ READ-Signalleitung de-aktivieren
↓ Adressbus freigeben ('Z' state)

→ Umsetzung der von-Neumann-Phase-4 ins VHDL (s. ==> 'von-Neumann-Zyklus'):

4 Instruction EXECUTE
diese Wartezeit für die kombinatorische Rechen-Logik im 'Rechenwerk' (arithmetic and logic unit ALU) besteht wieder aus Gatterlaufzeiten in den Rechenschaltungen der ALU

→ Umsetzung der von-Neumann-Phase-5 ins VHDL (s. ==> 'von-Neumann-Zyklus'):

5 Data SAVE
↓ DataPointer-Inhalt auf den Adressbus legen
↓ Ergebnisregister-Inhalt auf den Datenbus legen
↓ WRITE-Signalleitung aktivieren
↓ Wartezeit für Speicherbausteine
↓ WRITE-Signalleitung de-aktivieren
↓ Adress- und Datenbus wieder freigeben ('Z' state)

→ Abgabe: Handmitschrift-Foto Deines heute erweiterten Prozessor-Entwurfs in VHDL
→ Abgabe-Termin: Di-15Dez'20 nach der Dic-Stunde

```
ENTITY Prozessor IS
  PORT (
  DBus     : inout  STD_LOGIC_VECTOR(7 downto 0);
  ABus     : out    STD_LOGIC_VECTOR(15 downto 0);
  clk      : in     STD_LOGIC;
  notWrite : out    STD_LOGIC;
  notRead  : out    STD_LOGIC;
  );
END Prozessor;
ARCHITECTURE Innenleben of Prozessor IS BEGIN
  /* 'ProgramCounter', auch 'InstructionPointer'
   *  == RAM-Adresse des naechsten CPU-Befehls ('instruction'):
   */
   PC  :signal STD_LOGIC_VECTOR (15 downto 0);


  /* von-Neumann-Phasen-Zaehler 'PhaseCTR' 1 bis 5
   * also 3 Bit:
   */
   PhaseCTR :signal STD_LOGIC_VECTOR (2 downto 0);  --sind 3 Bit!


  /* 'Data Pointer', auch 'Memory Pointer'
```

```
 * == RAM-Adresse fuer Daten-Byte:
 */
 DPointer:signal STD_LOGIC_VECTOR (15 downto 0);


/* 'instruction register'
 * speichert den 'instruction code' waehrend der Datenbus
 * ALU-Daten zu uebertragen hat:
 */
 InstReg :signal STD_LOGIC_VECTOR ( 7 downto 0);


/* CPU-Rechen-Register A und B fuer arithmetische und
 * logische Operationen wie 'A+B' oder 'A and B':
 */
 CpuRegA,
 CpuRegB :signal STD_LOGIC_VECTOR ( 7 downto 0);--Rechen-Register


/* CPU-Rechen-Ergebnis-Register zur Zwischenspeicherung der Ergebnisse
 * bis zur von-Neumann-Phase-Nr5-'Data SAVE'
 */
 ErgReg  :signal STD_LOGIC_VECTOR ( 7 downto 0);



ALU:process(clk) begin

  --aus den Abgaben von '1_8', '1_6', '1_5', '2_7', '2_8':

  if (PhaseCTR=4) then
      case InstReg is
          when '00000000' => ErgReg <= A  +  1;
          when '00000001' => ErgReg <= B  +  1;
          when '00001000' => ErgReg <= A  -  1;
          when '00001001' => ErgReg <= B  -  1;
          when '00000100' => ErgReg <= A  +  B;
          when '00001100' => ErgReg <= A  -  B;
          when '00010000' => ErgReg <=    -  A;--'CPL'
          when '00010001' => ErgReg <=    -  B;--'CPL'
          when '00101000' => ErgReg <=   not A;
          when '00101001' => ErgReg <=   not B;
          when '00110000' => ErgReg <= A and B;
          when '00111000' => ErgReg <= A or  B;
          --when '11111111' => ErgReg <= ErgReg;--brauchma nid
          when  others    => ErgReg <= ErgReg;
      end case;
  end if;
end ALU;



CU: process(clk) begin
  PhaseCTR <= PhaseCTR+1;
  if rising_edge(clk) then
    if (PhaseCTR=1) then
      PC <= PC+1;
      ABus <=  PC;
      notRead <= '0';
          --'insert irrelevant Delay here'
      InstReg <= DBus;
      notRead <= '1';
      ABus <= (others => 'z');
    elseif (PhaseCTR=2) then
      --(tue gar nichts)
    elseif (PhaseCTR=3) then
      --'data LOAD'
      ABus <= DPointer;
      notRead <= '0';
        /* Huch!
         * da fehlen uns ja noch Befehle!
         * -> bitte selber erfinden
         */

      case InstReg is
        when ... => A <= DBus;
        when ... => B <= DBus;
      end case;
      notRead <= '1';
      ABus <= (others => 'z');
    elseif (PhaseCTR=4) then
      --'instruction EXECUTE'

      ...
```

```
      elseif (PhaseCTR=5) then
        --'data SAVE'
        ABus <= DPointer;
          /* Huch!
           * da fehlen uns ja  auch noch Befehle!
           * -> bitte selber erfinden
           */

        case InstReg is
          when ... => DBus <= A;
          when ... => DBus <= B;
        end case;
        notWrite <= '0';
          --'insert irrelevant Delay here'
        notWrite <= '1';
        DBus <= (others => 'z');
        ABus <= (others => 'z');
      end if;
    end if;  --rising_edge(clk)
  end CU;
END Innenleben;
```

13.1.23 RESET input

↪ Implementiere einen low-aktiven '-RESET' Eingang
und alle seine Prozessor-Funktionen
in VHDL

↪ '-RESET' setzt den ProgramCounter (PC, InstructionPointer), das InstructionRegister (InstREG), den von-Neumann-Phasen-Zähler (PahseCTR) und die CPU-Flags (CY, N, V, I, ...) auf Null (=> others '0'), verändert aber keine Register- oder RAM-Inhalte

13.1.24 von-Neumann Architektur

wie immer ist händische Mitschrift zu führen

* Schreib eine 8-Bit von-Neumann-Architektur in VHDL!
  (nicht den vonNeumann-Zyklus sondern die Architektur ie. die Bestandteile)
* Abgabe: Handmitschrift-Foto(s) Deines VHDL-Entwurfs per eMail an 'c.schoenherr' at 'tsn.at'
* Abgabe-Termin: Fr-08Jan'21 nach der Dic-Stunde

13.1.25 FETCH + SAVE

wie immer ist händische Mitschrift zu führen

* Schreib die vonNeumann-Phase-Nr1: 'Instruction Fetch' in VHDL,
  d.h. den Code im 'CU: process (clk)',
  und nur den Teil für die 'Instruction-Fetch'-Phase
  (die Architektur ie. die Bestandteile hast ja am 08.Jan'21 schon beschrieben)
* die "Wartezeit für Speicherbausteine" realisieren wir, indem wir auf die fallende Takt-Flanke 'falling_-edge(clk)' warten:

```
if falling_edge(clk) then
  InstReg &lt;= DBus;
  notREAD &lt;= '1';
  ...
```

* Schreib auch die vonNeumann-Phase-Nr5: 'Data Save' in VHDL,
  d.h. nur den Teil fü die 'Data Save'-Phase im 'CU: process (clk)'
* Abgabe: Handmitschrift-Foto(s) Deines VHDL-Entwurfs per eMail an 'c.schoenherr' at 'tsn.at'
* Abgabe-Termin: Di-12Jan'21 nach der Dic-Stunde

13.1.26 Register File

wie immer ist händische Mitschrift zu führen

* Moderne Prozessoren haben statt Register A und B ein sogenanntes 'Register File' (die 'AVR-8-Bit'-Controller zB. 'R0'..'R31'), welche sie mittels eines **internernen Datenbus** mit den Recheneinheiten der ALU verbinden.
  → Schreib eine 8-Bit vonNeumann-Architektur mit zusätzlich 2 internen Register-Datenbussen RDbus1, RDbus2 und 2 zusätzlichen Register-Adressbussen RAbus1, RAbus2 zur Realisierung einer ALU mit 32er-Registerbank in VHDL!
* Abgabe: Handmitschrift-Foto(s) Deines VHDL-Entwurfs per eMail an 'c.schoenherr' at 'tsn.at'
* Abgabe-Termin: Fr-15Jan'21 nach der Dic-Stunde

13.1.27 falling_edge

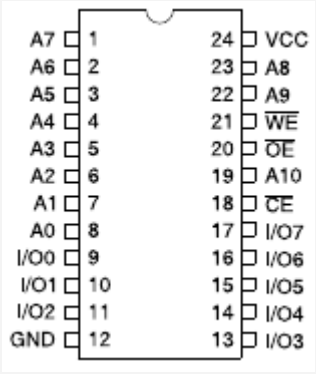
wie immer ist händische Mitschrift zu führen

* stell bitte alle 5 von-Neumann-Phasen in der CU so um, dass die 'Wartezeiten' durch fallende Flankentriggerung ersetzt werden
* Abgabe: Handmitschrift-Foto(s) Deines VHDL-Entwurfs per eMail an 'c.schoenherr' at 'tsn.at'
* Abgabe-Termin: Di-19Jan'21 nach der Dic-Stunde

13.1.28 SRAM

wie immer ist händische Mitschrift zu führen

* In der vorletzten Übung (Register-Bank) ist Dir wohl die Idee zum VHDL-Statement für die Verwendung der Register-Datenbusse RDbus1, RDBus2 abgegangen. Klar, es fehlen ja auch die 32 Register.
* Das 'Pin-Out' eines statischen RAM-Bausteins 'SRAM 6116':



-WE ... 'Write Ennable' == notWrite
-OE ... 'Output Ennable' == notRead
-CE ... 'Chip Ennable' == notCS chip select
I/O0 - I/O7 ... Data Bus D0-D7 (8 Datenleitungen)
A0-A10 ... 11 Adress-Leitungen (Adressen 0..2047)

* wie lasse ich das VHDL einen Datenspeicher erzeugen?

  fü unsere Prozessor-interne Registerbank schreibt man in VHDL:

```
  signal RgBank:  array(0 to 31) of STD_LOGIC_VECTOR(7 downto 0);

          wie in:

ARCHITECTURE Innenleben  of Processor IS BEGIN
    ...
  signal RgBank:  array(0 to 31) of STD_LOGIC_VECTOR(7 downto 0);
    ...
begin
  ALU: Process(clk) BEGIN
    ...
```

```
    elseif (PhaseCTR=3) then
      if rising_edge(clk) then
        RDBus1 <= RgBank(to_integer(unsigned( RABus1 )));
        ...
      end if;
    ...
    elseif (PhaseCTR=5) then
      if rising_edge(clk) then
        RgBank(to_integer(unsigned( RABus2 ))) <= RDBus2;
      end if;
    ...
  end process;
END Innenleben;


        * Einen reinen SRAM-Speicher-Baustein (ohne Prozessor) wie diesen '6116'
          wuerden wir so schreiben:

ENTITY RAM2k IS
  PORT (
  DBus   : inout  STD_LOGIC_VECTOR(7 downto 0);
  ABus   : out    STD_LOGIC_VECTOR(11 downto 0);
  nWR,
  nRD,
  nCS    : in    STD_LOGIC;
  );
END RAM2k;

ARCHITECTURE RamInnen of RAM2k IS BEGIN

  signal sRAM : ARRAY (0 to 2047) of STD_LOGIC_VECTOR(7 downto 0);

  Process(nRD, nWR) BEGIN
    if not( nCS ) AND NOT( nRD ) then
      DBUS  <=  sRAM( to_integer(unsigned( ABus )));
    elseif not( nCS ) AND NOT( nWR ) then
        sRRAM( to_integer(unsigned( ABus ))) <= DBus;
    end if;
  END;

END RamInnen;
```

* Abgabe: Handmitschrift-Foto(s) Deines VHDL-Entwurfs per eMail an 'c.schoenherr' at 'tsn.at'
* Abgabe-Termin: Fr-22Jan'21 nach der Dic-Stunde

13.1.29 JUMP

wie immer ist händische Mitschrift zu führen

* eine Klärung: (Dank an 'Boti'!) Die Trennung eines Prozessors in ALU und CU ist → von-Neumanns Idee. Müssen tut man nicht. Bis zur Erfindung des Mikroprozessor 1971 war diese Trennung zudem wegen der meterlangen Leitungen notwendig. Seither weicht diese Aufteilung auf den Prozessor-Chips zusehends einem gewissen Sauhaufen

  Wir halten die Trennung bitte weiterhin aufrecht.

  In der Programmcode-Struktur haben wir die 5 von-Neumann-Phasen in der CU, während Arithmetik und Kombinatorik in der ALU landen.

  (Das Prinzip, *alles was mit Daten zu tun hat in die ALU, Instructions in die CU*, is eher mutwillig)

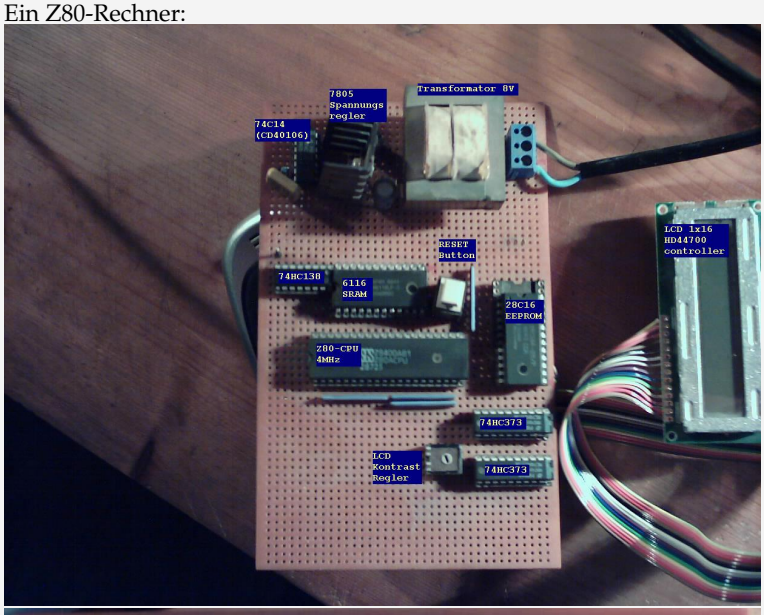
* Implementiere zusätzlich einen **GOTO**-Befehl (Prozessorfachleute sagen '**JUMP**-Instruction') in Deinem Prozessor in VHDL, bestehend aus
  - 1 Byte Instruction Code (als Instruction FETCH)
  - 1 Byte Sprung-Ziel-Adresse (als DATA LOAD)

  (für 16-Bit-Adressen isch des zwar 1 Byte zu wenig, des mochat ins die CU vorerst aber zu komplex)
* Bei diesem **'JUMP'** ist in Phase #3 der Datenbus-Inhalt in den Instruction-Counter/Pointer ('InstCTR') zu laden
* wir studieren im `Datenblatt des Microcontrollers AVR AT Tiny-13A`
  → das 'Pinout of ATtiny13A' im 8-pin-PDIP-Gehäuse (case)
  und im
  `avr-instruction-set-manual.pdf`

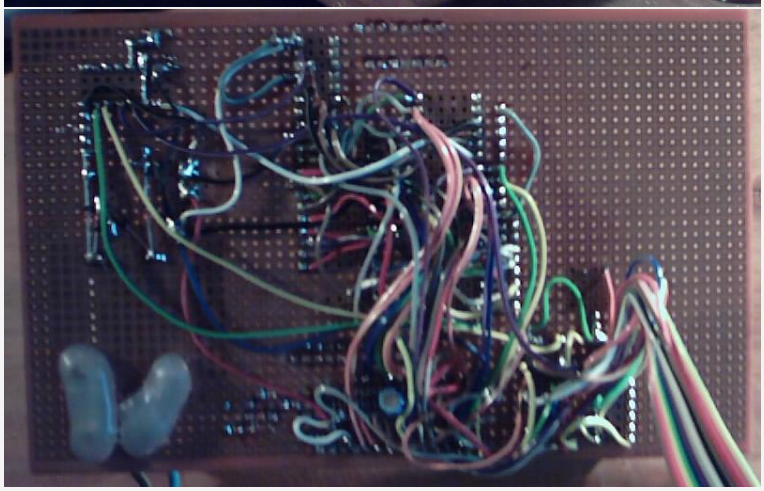
→ die **RJMP** Instruction im Vergleich zur **JMP** Instruction

* Abgabe: Handmitschrift-Foto(s) Deines VHDL-Entwurfs
  + Kurzfassung der heutigen Aufgabe
  per eMail an 'c.schoenherr' at 'tsn.at'
* Abgabe-Termin: Di-26Jan'21 nach der Dic-Stunde

Ein Z80-Rechner:



und die Unterseite:



13.1.30 BRANCH

wie immer ist händische Mitschrift zu führen

- zum Programmieren sind 'if'-Fallunterscheidungen notwendig
- studiere im `Datenblatt des Microcontrollers AVR AT Tiny-13A`
  und im `avr-instruction-set-manual.pdf`

  ↪ welche Instructions zur Fallunterscheidung existieren
  ↪ die CPU-'Flags' (status register SREG)
  ↪ 'conditional BRANCH' instructions
- notiere jedes gefundene Faktum plus der Stelle im Handbuch (manual)

- Abgabe: Handmitschrift-Foto(s) Deines Manual-PDF-Research per eMail an 'c.schoenherr' at 'tsn.at'
- Abgabe-Termin: Fr-29Jan'21 nach der Dic-Stunde

13.1.31 ASM = Assembler

wie immer ist händische Mitschrift zu führen

μC Programmierung (μC = MicroController):

Wir programmieren diese AVR-ATtiny13 (die wir noch gar nit haben) in der Programmiersprache **"Assembler"**, das ist die **Maschinensprache**, in der die Instruction-Codes durch **"Mnemonics"** genannte, sprechende Wortfetzen wie "ADD", "MOV", "JUMP" ersetzt sind.

→
Wir müssen die Assembler-Mnemonics im `avr-instruction-set-manual.pdf`
`http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf`
`https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-Instruction-Set-Manual-DS40002198A.pdf`
`https://en.wikipedia.org/wiki/Atmel_AVR_instruction_set`
lernen/verwenden.
Oft ist auch die Hardware-Ausstattung des μC lt.
local: `zuig/Datenblatt des Microcontrollers AVR AT Tiny-13A`
full: `https://ww1.microchip.com/downloads/en/DeviceDoc/doc8126.pdf`
full: `http://www.getchip.net/wp-content/uploads/ATTiny13.pdf`
short: `https://ww1.microchip.com/downloads/en/devicedoc/8126s.pdf`
anzusprechen.

Bitte noch keine Developement-Tools installieren!
(es würde due ersten Lernschritte nur stören).

(wir werden dann den `'gavrasm' - Gerd's AVR Assembler`
und später auch das
`Atmel Studio (vorm.AVR-Studio)` (mikrocontroller.net)
`www.microchip.com/avr-support/atmel-studio-7` (microchip.com)
verwenden)

Programm-1: Eine Zähl-Schleife:

```
        LDI     R30,10
LOOP:
        DEC     R30
        BRNE    LOOP
```

Fertige je ein
`Flussdiagramm`
`Struktogramm`
`Pseudocode`-Programm
dieses Code-Stückes an.

Programm-2: "LebenszeichenËndlosschleife mit PortB.1 Ausgabe

```
MAIN:           ;das HAUPTPROGRAMM: Nur "Lebenszeichen" ausgeben:
        IN      R17,PORTB       ;PINB...InputRegister von Port-B
        CBI     PORTB,1         ;Bit Nr.1 auf "Low"
        SBRS    R17,1           ;Uebersprinqe, wenn das Bit==1 war
        SBI     PORTB,1         ;Bit Nr.1 auf "High"
        RJMP    MAIN            ;RJMP == "Relative JuMP" == "goto"
```

Fertige je ein
`Flussdiagramm`

Struktogramm

Pseudocode-Programm

dieses Code-Stückes an.

Programm-3: das vollständige "Lebenszeichen-Assembler-Programm"

```
;das (kleine) "LEERE" Atmel AVR AT Tiny13 Programm
;  (fuer Linux-Assembler "gavrasm")
;-------------------------------------------------------------------------
.device ATtiny13        ;Gerd's AVR-Assembler taugt fyr viele Controller
.def    Rcnt    =R16    ;Schleifenzaehler Rcnt...alias/nickName fyr "Counter"
.def    Rtemp   =R17    ;"temporary" fuer kurzes Zwischenspeichern
 ;
        .CSEG           ;CodeSpeicher (ProgrammFlash) Segment
        .ORG    0x0000  ;Speicher-Adress-Festlegung "Origin"
        RJMP    INIT            ;wird beim Start und RESET aktiviert
         RETI   ;EXT_INT0       ;ISR ist "leer" (nur "RETI")
         RETI   ;PINCHG
         RETI   ;T0OVF
         RETI   ;EE_RDY
         RETI   ;ANACOMP
         RETI   ;TMR0_CMPA
         RETI   ;TMR0_CMPB
         RETI   ;WDR
         RETI   ;ADC
INIT:
        CLI                             ;disable all Interrupts
        LDI     Rtemp,Low(RAMEND)       ;"stack" einrichten:
        OUT     SPL,Rtemp
        LDI     Rtemp,0b00000011        ;PB0,PB1 ... output
        OUT     DDRB,Rtemp
        OUT     PORTB,Rtemp             ;Outputs auf "Hi" schalten
        SEI
;
MAIN:           ;das HAUPTPROGRAMM: Nur "Lebenszeichen" ausgeben:
        IN      Rtemp,PORTB     ;PINB ... InputRegister von Port-B
        CBI     PORTB,1         ;Bit Nr.1 auf "Low"
        SBRS    Rtemp,1         ;'SKIP if BIT in REGISTER is SET'
                                ;Ueberspringe, wenn Bit==1 war
        SBI     PORTB,1         ;SET BIT in I/O
                                ;Bit Nr.1 auf "High"
        RJMP    MAIN            ;RJMP ... "Relative JuMP" == "goto"
.EXIT                           ;Ende der Assemblierung
```

Dies dient jetzt nur einmal zum Anschauen.

Aufgaben:

- was tut Programm-1 ?
- was tut Programm-2 ?
- was tut Programm-3 ?

Abgabe: Handmitschrift-Foto(s) Deines Manual-PDF-Research
per eMail an 'c.schoenherr' at 'tsn.at'

Abgabe-Termin: Di-02.Feb'21 nach der Dic-Stunde

13.1.32 PulsGen

Fr 05Feb21 PulseGenerator

wie immer ist händische Mitschrift zu führen

Entwirf in einem selbsterdachten Maschinen-Pseudocode:
Ein Programm welches ewig kurze Pulse ausgibt,
indem es die Pausen zwischen den Pulsen sowie die Zeit während der Pulse mit nichtstuenden Warteschleifen verbringt ("busy waiting")

Schätze diese Zeiten bei 1MHz CPU-Takt (clock) anhand der Angaben
unter "Cycles"
im AVR8-instruction-set-manual.pdf

schreib es dann in "AVRischer" Assemblersprache.

13.1.33 SP, CALL u. RETURN

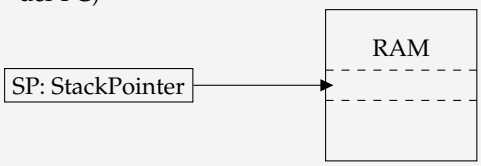
SP, CALL, RETURN Mo 15.Feb'21, Di 16.Feb'21

Übung: Schreibs in VHDL

wie immer ist händische Mitschrift zu führen

Der **Stack-Pointer SP**

Der **Stack Pointer "SP"** ist ein CPU-Register in der CU und speichert eine RAM-Adresse (ähnlich wie der PC)



Der SP-adressierte RAM-Bereich heißt "Stack", weil er als "Stapel" dient – man legt immer oben drauf und nimmt immer von oben wieder weg. Der AVR-uC-Stack ist kopfüber - er "wächst von oben nach unten".
bei gewohnter, "nach oben" wachsender RAM-Nutzung überschreiben sich Stack und Variablen auf diese Weise erst, wenn das gesamte RAM voll ist

Unterprogramm-Aufruf mit **CALL**

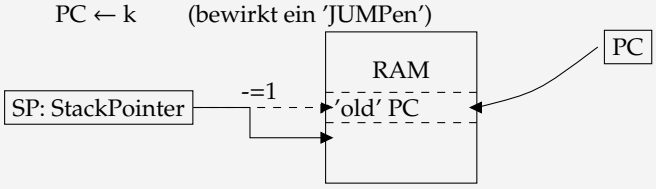
An der SP-Adresse speichert die **CALL**-Instruction die Angabe, wo es nach einer Unterbrechung '**wieder weiter** geht'. Der SP-adressierte RAM-Speicher ist quasi ein PC-'Backup'. Bei "CALL" (und bei "Interrupt") wird automatisch so ein 'Backup' im RAM angelegt,

Ablauf von 'CALL k':

CALL (8 Bit OpCode)

k (2 x 8 Bit = 16 Bit Sprungziel-Adresse)

RAM[SP] ← PC+3 (Adresse der naechsten Instruction)
SP ← SP-2 (2 Byte der Ruecksprung-Adresse)
PC ← k (bewirkt ein 'JUMPen')



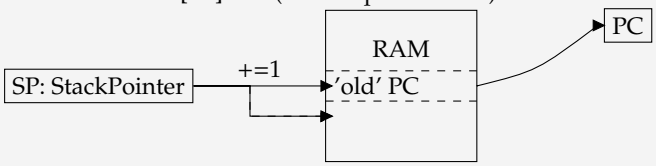
Die Rückkehr vom Unterprogramm mit **RETURN** bzw. **RETI**

Bei "RET" (bzw. "RETI") wird das 'PC-Backup' wieder aus dem RAM gelesen und in den PC kopiert. Das entspricht einem 'JUMP' zurück dorthin, wo vorher mit 'CALL' (bzw. 'Interrupt') unterbrochen wurde.

Ablauf von 'RET':

RET (8 Bit OpCode)

SP ← SP+2
PC ← RAM[SP] (StackTop in den PC)



Abgabe: Handmitschrift-Foto(s) per eMail an 'c.schoenherr' at 'tsn.at'

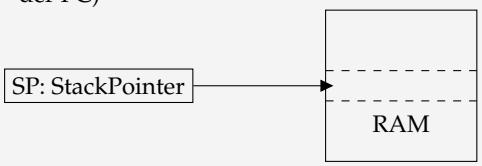
Abgabe-Termin: Di-16.Feb'21 nach der Dic-Stunde

13.1.34 PUSH u.POP

SP, PUSH + POP
Übung: Schreibs in VHDL

Der **Stack-Pointer SP**

Der **Stack Pointer "SP"** ist ein CPU-Register in der CU und speichert eine RAM-Adresse (ähnlich wie der PC)

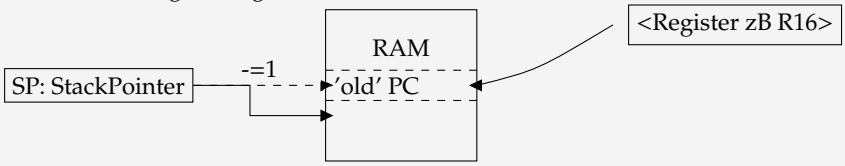


Der SP-adressierte RAM-Bereich heißt "Stack", weil er als "Stapel" dient – man legt immer oben drauf und nimmt immer von oben wieder weg. Der AVR-uC-Stack ist kopfüber - er "wächst von oben nach unten" s. **'CALL'**.
bei gewohnter, "nach oben" wachsender RAM-Nutzung überschreiben sich Stack und Variablen auf diese Weise erst, wenn das gesamte RAM voll ist

Um auch Anderes als die Unterprogramm-Rücksprungadresse 'stapeln' zu können, gibt es die Instructions

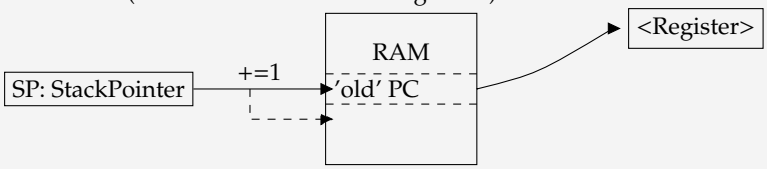
`PUSH <Register>`

PUSH (gib <Register>-Inhalt auf den 'Stack')



`POP <Register>`

POP (nimm vom 'Stack' ins <Register>)



Abgabe: Handmitschrift-Foto(s) per eMail an 'c.schoenherr' at 'tsn.at'

Abgabe-Termin: glei nach der Dic-Stunde

13.1.35 Interrupt+RETI

Übung: Schreibs in VHDL

'Interrupt' (im Englischen ein Zeitwort (verb)!!) ist eine elektrisch ausgelöste 'CALL'-Instruction an eine hardwaremäßig fix 'verdrahtete' Unterprogramm-Adresse (zB. bei 'Timer0-Ablauf' nach Programmspeicher-Adresse '0003' (16-Bit-Instruction-Word), wo als Unterprogramm eine sog, 'Interrupt-Service-Routine' (ISR) gespeichert wird.

So ein 'Interrupt' kann an jeder Stelle des Hauptprogramms ausgelöst werden und dieses (unvorhersehbar) **unterbrechen** (daher der Name 'interrupt')

Dabei setzt die CPU das 'I'-Flag im Statusregister 'SREG', welches weitere Interrupts sperrt.

Statt 'RETURN' wird eine ISR mit 'RETI' (Return from Interrupt) beendet - das 'RETI' löscht das 'I'-Flag im SREG wieder, sodass nachfolgende Interrupts wieder aktiviert werden.

Sollte schon während der ISR ein weiteres Interrupt-Signal eingetroffen sein, so wird dieser 'Interrupt' sofort nach dem 'RETI' ausgelöst (nachdem zuvor jedoch genau 1 Hauptprogramm-Instruction ausgeführt wurde und diese nicht ausgerechnet ein 'CLI' ist)

Aufgabe-1: Formuliere eine Interrupt-Aktivierung in VHDL:

```
    if(external_interrupt-signal) then
      --in der FETCH -Phase:
      PC <= ''0000 0000 0000 0001'';    --PGM-Adress 0x0001

      --in LOAD und SAVE-Phasen tut sich in diesem Fall gar nix
    end if
```

Aufgabe-2: Formuliere ein RETI in VHDL:

```
    if( InstReg = RETIcode ) then
      --Stack-Top in den PC
        ... (Du!)

      --SP adjustieren
        ... (Du!)

      --'I'-Flag loeschen
        ... (Du!)
    end if
```

13.1.36 ISR = InterruptServiceRoutine

ISR - InterruptServiceRoutine

```
Beispiel fuer eine primitive 'Anacomp-Analogkomparator'-ISR:

ANACOMP:PUSH    R16             ;R16 retten
        IN      R16,SREG        ;SREG nach R16 retten
        ADDIW   R30,1           ;R30 | R31 incrementieren
        OUT     SREG,R16        ;SREG restaurieren
        POP     R16             ;R16 restaurieren
        RETI                    ;ISR Ende
```

13.1.37 Interrupt Vectors Vc2

AVR Interrupt Vectors

```
die 'AT tiny13A:
Pgm     Int.Vector
Address
------- -----------------
0000    RJMP    INIT            ;wird beim Start und RESET aktiviert
0001    RETI    ;EXT_INT0       ;ISR ist "leer" (nur "RETI")
0002    RETI    ;PINCHG
0003    RETI    ;T0OVF
0004    RETI    ;EE_RDY
0005    RJMP    ANACOMP
0006    RETI    ;TMR0_CMPA
0007    RETI    ;TMR0_CMPB
0008    RETI    ;WDR
0009    RETI    ;ADC
```

13.2 aus der Webseite

```
          \input{KWx/latex/modDic1/HDL/webDic3a.html}
<!--SP + CALL + RETURN-->
<!--Pulse+PWM-->



</table><table border=1 width=100% style="background-color:#E0F0F0;">
<tr><td colspan=2>
        <div style="font-size:18pt;background-color:#004040;color:#C0F0FF;width:150mm;">
          welcome to se <b><big>Dic3</big></b>
          <div style="height:7mm;"></div>
        </div>
        <div style="font-size:18pt;background-color:#E0F0F0;color:#000000;width:150mm;">
          <a href="zuig/Dic3UXh01.pdf">==> Dic3a PDF</a>
        <div>
        <div style="height:7mm;"></div>
        <div style="font:500 12pt/120% times;background-color:#F0F0F0;color:#000000;padding:1mm 0mm 1mm 1m
          <big><u>Mo   -   21Dez20   Grp1+2 je 1 Std</u><br>   sowie
            <br><u>Di   -   22Dez20   ganzeKlasse </u></big>
          <div style="height:4mm;"></div>
          <div style="padding:20 0 2 30;">
            leider werdiXH wegen Maturaklassen-Unterrichts in der Anstalt
            am Dialog-Unterricht mit Dir gehindert,
            sodass Du das bitte wieder alleine im Arbeitsauftrag machen musst.
          </div>
          <div style="height:3mm;"></div>
          wie immer ist h&auml;ndische Mitschrift zu f&uuml;hren
          <div style="height:5mm;"></div>
        </div>
        <ul style="width:150mm;font:500 12pt times;">
        <li>&Uuml;bung:   Schreib in VHDL:
          <div style="height:5mm;"></div>
        <ul>
        <li>es muss eine 'Instruction' geben (die uns noch fehlt), die ein Byte aus einer RAM-Speicher-Ste
          ('RAM read')
          und ins CPU-Register 'A' <b>liest</b>/kopiert.
          <div style="height:.5mm;"></div>
          Die RAM-Speicher-Adresse daf&uuml;r stehe im ALU-'DataPointer' Register
          <div style="height:1mm;"></div>
          a) Bitte beschreibe diese Prozessor-'Instruction' gem. <a
                \hyperref[labelVonNeumannZyklus01]{==> 'von-Neumann-Zyklus'}
                in VHDL
          <div style="height:5mm;"></div>
        \item   b) bitte auch eine f"urs ALU-Register 'B'
        \item   c) bitte auch eine 'Instruction', die aus ALU-Register 'A' ins RAM zur"uck
                {\bf speichert ('RAM write')},
                wird man brauchen
        \item   d) und bitte ebenso f"ur das ALU-Register 'B'
        \item   Abgabe: Handmitschrift-Foto(s) Deines  VHDL-Entwurfs
        \item   Abgabe-Termin: Di-22Dez'20 nach der Dic-Stunde
        \item   dann w"unsch' iXH Dir
                \\[1mm] gaaaaaanz tolle Weihnachten (Xmas)
                \\[1mm] und Ferien mit viel Schlaf und traumhaften Prozessor-Entwicklungen
                \\in VHDL ...
                \\[1mm] aber fall mir nicht vom Christbaum :-))
        }



</table><table border=1 width=100% style="background-color:#E0F0F0;">
<tr><td colspan=2>
        <div style="font-size:18pt;background-color:#004040;color:#C0F0FF;width:150mm;">
          welcome to se <b><big>Dic3</big></b>
          <div style="height:7mm;"></div>
        </div>
        <div style="font-size:18pt;background-color:#E0F0F0;color:#000000;width:150mm;">
          <a href="zuig/Dic3UXh01.pdf">==> Dic3a PDF</a>
        <div>
        <div style="height:7mm;"></div>
        <div style="font:500 12pt/120% times;background-color:#F0F0F0;color:#000000;padding:1mm 0mm 1mm 1m
          <big><u>Fr-18Dez20 Grp1, 1210-1300h</u></big>
          <div style="height:4mm;"></div>
          <div style="padding:20 0 2 30;">
```

```
            leider werdiXH wegen Maturaklassen-Unterrichts in der Anstalt
            am Dialog-Unterricht mit Dir gehindert,
            sodass Du das alleine im Arbeitsauftrag machen musst.
          </div>
          <div style="height:3mm;"></div>
          wie immer ist h&auml;ndische Mitschrift zu f&uuml;hren
          <div style="height:5mm;"></div>
        </div>
        <ul style="width:150mm;font:500 12pt times;">
        <li>&Uuml;bung:<br> Schreib folgende Prozessor-Bestandteile in VHDL:
        <ul>
        <li>16-Bit-Datenbus
        <li>24-Bit-Adressbus
        <li>'clk'-Eingang
        <li>'notRead'-Ausgang
        <li>'notWrite'-Ausgang
        <li>16-Bit-ALU-Register 'A'
        <li>16-Bit-ALU-Register 'B'
        <li>24-Bit-ALU-Register 'DataPointer'
        </ul>
        <li>Abgabe: Handmitschrift-Foto Deines heutigen  VHDL-Entwurfs
        <li>Abgabe-Termin: Fr-18Dez'20 nach der Dic-Stunde
        </ul>
        <div style="height:50mm;"></div>


</table><table border=1 width=100% style="background-color:#E0F0F0;">
<tr><td colspan=2>
        <div style="font-size:18pt;background-color:#004040;color:#C0F0FF;width:150mm;">
          welcome to se <b><big>Dic3</big></b>
          <div style="height:7mm;"></div>
        </div>
        <div style="font-size:18pt;background-color:#E0F0F0;color:#000000;width:150mm;">
          <a href="zuig/Dic3UXh01.pdf">==> Dic3a PDF</a>
        <div>
        <div style="height:7mm;"></div>
        </ul>



</table><table border=1 width=100% style="background-color:#E0F0F0;">
<tr><td colspan=2>
        <div style="font-size:18pt;background-color:#004040;color:#C0F0FF;width:150mm;">
          welcome to se <b><big>Dic3</big></b>
          <div style="height:7mm;"></div>
        </div>
        <div style="font-size:18pt;background-color:#E0F0F0;color:#000000;width:150mm;">
          <a href="zuig/Dic3UXh01.pdf">==> Dic3a PDF</a>
        <div>
        <div style="height:7mm;"></div>


</table><table border=1 width=100% style="background-color:#E0F0F0;">
<tr><td colspan=2>
        <div style="font-size:18pt;background-color:#004040;color:#C0F0FF;width:150mm;">
          welcome to se <b><big>Dic3</big></b>
          <div style="height:7mm;"></div>
        </div>
        <div style="font-size:18pt;background-color:#E0F0F0;color:#000000;width:150mm;">
          <a href="zuig/Dic3UXh01.pdf">==> Dic3a PDF</a>
        <div>
        <div style="height:7mm;"></div>


        <div style="height:7mm;"></div>
          <div style="height:2mm;"></div>


          Es wird Zeit, dass Du einen ganzen Prozessor baust!
        <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                border:1px solid black;font: 500 8pt monospace;color:navy;">
ENTITY Prozessor IS
  PORT (
  Databus  : inout  STD_LOGIC_VECTOR(7 downto 0);
  Adressbus: out    STD_LOGIC_VECTOR(15 downto 0);
  clk      : in     STD_LOGIC;
```

```
  notWrite : out    STD_LOGIC;
  notRead  : out    STD_LOGIC;
  );
END Prozessor;
ARCHITECTURE Innenleben of Prozessor IS BEGIN
  ...
  ALU:process(clk) begin
  ...
  end ALU;
  CU: process(clk) begin
  ...
  end CU;
END Innenleben;
END Prozessor;
</pre>
          <div style="height:2mm;"></div>
          bitte fang damit schon einmal an!
          Vielleicht beginnst Du bei der <b>ALU</b>, des hast schon enmal gemacht:
          &Uuml;berlege, welche 'operations' Dein Prozessor haben soll.
        <div style="height:60mm;"></div>
        <hr>


</table><table border=1 width=100% bgcolor=#E0F0F0>
<tr><td colspan=2><div style="font-size:18pt;background-color:#004040;color:#C0F0FF;">
        welcome to se <b><big>Dic3</big></b>
        </div>
        <div style="height:7mm;"></div>
        <hr>
        <div style="height:7mm;"></div>




</table><table border=1 width=100% bgcolor=#E0F0F0>
<tr><td colspan=2><div style="font-size:18pt;background-color:#004040;color:#C0F0FF;">
        welcome to se <b><big>Dic3</big></b>
        </div>
        <div style="height:7mm;"></div>
        <hr>
        <div style="height:7mm;"></div>



</table><table border=1 width=100% bgcolor=#E0F0F0>
<tr><td colspan=2><div style="font-size:18pt;background-color:#004040;color:#C0F0FF;">
        welcome to se <b><big>Dic3</big></b>
        </div>
        <div style="height:7mm;"></div>
        <a href="zuig/Dic3UXh01.pdf">==> Dic3a PDF</a>
        <div style="height:7mm;"></div>
        <div style="font:500 12pt/120% times;background-color:#F0F0F0;color:#000000;padding:1mm 0mm 1mm 1m
          <big><u>Mo-30Nov'20:</u></big>
          <div style="height:3mm;"></div>
          wie immer ist h&auml;ndische Mitschrift zu f&uuml;hren
          <div style="height:5mm;"></div>
          <a href="#Mo30Nov" style="font:700 40pt serif;"> Link: --&gt;
                <br> --&gt; &Uuml;bung Mo-30Nov'20 &lt;--
                <br>--   ^   ^   ^   ^   ^   ^   ^     --
                <br>--   |   |   |   |   |   |   |     --</a>
        </div>
        <div style="height:30mm;"></div>
        <hr>




        <div style="height:97mm;"></div>
        <div style="font:500 12pt/120% times;background-color:#F0F0F0;color:#000000;padding:1mm 0mm 1mm 1m
          Fr-20Nov20 und
          Di-17Nov20:
          <br>wie immer ist h&auml;ndische Mitschrift zu f&uuml;hren
          <div style="height:5mm;"></div>
```

```
        </div><div style="font:500 12pt/120% times;background-color:#E0FFC0;color:#000000;padding:1mm 0mm
          Fallunterscheidungen
          <div style="height:5mm;"></div>
          <ul style="width:150mm;">
          <li>with/select
<pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
               border:1px solid black;font: 500 10pt monospace;color:navy;">
with &lt;auswahl-vector> select
ergebnis &lt;= &lt;Verkn&uuml;pfung_1> when &lt;auswahlwert_1> ,
           &lt;Verkn&uuml;pfung_2> when &lt;auswahlwert_2> ,
           &bull;&bull;&bull;
           &lt;Verkn&uuml;pfung_n> when others;</pre>
          <div style="height:2mm;"></div>
          <li>when/else
<pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
               border:1px solid black;font: 500 10pt monospace;color:navy;">
&lt;ergebnis> &lt;=   &lt;Verkn&uuml;pfung_1> when &lt;Bedingung_1>
           else &lt;Verkn&uuml;pfung_2> when &lt;Bedingung_2>
           &bull;&bull;&bull;
           else &lt;Verkn&uuml;pfung_n>;
</pre>
          <div style="height:2mm;"></div>
          <li>if/then: only in sequential environment! (lernen wir erst)
<pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
               border:1px solid black;font: 500 10pt monospace;color:navy;">
if    &lt;Bedingung_1> then &lt;Sequentielle Anweisungen 1>;
elsif &lt;Bedingung_2> then &lt;Sequentielle Anweisungen 2>;
elsif &lt;Bedingung_3> then &lt;Sequentielle Anweisungen 3>;
           &bull;&bull;&bull;
else  &lt;Sequentielle Anweisungen n>;
endif;</pre>
          <div style="height:2mm;"></div>
          <li>case/is: only in sequential environment! (lernen wir erst)
<pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
               border:1px solid black;font: 500 10pt monospace;color:navy;">
case &lt;testsignal> is
     when &lt;Wert_1> => &lt;Sequentielle Anweisungen 1>;
     when &lt;Wert_2> => &lt;Sequentielle Anweisungen 2>;
          &bull;&bull;&bull;
     when others => &lt;Sequentielle Anweisungen n>;
end case;
</pre>
          <div style="height:4mm;"></div>
          <li>ein <em>"sequential environment"</em> entsteht durch ein <em>PROCESS ... BEGIN ... END PROCE
            (is heute no ka Thema)
<pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
               border:1px solid black;font: 500 10pt monospace;color:navy;">
process &lt;empfindlichkeitsliste>
    -- hier k&ouml;nnen lokale Signale
       oder Variablen deklariert werden
begin
    -- hier ist ein 'sequential environment'
end process;</pre>
            (is heute no ka Thema)
          <div style="height:4mm;"></div>
          <li>Aufgabe: Schreib (in Dei Mitschrift)  VHDL mit
            <ul>
            <li>8-Bit-Inputs A, B
            <li>8-Bit-Output Y
            <li>einem 1-Bit-Input "AddSub"
            <li>wenn AddSub='0' gebe Y die Summe A+B aus,
              wenn AddSub='1' die Differenz A-B
            <li>mache die &Uuml;bung einmal mit 'with/select'
              und einmal mit 'when/else'
            </ul>
          <div style="height:4mm;"></div>
          <li>Abgabe: Di-17Nov20: keine
          <li>Abgabe: Fr-20Nov20: obgenannte Aufgabe   ---   Deadline: Samstag Abend
            <div style="font:300 10pt times;">"Du-u, XH, wann is Samstag Abend ??"
              --&gt; Geheimtip: wenns wieder hell wird, isser vorbei
            <br>"geht des no nach der Sommerzeit ?" --&gt; mit Sommerzeit wirds fiaher hell
            </div>
          </ul>
          <div style="height:2mm;"></div>
        </div>
        <div style="height:57mm;"></div>
```

```
        </div><div style="font:500 12pt/120% times;background-color:#FFE0C0;color:#000000;padding:1mm 0mm
          10Nov'20:
          <div style="height:5mm;"></div>
          <ul style="width:150mm;">
          <li>Anstelle von VHDL-Programmen wurden Gatterschaltungs-Zeichnungen angefertigt,
            die aber sehr wohl das Problem l&ouml;sten,
            nur sind sie kein VHDL
          <li>nun wieder VHDL:
          <br>das Grundger&uuml;st: Du hast
          <br>a) einen Header (so was wie diese '#include' in 'C')
              --&gt; das schreiben wir einfach jedes mal so ab
        <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                 border:1px solid black;font: 500 8pt monospace;color:navy;">
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;</pre>
            <div style="height:4mm;"></div>
            <br>b) eine Modul-Beschreibung ('Entity') mit Ein- und Ausg&auml;ngen:
        <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                 border:1px solid black;font: 500 8pt monospace;color:navy;">
ENTITY &lt;Modulname> IS PORT(
  ...
);
END &lt;Modulname>;</pre>
            <div style="height:4mm;"></div>
            und
            <br>c) mehrere Schaltungsbeschreibungen ('Architecture'):
        <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                 border:1px solid black;font: 500 8pt monospace;color:navy;">
ARCHITECTURE &lt;Schaltungsname> OF &lt;Modulname> IS
BEGIN
  ...
END &lt;Schaltungsname>;</pre>
            <div style="height:4mm;"></div>
            --&gt;
            <br>das Ganze schaut dann so aus:
        <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                 border:1px solid black;font: 500 8pt monospace;color:navy;">
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY &lt;Modulname> IS PORT(
  ...
);
END &lt;Modulname>;

ARCHITECTURE &lt;Schaltungsname> OF Modulname IS
BEGIN
  ...
END &lt;Schaltungsname>;</pre>
            <div style="height:10mm;"></div>
          <li>Ein - und Ausg&auml;nge:
          <br>Es gibt allerlei 'Datentypen', wir verwenden vorerst aber nur 'STD_LOGIC'.
          <br>einzelne Ein- und Ausgangsleitungen erzeugt man mit zB.:
        <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                 border:1px solid black;font: 500 8pt monospace;color:navy;">
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
  x   : IN  STD_LOGIC;
  y   : OUT  STD_LOGIC;
);
END reg8;</pre>
            <div style="height:4mm;"></div>
            ganze Leitungs-B&uuml;ndel erzeugt man mit 'vector':
            <br>(hier ein 8-faches B&uuml;ndel mit den Leitungen Nr.0 bis Nr.7)
            <br>---&gt: es ist <i>ratsam</i>, die Reihenfolge abw&auml;ts anzugeben,
              sonst verdreht es Dir sp&auml;ter bei Registern und Addierern die Bits
        <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                 border:1px solid black;font: 500 8pt monospace;color:navy;">
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
  d   : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
  q   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
);
END reg8;</pre>
            <div style="height:4mm;"></div>
            Mit 'in', 'out' (und 'inout') gibt man die Daten-Richtung an:
          <pre width=80 style="margin:0 0 0 1mm;padding: 3 3 3 4mm;background-color:beige;
                border:1px solid black;font: 500 8pt monospace;color:navy;">
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY reg8 IS PORT(
  x   : <big><b>IN</b></big>  STD_LOGIC;
  y   : <big><b>OUT</b></big>  STD_LOGIC;
);
END reg8;</pre>
            <div style="height:4mm;"></div>
            <div style="height:4mm;"></div>
          <li>&Uuml;bungsaufgabe:
            <br>Schreibe die 'PORT'-'ENTITY' f&uuml;r ein 16-Bit-Addierwerk in VHDL
          </ul>
```

13.3 Unterhaltung: der Brief

```
          \input{KWx/latex/modDic1/HDL/Dic3aMail-Vb51.txt}
Di26Jan21-0539:
zu
vorerst: XH mag keine Texte
==>oh!
   des ischa lange G'schicht.
   a Bissi ausfuehrlicher heisst des:
   Ingenieure, die wir ja werden wollen ina HTL,
   machen Planzeichnungen, Entwuerfe, Diagramme
   und keine Schriftsteller-Romane.
   Die Diplomarbeitskandidaten meinen immer,
   wenn sie 40 Seiten irgendwelchen Geschwafels aus Webseiten
   kopieren, seien sie fertig und Sehrgut. Aber sGegenteil
   isch der Fall.
   XH mag Planzeichnungen, Blockdiagramme, Top-Down-Designs, Schaltplaene,
   Zustandsdiagramme und -Tabellen, Aktivitaetsdiagramme, Sequenzdiagramme,
   xy-Graphen, Bodediagramme, SPICEnetzlisten, Designpatterns,
   Klassendiagramme, DB-Normalformen, Synapsenmatrizen und derlei
   Entwurfszeug,
   aber kein rechtschreibfehlerstrotzendes Bildzeitungsgefasel mit
   dem Formulierungsniveau von Volksschulaufsaetzen der Kategorie
   "mein schoenstes Ferienerlebnis".
   Techniker sein meist miese Schriftsteller, deswegen sollen sie
   auch Technik entwerfen und keine Romane schreiben.

nice time LGXH Di26Jan21-0539



26Jan21-0614
Ich bin bei der Programmierung in VHDL auf die Idee gekommen,
nur eine Zeilenänderung reichte um den  GOTO-Befehl einzufügen.
Dafür muss ich aber zum Verständnis noch einmal nachfragen:

* Zeigt der Data-Pointer in VN3 zeigt auf eine RAM-Adresse,
  welche von welcher wir Daten laden wollen? (Erhoffte Ant. "JA")

==> YESS!!!


* Das heißt der RAM erfährt über das Übertragen von Data-Pointer
  auf RAM-Adresse, welche Daten wir laden wollen und schreibt diese
  auf den DBus? (Erhoffte Antw. "JA")

==>YESS!!!


* Wie wird aber nun der Data-Pointer auf seinen Wert festgelegt?
```

```
==>in vorhergehenden Instruktionen der Art
   "copy RegisterXY to Datapointer"

   bei typischen 8-Bit-Maschinen wie zB. der "Z80" sein die
   CPU-Register 8 Bit breit, der Datapointer "HL" aber 16 Bit,
   weswegen dieser Datapointer auch 8-Bit-weise ansprechbar is
   "copy A nach H" oder "copy A nach L"
   dann erfolgt der Memory-Zugriff mit zB
   "LOAD Ram[HL] nach A" oder
   "SAVE B nach Ram[HL]"

   Die AVR-8-Bit-Microcontroller (wie diese TN-13A)
   haben gar keinen Datapointer, weil se gar keinen externen
   Daten- oder Adressbus haben (ma kann ka RAM anschliessen)

gm de LGXh 26Jan21-0614




Bis jetzt war der Data-Pointer für mich ein wunderbares Zauberinstrument.

==>Ja.
   Dessen Verwendung habi totgeschwiegen,
   damit mein Unterricht nicht no komplexer wird.
   Schliesslich willi ja eh schon die Themen
   'Prozessor' und 'VHDL' gleichzeitig durchnehmen.

   Wenn da nur die 1er-Kandidaten waeren,
   haettma ez schu die Diplomarbeit.
   iXH hoff nur, dass Dir die anderen Faecher auch so guat
   gelingen, wie des Prozessorzeug


Dieser könnte ja auch auf eine RAM-Adresse zeigen, welche ich in den PC einlesen kann.
Jedoch: Habe ich dann nicht zwingend, wenn ich es
so mache, eine Dauerschleife?

==>nicht, wenn alle Uebertragungen binnen einer einzigen Instruction
   ablaufen.
   Deswegen muessten wir unseren "CU: process"
   um weitere 'PhaseCTR'-Schritte aufblasen

   Des mochat aber mein Unterricht wieder komplizierter,
   und aus klaerender Uebung wird a Stress-Chaos.
   Des is mei Angst.

   Deswegen derzaehli, die CU hole in der DATA LOAD Phase
   die Sprungzieladresse und schreibe sie in der EXECUTE Phase
   in den ProgramCounter / InstPTR,
   was so gar nit stimmt.
   Aber meins is in VHDL halt simpler.

   Mir waer auch das Gefuehl der Teilnehmer "ich kann des,
   ich bin ProzessorDesigner" als Motivation wichtig.

   Genaugenommen derzaehli also an Shas zwecks Suggestion.
   Arschlochlehrer eben.

nice time, LGXh
---




LGXh Di26Jan21-0654:

Braucht es dann nicht eine seperate Logik,
die mir entweder den Goto-Befehl durch ein NOP tauscht,
oder gar die Sprungadresse im RAM ändert?

==>Du denksch da scho richtig und detailliert!
```

```
   Im Grunde lassma durch die Erweiterung
   des von-Neumann-Zyklus sowas wiea NOP mittendrin
   ablaufen, ja
   (so habiXH des no gar nie gsehen, aber stimmt -
    danke fuer die Bewusstseinserweiterung!)


* Verstehe ich das richtig,
  dass wir immer nur 16 Bit (Instruction+8BitDaten)/
  also 1 Register pro Durchlauf der 5 Phasen einlesen können?

==>YESS!!!
   aber:

==>Als Prozessordesigner kannsch Du Dei CPU so komplex machen,
   wie Du grod mogsch.
==>Obs "besser" isch, kleine, schnelle InstructionSets mit
   kurzen Instructions (RISC)
   oder
   viele komplexe, lange Instructions (CISC) zu realisieren,
   isa laufender wissenschaftlicher Richtungsstreit,
   der scho uebera halbs Jahrhundert lauft
   und sich wie die Kleidermode-Saisonen mal dahin
   und mal dorthin entwickelt.

   So nennt sich ja die Prozessor-Schmiede
   "ARM"-"Advanced RISC Machines"
   was decht irgendwie nach
   "komplizierte simple Prozessoren"
   klingt :-)))


> Das heißt eine Addition braucht 10 Clockzyklen?

==>schau mal ins AVR-Instruction set (DIC-XH-Webseite),
   do findesch Angaben zu Byte-Laengen und
   Zyklen-Bedarf
   und eben verschieden "lange" instructions
   je nachdem, wie oft sie ins (interne) RAM
   zugreifen



   Man!
   Du hasch sowas von Recht
   mit all Deine Vorstellungen,
   gewaltig!

==>schreib mal den David RIESER aus Deiner alten Klass an,
   der werkt schu laenger an einem eigenen Prozessordesign
   und stellt sehr aehnliche Fragen.

nice *greez* LGXh Di26Jan21-0654




LGXh Di26Jan21-0721:
Meine Codeänderung wäre nur...

                        elsif(PhaseCTR=3) then
                                case InstReg is
                                        when ... => A <= DBus;
                                        when ... => B <= DBus;
                                        when ... => PC <= (DBus-1);
                                end case;
==>mmmmmmmmmhhhjaaaahh...
   ...in der CU. (DBus-1, da wir ja am Anfang von der VN1 PC+1 machen,
==>(PC-1) is die Instruction, die wir gerade eben schon geladen haben,
   also die Instruction, die wir gerade ausfuehren,
   die schon im InstREG steht.

==>"PC zeigt immer auf die naxte Instruction"
   fuer die nexte kommende FETCH-Phase
   (genau bei JUMPs wirds sonsch komplex)
```

```
>sonst springen wir ja immer eine Adresse weiter als geplant

==>ja, des is auch wirklich so.
   ein Assembler oder Compiler muss des bei der Uebersetzung
   von "GOTO" und "if" beachten.

   ein JUMP +/-0 springt eins weiter, is also wirkungslos
   und ein JUMP -1 isa Endlosschleife.

   jaja.


>(Weitere Lösungen für dieses Problem = PC+1 im VN5-Ende

==>des PC+1 darf jedenfalls erscht nach dem FETCH erfolgen.


>oder Angeben, dass bei GOTO manuell immer das +1 beachtet werden muss,
 sozusagen ein 'GOAFTER'))

 ==>confirm.
    tunse auch (s.o.)


>Irgendwie kommt mir das jetzt aber doch mehr als dubios vor,

==>Haha, des geht mir schon auch so.
   Reine vonNeumann baut ja eh keiner,
   jeder machelt an eigenen Modifikationen umma.
   Was dann siegt, entscheiden leider nicht die technischen Eigenschaften:
   So hat sich aus der 80x86- / 68000- / NS32000- Aera
   nicht die eleganteste NS32032 durchgesetzt,
   sondern der kotzige 80x86 Sauhaufen und die zweitbesten
   68020-Nachfolger.


>ich weiß ja gar nicht,
 wie ich den Data-Pointer wirklich erzeuge.
 Ich weiß nur,
 dass damit (eig für die ALU) auf einen Ram-Teil
 über den Adressbus gezeigt wird.

 ==>cfm.
    is fuer die DATA LOAD und DATA SAVE Phasen gedacht


>- Knoble jetzt schon lange umher,
 wirkt aber alles nicht wirklich richtig
 was bis jetzt an Überlegungen gekommen ist

==>iXH hoffe, dass meiXhne Beitraege Deine Knoblerei
   noch genialer und undurchsichtiger machen
   :^))

nice Moin, ma trifftsi ez ja scho bald in Chat...
LGXh Di26Jan21-0721




02Feb'21:
>>
>> Und... kommt es nur mir so vor, oder ist Assembler eigentlich ziemlich einfach
>> (wenn man ein bisschen Logik versteht)?

==>des scheint eine Geschmacksfrage zu sein.
   Die Einen (zB. Du und iXH) sagen ganz klar "JAWOHL",
   und die Anderen "NIEMALS!"

   Da "Assembler" nit EINE Sprache is,
   sondern ein Sammelname fuer alle Prozessorarchitekturen.

   Kennen und gmacht haben tuis zerst noch auf der
```

```
    IBM-370/155 und -115
    (https://de.wikipedia.org/wiki/System/370),
    dann 8080, 8085, Z80, dann uC 8051, 6805, 80515, 80535
    bis schliesslich ab Ende 90erJahre auf den AVR-8Bit.

    Nach 2-taegigem Blick in ein 8086-Programm habimi wuergend
    den x86-Dingern in Assembler abgewendet.

    Die der x86 verwandte 8051 Architektur isch sehr unsymmetrisch,
    dh. jedes Register hat Spezialanwendungen: 'B' als Schleifenzaehler,
    'C' fuer Port-IO, 'D' und 'E' fuer allgemeine Adressierungen
    und 'HL' schliesslich als 16-Bit-DataPointer.
    An sich aber nicht nur unsympathisch,
    weil jeder Programmierer die Register gleich verwenden musste.
    Die dortige Instruction 'DJNZ' = 'Decrement and Jump if Not Zero'
    fuer Zaehlschleifen geht mir heute noch ab.

    Die AVR sind schoen symmetrisch - mit jedem Register kannst
    alles machen, da musst in der Registerverwendung selber a Bissl
    Disciplin halten, synsch kennschdi rasch in Dein eigenen
    Sauhaufen numma aus.

    Die sind schoen skalierbar - von 'klein und langsam'
    (TN11, -12, -13, -15, -25) bis 'gross und schnell'
    (wennma davon absieht, dass die AVR grundsaetzli weder
    gross no schnell sind) Mega328, -644, -2560 ua.
    v. 15Byte RAM bis 64kRAM, 1kPROM-256kPROM, 4MHZ-75MHz...
    Die voll vorteilhafte Verwendung derer 'SBIC', 'SBIS', 'SBRC'
    und 'SBRS' Instructions huni no nia richtig unter 'der Haut'

    Bei den 'PIC' (Microchip) hatma sehr viel mehr spezielle Ausfuehrungen
    (die AVR sind eher 'alle gleich')

    Mita 'Celeron' oder 'Itanium' is gonz a ondare Assemblerei
    als mita Tiny13.

==>mit 'insern' AVRtiny unterstreichi Dein 'ziemlich einfach'
   mita x86 sogi "dann machs bitte DU"


==>ba de "essbaren" uC und uP hosch 2 kotzige Bereiche:
   (1) Realzahlen-Rechnerei
   (2) Formatierte Ein-/Ausgabe
   Beides resultiert in eselslangen Unterprogrammen mit
   widerholend immer wieder fascht es Gleiche,
   nur kloane Unterschiede.
   Dann fangsche un kopieren und nachmodifizieren,
   dabei mochsch Fahlerlen
   und suachsche ewig.

   Wegen (1) und (2) hatma
      'C'
   erfunden,
   d.h.
   'C' isch fuer Assemblerdenker
   mit ohne (1) u. (2).
   (1) erledigt die 'arithmetic expression' und
   (2) die 'printf - "%"'-Formatierungen.
   (viel mehr wie des isch in 'C' gor nit drin :-)))

   Auch des anfaengliche FORTRAN war so. Dort hats
   statt 'printf' halt 'write' gheissen

==>wenn heid a Programmierer von zB. Java oder Python kimmt,
   der schreib gonza onders 'C', als Du und iXH,
   und er hasstz,
   waehrend mir jubeln.

   'schoen' is 'C' naemli NIT!
   Es is 'kurz' und 'praktisch'.

   zB. schreib er
      'while(TRUE)'
   und mir schreim
      'for(;;)'
```

```
   :-))

==>wennma vom ezigen Dic3-Thema 'AVR-Assembler' reden,
   sieXhis genau wia Du!
   (sorgfaeltig und genau sein muassma schun,
   isch Nix fuer Hudler)

nice nite LGXh
```

Crashkurs VHDL

FH München, FB 06

1 Einleitung

Die Hardwarebeschreibungssprache VHDL hat sich (Stand 2002) zum weltweiten Standard für den Entwurf digitaler Schaltungen entwickelt. Kenntnisse in VHDL entsprechen in ihrer Bedeutung in der Hardwareentwicklung in etwa Kenntnissen in der Programmiersprache „C“ in der Softwareentwicklung.
In diesem Kurs werden Ihnen die grundlegenden Konzepte eines Schaltungsentwurfs in VHDL erläutert. Dabei geht es ausschließlich um die Schaltungssynthese, der gesamte Bereich der Simulation wird ausgespart. Mit den vermittelten Kenntnissen können Sie zwar weder elegante Beschreibungen für kleine Schaltungen noch parametrisierbare komplexe Entwürfe formulieren– sie können aber kleinere digitale Schaltungen nach „Schema F“ in VHDL beschreiben.

Kursziele:

- Verständnis für grundlegende Konzepte in VHDL
- Sprachreferenz (Syntax) für wichtige Befehle
- Schemata für grundlegende Aufgabenstellungen (Kombinatoriken, Automaten)

Theoretische Voraussetzungen:

- Boolesche Algebra
- Automatenentwurf (Moore-Automat)

Bitte beachten Sie, dass die Programmiersprache, die Ihnen das Nachdenken abnimmt, noch nicht erfunden worden ist. Sie werden im Verlauf des Kurses feststellen, dass Ihnen VHDL teilweise sehr einfache und eingängige Beschreibungsmöglichkeiten bietet – sofern Sie wissen, was beschrieben werden soll! Verwenden Sie also bei einem neuen Entwurf zunächst einen guten Teil Ihrer Zeit darauf, Ihr Problem günstig zu strukturieren und die einzelnen Teilaufgaben sauber zu definieren. Scheuen Sie sich nicht, zunächst ganz ohne VHDL Papier zu benutzen um darauf Blockdiagramme für die Grobstrukturierung oder Zustandsdiagramme für Automaten zu zeichnen. Erst wenn Sie sicher sind, das Problem in sinnvolle Teile zerlegt zu haben und für jeden Teil wissen, was er genau leisten soll, denken Sie an die Umsetzung in VHDL. Bei diesem Vorgehen werden Sie feststellen, dass die reine Programmierung in VHDL zum Kinderspiel wird, da die einzelnen Teile einfach aufgebaut sind (und damit mit Methoden aus „Schema F“ abgedeckt werden können) und leicht unabhängig voneinander ausgetestet werden können.
Wenn Sie stattdessen ohne klares Ziel vor Augen sofort mit einem VHDL-Programm beginnen, so werden Sie nach aller Erfahrung sehr bald den Überblick verlieren und ständig neue Variablen und Verzweigungen einführen, um „merkwürdige“ (weil nicht bedachte) Effekte in der Schaltung zu eliminieren.
Das Resultat, sofern es eines gibt, ähnelt dann einer Bauruine, die nur mit Stützen und Streben notdürftig aufrechterhalten wird und vermutlich beim ersten Unwetter (sprich Praxiseinsatz) einstürzt.

2 Signale, Typen und Vektoren

Für die folgenden Erläuterungen soll als Beispiel die Schaltung in Abbildung 1 verwendet werden.

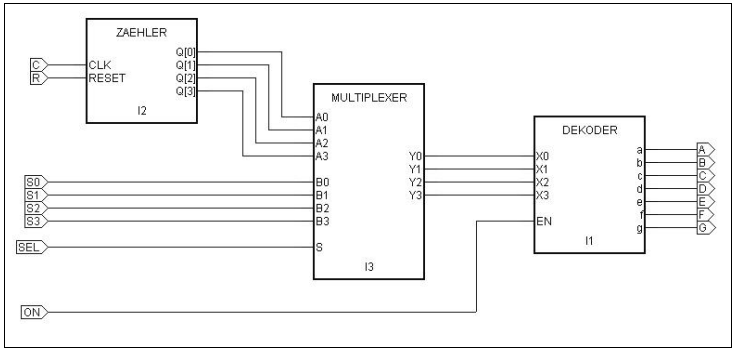


Abbildung 1: Blockschaltbild

Die Schaltung besteht aus drei **Blöcken** (Zaehler, Multiplexer, Dekoder), die untereinander durch **Signale** verbunden sind. Diese Darstellung entspricht dem Blockdiagramm auf Papier zur Aufteilung des Gesamtproblems in kleinere Teilprobleme. Die Gesamtschaltung hat die Aufgabe, entweder einen Zählerstand (0 bis 9) oder eine direkte Zahleneingabe (0 bis 9) an den Schaltern S0 bis S3 in Abhängigkeit von einem Wahlschalter SEL auf einer Siebensegmentanzeige auszugeben. Die Siebensegmentanzeige kann über einen Schalter ON ein- bzw. ausgeschaltet werden. Der Zähler kann mit zwei Tastern C und R entweder hochgezählt oder auf Null zurückgesetzt werden.
Diese Darstellung eignet sich für eine weiteren Entwurf in VHDL ideal, da in VHDL ebenfalls **Blöcke** beschrieben werden, die durch **Signale** untereinander verbunden sind.

Sofern in den Beispielen VDHL-Code verwendet wird, werden Schlüsselworte fett geschrieben oder farbig markiert.

2.1 Konventionen in VHDL

Vorab einige wichtige Regeln in VHDL, die Erläuterung der Beispiele folgt später.

2.1.1 Namensregeln

Für alle folgenden Ausführungen und natürlich den eignen Entwurf müssen Sie bei der Wahl von Bezeichnern (Namen von Signale etc.) beachten:

- Zwischen Groß- und Kleinschreibung wird in VHDL nicht unterschieden
- Namen müssen mit einem Buchstaben beginnen
- Schlüsselworte sind nicht als Namen erlaubt.

Da Sie nicht alle Schlüsselworte kennen können, sollten Sie entweder einen Editor verwenden, der VHDL-Schlüsselworte erkennt und von sich aus bereits hervorhebt oder in weiser Voraussicht verdächtige Namen wie „and“, „else“ oder „begin“ von sich aus vermeiden.

2.1.2 Kommentare

Kommentare werden durch einen doppelten Bindestrich eingeleitet und gelten bis zum Zeilenende (Abbildung 2).

```
-- Dieser Block enthält einen einfachen 2_zu_1 Multiplexer

entity multiplexer is
port
(
  S:        in bit;     -- Steuereingang zur Signalauswahl
  A, B:     in bit;     -- Dateneingänge
  Y:        out bit     -- Datenausgang
);
end;
```

Abbildung 2: Kommentare

2.1.3 Zuweisungen

Zuweisungen eines Signals an ein anders Signal bzw. eines konstanten Wertes an ein Signal werden in VHDL von rechts nach links vorgenommen und durch die Zeichenkombination <= dargestellt (Abbildung 2).

```
Y <= S;           -- S wird nach Y kopiert
Y <= A or B;      -- A und B werden mit “oder” verknüpft und das
                  -- Ergebnis wird Y zugewiesen
A => Y;           -- so rum geht es NICHT!
```

Abbildung 3: Zuweisungen

2.2 Typologie

VHDL ist eine streng typgebundene Sprache, d.h., jedem verwendeten Signal (bzw. Variablen) muss ein Typ zugewiesen werden. Eine implizite Typkonversion wie in „C“ oder einen Universaltyp wie „void“ gibt es nicht. Für den Entwurf genügt die Kenntnis der folgenden drei Typen: **boolean**, **bit** und **std_logic**.

Typ	**Wertevorrat**	**Verwendung**
boolean	true, false	logische Abfragen (if)
bit	0, 1	Entwurf
std_logic	0, 1, Z, -, U, L, H, X, W	Entwurf und Simulation

Tabelle 1: Standardtypen

Eine Deklaration eines Signals mit einem Typ sieht in VDHL wie in Abbildung 4 gezeigt aus.

Syntax:
signal <signalnamen>: typ;

Beispiele:

```
signal X0, X1, X2, X3: bit;   -- vier Signale X0 bis X3 vom Typ bit
signal EN: std_logic;         -- signal EN hat den Typ std_logic
signal ein_aus: boolean;      -- ein boolesches Signal ein_aus
```

Abbildung 4: Signaldeklarationen

Bei Verknüpfungen müssen die einzelnen Signale vom gleichen Typ sein. Das Ergebnis kann natürlich auch nur einem Signal vom gleichen Typ zugewiesen werden. Einzelne konstante Werte der Typen bit und std_logic müssen in Hochkommas eingeschlossen werden (Abbildung 5).

```
X0 <= ´0´;              -- X0 ist konstant 0
EN <= ´Z´;              -- EN wird Z zugewiesen (hochohmig)
ein_aus <= true;        -- ein_aus ist wahr
X1 <= ´Z´;              -- geht nicht, weil der Typ bit den Wert
                        -- Z nicht kennt
ein_aus <= ´1´;         -- geht auch nicht, der Typ boolean den
                        -- Wert 1 nicht kennt (1 ist nicht true)

X0 <= (´1´xor X1) and X2 or not X3;      -- geht problemlos
X0 <= X1 or EN;         -- geht nicht, da verschiedene Typen
ein_aus <=(X0=X1);      -- das geht, weil zuerst X0 mit X1 verglichen
                        -- wird und das Ergebnis eines Vergleichs wahr
                        -- oder falsch ist. Dieser boolesche Wert kann
                        -- dann auch nur einem Signal vom typ boolean
                        -- zugewiesen werden!
```

Abbildung 5: Typprüfung bei Zuweisungen

Die neun Werte des Typs std_logic sind in Tabelle 2 kurz erklärt.

Wert	Bedeutung	Verwendung
0	“starke” log. Null	Synthese, Simulation (wie 0 im Typ bit), technisch Push/Pull
1	„starke“ log. Eins	Synthese, Simulation (wie 1 im Typ bit), technisch Push/Pull
Z	hochohmig	Synthese, Simulation, technisch Tristate-Buffer, z.b. Bus
-	don’t care	Synthese, der Wert ist egal; speziell bei Wertetabellen
U	unbekannt	Simulation, der Wert ist nicht bekannt
X	Konflikt	Simulation, Erkennung von 0 gegen 1 - Treiberkonflikten
L	„schwache“ 0	Synthese, technisch open source (offener Emitter)
H	„schwache“ 1	Synthese, technisch open drain (offener Kollektor)
W	„schwaches“ X	Simulation, Erkennung von L gegen H - Treiberkonflikten

Tabelle 2: Wertevorrat in std_logic

Für den Schaltungsentwurf sind neben den Werten ´0´ und ´1´ noch speziell die Werte ´Z´ und ´-´ interessant.

2.2.1 Wert ´Z´

Der Wert ´Z´ kann einem Signal zugewiesen werden.
In der technischen Schaltung entsteht bei der **Zuweisung** ein Tristate-Buffer. Solange der Wert ´Z´ zugewiesen ist, wird der Buffer in den hochohmigen Zustand geschaltet. Wird dagegen (zu einem anderen Zeitpunkt) diesem Signal der Wert ´0´ oder ´1´ zugewiesen, dann wird der Buffer automatisch wieder auf Gegentaktbetrieb umgestellt und der zugewiesene Wert ausgegeben. Damit können also Bustreiber, die einen Tristatebetrieb erfordern, modelliert werden.
Aber Achtung: Wenn die Zielschaltung, beispielsweise ein CPLD oder FPGA, keine Tristate-Buffer bereitstellt, kann diese Methode natürlich nicht verwendet werden!
Die **Abfrage** eines Signals auf den Wert ´Z´ ist zwar in der Simulation sinnvoll, in der Synthese jedoch nicht.

2.2.2 Wert ´-´

Der Wert ´-´ kann ebenfalls einem Signal zugewiesen werden.
Bei der **Zuweisung** sucht sich das Synthesewerkzeug selbständig einen Wert (´0´ oder ´1´) aus, den das Signal dann in der technischen Schaltung führt. Welcher von beiden Werten das ist, wissen Sie nicht – sie haben ja angegeben, dass Ihnen das auch egal ist.
Bei der **Abfrage** würde man sich erhoffen, dass das Synthesewerkzeug selbständig eine Minimierung vornimmt. Das ist jedoch nicht der Fall; vielmehr müsste ein technisch nicht vorhandener Wert ´-´ verarbeitet werden können. Die Abfrage dieses Wertes ist also zwar möglich, wird aber in aller Regel weder in der Simulation noch in der Synthese zu den erwarteten Ergebnissen führen!

2.3 Vektoren

Eine wesentliche Erleichterung, die VHDL bietet, ist die Zusammenfassung von Signalen gleichen Typs zu Vektoren. Als Vektor wird dabei ein eindimensionales Array (Feld) verstanden. Der Laufindex des Vektors ist eine Integerzahl. Obwohl es mit dem Kommando **array** in VHDL prinzipiell möglich ist, selber beliebige Felder zu definieren, sind für die Standardtypen bit und std_logic entsprechende Felder als Typen bereits vordefiniert. Sie sollten diese vordefinierten Typen verwenden, weil diese Typen als Dualzahlen interpretiert werden können und Berechnungen und Vergleiche damit durchgeführt werden können.

2.3.1 Deklaration

Eine Deklaration eines Vektors sieht wie in Abbildung 6 gezeigt aus:

Syntax:
signal <signalnamen>: typ(<lower> to <upper>);
signal <signalnamen>: typ(<upper> downto <lower>);

Beispiele:

```
signal x: bit_vector(0 to 7);          -- acht Signale vom Typ bit: x(0), x(1), x(2), …,x(7)
signal a: std_logic_vector (2 to 4);   -- drei Signale vom Typ std_logic: a(2), a(3), a(4)
signal r: bit_vector(3 downto 0);      -- vier Signale vom Typ bit: r(0), r(1), r(2), r(3)
```

Abbildung 6: Syntax Vektordeklaration

Die Namen der Einzelsignale werden aus dem Signalnamen und dem Laufindex in runden Klammern gebildet. Der Signalvektor ´signal ena_2: bit_vector(3 to 5)´ besteht also aus den Einzelsignalen ena_2(3), ena2_(4) und ena_2(5). Die Einzelsignale ´e1, e2 e3' können dagegen nicht als Vektor 'e': geschrieben werden, da hier die runden Klammern fehlen!

2.3.2 Zuweisungen und Verknüpfungen

Mit Vektoren ist es besonders einfach, mit nur einer Anweisung parallel eine Operation auf alle Einzelsignale (oder eine Teilmenge davon) auszuführen Einige Beispiele zeigt Tabelle 3.

```
signal a,b,c: bit_vector(0 to 3);  -- Deklaration dreier Vektoren

c <= a or b;                       -- bitweises oder auf alle Elemente
c <= ('1','0','0','0');            -- Zuweisung als Aggregat
c <= "1000";                       -- Zuweisung als Bitstring
c <= x"A";                         -- Zuweisung als Hexadezimalzahl
c <= a(0) & "001";                 -- Zusammensetzen mit &
c(1 to 2) <= "11";                 -- Auswahl eines Teilvektors im Ziel
c <= a(0 to 1) & '1' & b(0);       -- Auswahl eines Teilvektors in der Quelle
(w,x,y,z) <= c;                    -- Zuweisung von c an Einzelsignale w,x,y,z
```

Ziel				VHDL-Kommando
c(0)	c(1)	c(2)	c(3)	
a(0) or b(0)	a(1) or b(1)	a(2) or b(2)	a(3) or b(3)	c <= a or b;
1	0	0	0	c <= ('1', '0', '0', '0');
1	0	0	0	c <= "1000";
1	0	1	0	c <= x"A";
a(0)	0	0	1	c <= a(0) & "001";
unverändert	1	1	unverändert	c(1 to 2) <= "11";
a(0)	a(1)	1	b(0)	c <= a(0 to 1) & '1' & b(0);

Tabelle 3: Arbeiten mit Vektoren

Beachten Sie dabei besonders, dass Sie sich sowohl aus dem Ziel als auch aus der Quelle jeweils Teile herausgreifen können und dass Sie Teile mit ‚&' zu breiteren Vektoren zusammensetzen können. Sie müssen lediglich darauf achten, dass die Gesamtbreite von Ziel und Quelle bzw. Quellen jeweils gleich ist. Speziell bei der Zuweisung konstanter Werte sehen Sie, dass Sie auch eine Darstellung als Hexadezimalzahl mit x"zahl" bzw. als Oktalzahl mit o"zahl" wählen können. Die angegebene Zahl wird dabei in einen Vektor mit binären Einzelwerten umgerechnet.
Sie können bei der Zuweisung konstanter Werte den Unterstrich zur besseren Kennzeichnung von zusammengehörenden Gruppen verwenden. Er hat aber sonst keine Bedeutung, d.h. er vertritt keine Position im Vektor. Falls Sie in einem Vektor alle bisher nicht belegten Positionen (d.h. Einzelsignale) mit demselben Wert belegen wollen, dann können Sie das Kommando (others => 'wert') verwenden (Abbildung 7).

```
signal c: bit_vector(0 to 7);

c <= "01001001";        -- diese und die beiden folgenden
c <= "0100_1001";       -- Zuweisungen haben gleiche Wirkung
c <= x"49";

c <= (others => '0');   -- allen Einzelsignalen von c wird '0'
-- zugewiesen
```

Abbildung 7: Zuweisung an Vektoren

2.3.3 Positionsbestimmung im Vektor

Die Anordnung der Einzelsignale im Vektor geschieht immer von links nach rechts. Bei der Zuweisung bzw. bitweisen Verknüpfung von Vektoren werden immer die Einzelsignale an gleicher Position miteinander verknüpft. Bei der Deklaration des Vektors kann mit Hilfe der Schlüsselworte to bzw. downto festgelegt werden, ob das am weitesten links stehende Einzelsignal die niedrigste bzw. höchste Nummer erhält.
Der Unterschied in beiden Deklarationsvarianten kommt vor allem dann zum Tragen, wenn der Vektor nicht mehr als ein Feld von Einzelsignalen sondern als Dualzahl interpretiert wird.

```
signal x: std_logic_vector(0 to 3);
signal y: std_logic_vector(3 downto 0);
x <= x"3";                  -- Zuweisung erfolgt nach Position
y <= x"3";                  -- Zuweisung erfolgt nach Position
r <= x+y;                   -- Arithmetik erfolgt nach Stellenwertigkeit
```

Vektor x				**Vektor y**				**Anweisung**
x(0)	x(1)	x(2)	x(3)	y(3)	y(2)	y(1)	y(0)	
0	0	1	1					x <= x ´´3´´ ;
				0	0	1	1	y <= x ´´3´´ ;
Stellenwertigkeit				**Stellenwertigkeit**				**Interpretation**
2^0	2^1	2^2	2^3	2^3	2^2	2^1	2^0	
0	0	1	1					x = 4+8 = 12
				0	0	1	1	y = 2+1 = 3

Tabelle 4: Position und Stellenwertigkeit

Das Beispiel nach Tabelle 4 zeigt, dass die Variante downto der gewohnten Stellenwertigkeit für Dualzahlen entspricht. Wenn Sie also Standardarithmetik auf Vektoren betreiben wollen, dann sollten Sie die Vektoren mit downto deklarieren.

3 Aufbau einer Schaltungsbeschreibung

Sie kennen jetzt die Bedeutung von Typen, können Signale in VHDL deklarieren und sie entweder einzeln oder als Vektoren bearbeiten. In diesem Kapitel lernen Sie den grundsätzlichen Aufbau einer Schaltungsbeschreibung, speziell die Beschreibung eines einzelnen Blocks, kennen. Verwenden Sie am besten für jeden Block eine eigene Datei und benennen Sie die Datei so, dass Sie die darin enthaltene Schaltung auch wiedererkennen. Die Datei besteht aus drei getrennten Abschnitten

- Header mit Bibliotheks- und Packageeinbindungen
- Entity für die Schnittstellendefinition
- Architecture für die Funktionsbeschreibung.

3.1 Header

In diesem Teil legen Sie die Definitionen fest, die für die folgende Schaltungsbeschreibung gelten sollen. Da diese Definitionen nur für die unmittelbar folgende Schaltung (definiert durch ihre Schnittstelle) gilt, müssten Sie den Header vor der nächsten Schaltung in derselben Datei wiederholen. Da Sie ja in jeder Datei nur eine Schaltung beschreiben, stellt sich das Problem für Sie nicht ...
Der Header sieht so gut wie immer wie in Abbildung 8 gezeigt aus.

```
library ieee;                          -- die verwendete Bibliothek
use ieee.std_logic_1164.all;           -- die Definition der Grundtypen und Werte
use ieee.std_logic_unsigned.all;       -- die Definition einer unsigned-Arithmetik
```

Abbildung 8: Typischer Header

Sofern Sie keine Arithmetik auf Dualzahlen betreiben, können Sie die dritte Zeile weglassen. Diese Art der Arithmetik funktioniert nur auf Vektoren des Typs std_logic und interpretiert die Zahlen als vorzeichenlos. Es gibt auch andere Arithmetiken, die teilweise mehr leisten, aber auch herstellerspezifisch sind (speziell Synopsys). Die angegebene Arithmetik ist standardisiert und funktioniert mit allen standardkonformen Syntheseprogrammen.
Der Header ist in „C“ vergleichbar mit #include-Anweisungen zu Beginn eines Programms.

3.2 Die Schnittstelle – Entity

Der nächste Abschnitt ist die Definition der Schnittstelle der Schaltung. An dieser Stelle wird der Name der Blockes, der die Schaltung repräsentiert, definiert sowie sämtliche Signale, die an diesen Block angeschlossen sind. Dieser Block ist in „C“ vergleichbar mit der Deklaration einer Funktion. Auch hier wird der Funktionsname sowie die Typen und Zahl der übergebenen Parameter definiert.
An einem Block gibt es Eingangssignale, Ausgangssignale sowie Signale, die ihre Richtung wechseln können.

Signalrichtung	VHDL-Schlüsselwort	Bemerkung
Eingang	in	kann nur gelesen werden (Quelle)
Ausgang	out	kann nur geschrieben werden (Ziel)
Bidirektional	inout	kann sowohl gelesen als auch beschrieben werden

Tabelle 5: Signalrichtungen

Die formale Syntax für die einfache Deklaration einer Schnittstelle (entity) zeigt Abbildung 9.

```
entity <blockname> is
port
(
  <signalnamen>: <richtung> <typ>;  -- bei weiteren Signalen: ´;´
  <signalnamen>: <richtung> <typ>   -- beim letzten Signal KEIN ´;´
);                                  -- dafür nach der Klammer der ´;´
end;
```

Abbildung 9: Rahmen einer Entity

Mit dieser Kenntnis können Sie jetzt die Schnittstelle des Multiplexers aus Abbildung 1 wie in Abbildung 10 gezeigt deklarieren.

```
entity MULTIPLEXER is
port
(
  A0, A1, A2, A3: in bit;      -- Datenwort A
  B0, B1, B2, B3: in bit;      -- Datenwort B
  S:              in bit;      -- Steuereingang zur Auswahl (0: A, 1: B)
  Y0, Y1, Y2, Y3: out bit      -- Datenausgang
);
end;
```

Abbildung 10: Deklaration einer Entity

Bitte beachten Sie zunächst, dass der Blockname exakt dem Namen im Schaltplan entsprechen muss. Das betrifft auch die Groß-/Kleinschreibung. Innerhalb einer reinen VHDL-Beschreibung wäre dies zwar nicht erforderlich; wenn Sie aber den Block in einer anderen Umgebung (wie dem Schaltplan) wiedererkennen wollen, dann kommt es ja auch darauf an, ob das andere Programm eine Unterscheidung zwischen Groß- und Kleinschreibung vornimmt. Um mögliche Fehler auszuschließen verwenden Sie daher gleiche Schreibweisen. Dies betrifft natürlich auch die Signalnamen!
Weiterhin sehen Sie, dass Sie mehrere Signale vom gleichen Typ auch zusammen deklarieren können. Es empfiehlt sich, die Signale bei der Deklaration sinnfällig zu gruppieren.
Des weiteren fördert es die Lesbarkeit, wenn Sie Einrückungen verwenden, um den Gültigkeitsbereich eines Konstrukts (hier ‚port') zu verdeutlichen.
In diesem Beispiel können Sie die Signale nicht als Vektor deklarieren, weil ja die runden Klammern nicht im Signalnamen auftauchen. Für den Block „Zaehler" ist das aufgrund der gewählten Signalnamen aber möglich (Abbildung 11).

```
entity ZAEHLER is
port
(
  CLK:   in bit;                        -- Taktsignal
  RESET: in bit;                        -- asynchroner Reset auf Null
  Q:     out bit_vector(3 downto 0)     -- Ausgabe
);
end;
```

Abbildung 11: Entitydeklaration mit Vektor

Die in der entity deklarierten Signale sind sowohl außerhalb des Blockes sichtbar (das ist für die Verbindung zu anderen Blöcken im Schaltplan wichtig) als auch innerhalb der nun folgenden Schaltungsbeschreibung. Auch in „C" sind ja die übergebenen Parameter unmittelbar, d.h. ohne weitere Deklaration, in der Funktion verwendbar. Im Beispiel (Abbildung 1) sehen Sie, dass das Ausgangsignal Q(0), das in der entity ZAEHLER deklariert ist, mit dem Eingangsignal A0 der entity MULTIPLEXER im Schaltplan verbunden ist. Die beiden entities in den beiden VHDL-Dateien brauchen dazu nichts voneinander zu wissen.

3.3 Die Funktion - Architecture

Der letzte Abschnitt ist die Beschreibung der Funktion des eben definierten Blocks. Es kann durchaus für eine entity mehrere verschiedene Funktionsbeschreibungen geben. Für einen Zähler gäbe es die Möglichkeit, ihn als Synchron- bzw. Asynchronzähler auszuführen. Je nach Bedarf könnte dann eine der beiden Möglichkeiten für einen bestimmten Entwurf verwendet werden. Zur Identifizierung muss daher auch die Funktionsbeschreibung einen eindeutigen Namen erhalten. In diesem Kurs wird immer nur eine Beschreibung pro entity verwendet und sie heißt immer 'verhalten'.
Die formale Syntax für die einfache Deklaration einer Funktionsbeschreibung (architecture) sieht wie folgt aus:

```
architecture <beschreibungsname> of <blockname> is
 -- hier können lokale Signale deklariert werden
begin

 -- hier steht die Funktionsbeschreibung (nebenläufig)

end;
```

Abbildung 12: Rahmen einer Architecture

Damit ist der größte Teil der Funktionsbeschreibung des Multiplexers bereits verständlich (Abbildung 13).

```
architecture verhalten of MULTIPLEXER is
  signal a,b,y: bit_vector (0 to 3);  -- Hilfssignal als Vektor

begin
  a <= (a0,a1,a2,a3);          -- Zuweisung der Eingangssignale an den Vektor A
  b <= (b0,b1,b2,b3);          -- Zuweisung der Eingangssignale an den Vektor B

  y <= a when (s='0') else b; -- Auswahl durch S

  y0 <= y(0); y1 <= y(1);      -- Zuweisung des Ergebnisses an die Ausgangssignale
  y2 <= y(2); y3 <= y(3);
end verhalten;
```

Abbildung 13: Funktion des Multiplexers

Zunächst sehen Sie, dass der Blockname genau wie in der entity deklariert lautet. In der zweiten Zeile werden lokale Hilfssignale definiert. Da der Multiplexer ja vier Signalpaare jeweils völlig identisch behandelt, ist eine Vektordarstellung dieser Signale angebracht.
In den ersten beiden Zeilen nach dem 'begin' werden zunächst die Vektoren a und b mit den entsprechenden Signalen aus der Schnittstellendeklaration belegt.
Anschließend erfolgt in Abhängigkeit vom Wert von s die bedingte Zuweisung an den internen Ergebnisvektor y. Dieses Kommando wird später erklärt.
Zuletzt werden die Signale y0 bis y3 aus der Schnittstellendeklaration wieder aus dem internen Vektor y gebildet.
Natürlich hätte der Multiplexer auch gleich mit Signalnamen a(0), .. a(3), die sich in VHDL als Vektor darstellen lassen, definiert werden können. In diesem Fall hätten die Deklaration der internen Vektoren a, b und y sowie die einleitenden und abschließenden Umkopieraktionen vollständig entfallen können. Dies ist hier nur aus Demonstrationsgründen nicht erfolgt.

4 Nebenläufige und sequentielle Umgebungen

Ein wesentlicher Unterschied zwischen VHDL und einer normaler rein sequentiellen Programmiersprache wie „C“ besteht darin, dass Anweisungen in VHDL in der Regel nebenläufig, d.h. parallel abgearbeitet werden. Das liegt daran, dass aus der Beschreibung eine Schaltung erzeugt wird, deren einzelne Bestandteile (Gatter) ebenfalls ständig parallel arbeiten. Sie müssen sich bei der Beschreibung in VHDL zu jedem Zeitpunkt darüber klar sein, ob Sie sich derzeit in einer nebenläufigen oder einer sequentiellen Umgebung befinden.
Eine Umgebung ist dabei zwischen einem ’begin’ und dem zugehörigen ’end’ definiert. Die erste, von der architecture bereits eröffnete, Umgebung ist nebenläufig.
VHDL unterstützt Sie bei der Unterscheidung zwischen nebenläufigen und parallelen Umgebungen dadurch, dass die meisten Kommandos nur in einer der beiden Umgebungen erlaubt und bei illegaler Verwendung solcher Kommandos Fehlermeldungen erzeugt werden.

4.1 Nebenläufige Umgebung

In dieser Umgebung werden alle Kommandos bzw. zu Blöcken zusammengefassten Kommandos parallel ausgeführt. Die Reihenfolge, in der Anweisungen in der Umgebung stehen, hat keinen Einfluss auf das Ergebnis! Sehen Sie sich dazu den Code in Abbildung 14 an, bei der die Reihenfolge der Zeilen gegenüber Abbildung 13 umgestellt worden ist.

```
architecture verhalten of MULTIPLEXER is
  signal a,b,y: bit_vector (0 to 3);  -- Hilfssignal als Vektor

begin
  a <= (a0,a1,a2,a3);          -- Zuweisung der Eingangssignale an den Vektor A
  b <= (b0,b1,b2,b3);          -- Zuweisung der Eingangssignale an den Vektor B
  y0 <= y(0); y1 <= y(1);      -- Zuweisung des Ergebnisses an die Ausgangssignale
  y2 <= y(2); y3 <= y(3);

  y <= a when (s='0') else b; -- Auswahl durch S
end verhalten;
```

Abbildung 14: Nebenläufigkeit I (erlaubt)

Da die erste Umgebung in einer Architecture nebenläufig ist, spielt die Reihenfolge, in der Anweisungen aufgeschrieben werden, keine Rolle für die Funktion. In einer sequentiellen Programmiersprache wäre die Berechnung von y (letzte Zeile) **nach** der Verwendung in der dritten und vierten Zeile doch etwas merkwürdig.
Die Nebenläufigkeit führt allerdings auch dazu, dass die Zuweisung verschiedener Werte an ein- und dasselbe interne Signal illegal ist.

	VHDL	C
1	`signal x0,x1,a,b,y1,y2: bit;`	`int a,b,c;`
2	`c  <= a and b;`	`c = a + b;`
3	`y1  <= x0 when (c = ‘0’) else x1;`	`if (c>1) then …`
4	`c  <= a or b;`	`c = a-b;`
5	`y2  <= x0 when (c = ‘0’) else x1;`	`if (c>0) then …`

Abbildung 15: Nebenläufigkeit II (verboten)

Die Idee ist in beiden Fällen, ein Hilfssignal ’c’ zur leichteren Lesbarkeit vor der Auswertung zu berechnen (Zeilen 2 und 4). In „C“ ist das auch kein Problem, da die Anweisungen ja nacheinander abgearbeitet werden. In einer nebenläufigen VHDL-Umgebung werden aber alle Zeilen, speziell also auch die beiden Zuweisungen an dasselbe Signal ’c’, gleichzeitig ausgeführt. Damit entsteht ein Konflikt, da ’c’ nicht zur gleichen Zeit das Ergebnis der beiden Verknüpfungen von ’a’ und ’b’ sein kann.

4.2 Sequentielle Umgebung

4.2.1 Der Prozess

Die einzige sequentielle Umgebung, die in VHDL existiert, wird durch einen Prozess definiert (Abbildung 16).

```
process <empfindlichkeitsliste>
 -- hier können lokale Signale oder Variablen deklariert werden
begin
 -- hier ist eine sequentielle Umgebung
end process;
```

Abbildung 16: Prozess, Syntax

Die Anweisungen werden in der sequentiellen Umgebung nacheinander abgearbeitet. Wenn das Prozessende erreicht ist, startet die Ausführung sofort wieder am Prozessbeginn.
Eine parallele Umgebung kann auch mehrere Prozesse beinhalten. Diese Prozesse werden dann alle parallel ausgeführt, wobei die Anweisungen innerhalb eines Prozesses nacheinander abgearbeitet werden. Prozesse werden sehr häufig für die Beschreibung sequentieller Schaltungen (Automaten) verwendet. Ein Moore-Automat besteht ja definitionsgemäß aus zwei Schaltnetzen (Zustandsübergangsfunktion und Ausgabefunktion) sowie dem Zustandsspeicher. Hier bietet es sich geradezu an, die beiden Schaltnetze nebenläufig zu beschreiben und den sequentiell arbeitenden Speicher innerhalb derselben architecture mit einem Prozess zu beschreiben.
Die Empfindlichkeitsliste soll alle Signale enthalten, die innerhalb des Prozesses zu einer Änderung führen können.
Die lokal definierten Signale und Variablen sind grundsätzlich nur innerhalb des Prozesses sichtbar. Natürlich muss der Prozess auch Ausgangssignale an seine Umgebung zur weiteren Auswertung weiterleiten können. Im Beispiel des Moore-Automaten wäre das der aktuelle Zustand. Dies kann nur über Signale erfolgen, die außerhalb des Prozesses deklariert sind. Dabei muss beachtet werden, dass dann Zuweisungen an diese Signale nur im Prozess erlaubt sind. Das wird offensichtlich, wenn man sich überlegt, dass die Signalzuweisungen zwar in einem Prozess nacheinander ablaufen, diese aber mit allen nebenläufigen Anweisungen außerhalb des Prozesses konkurrieren. Betrachten Sie das Beispiel in Abbildung 17.

```
architecture verhalten of ZAEHLER is
  signal qint: std_logic_vector(3 downto 0);

begin

  process (reset, clk)                -- Prozess hängt von clk und reset ab
  begin
    if (reset='0') then qint <= x"0"; -- Asynchrones Rücksetzen auf Null
    elsif (clk='1') and clk'event     -- sonst warten auf Taktflanke
    then
      qint <= qint+1;                 -- Zählerstand hochzählen
    end if;
  end process;

  q <= qint;                          -- Nebenläufige Ausgabe

end;
```

Abbildung 17: Zähler als Prozess

Es handelt sich um eine Funktionsbeschreibung des Zählers aus Abbildung 1. Innerhalb der nebenläufigen Umgebung der Architecture laufen ein Prozess und eine Zuweisung (letzte Zeile) parallel. Sie können den Zähler als einen Moore-Automaten betrachten, hier wäre die Ausgabefunktion der Zustandsvektor ’qint’ selbst. Der Prozess selbst reagiert sowohl auf den Takt ’clk’ als auch das asynchrone Rücksetzsignal ’reset’. Daher stehen diese beiden Signale

in der Empfindlichkeitsliste. Innerhalb des Prozesses wird eine bedingte Zuweisung (Syntax wird später erklärt) ausgeführt. Falls 'reset' aktiv ist wird der Zählerstand sofort zurückgesetzt. Ansonsten wird auf eine steigende Taktflanke gewartet. Trifft diese ein, dann wird der Zählerstand um eins erhöht. Da hier das Prozessende erreicht ist, beginnt die Ausführung wieder am Prozessbeginn. Damit wird wieder das Signal 'reset' abgefragt bzw. auf die nächste steigende Taktflanke gewartet.

4.2.2 Signalzuweisung innerhalb und außerhalb des Prozesses

Betrachten Sie jetzt den Code in Abbildung 18.

```
architecture verhalten of ZAEHLER is
  signal qint: std_logic_vector(3 downto 0);

begin

  qint <= x"0" when (reset='0');       -- Asynchrones Rücksetzen auf Null

  process (clk)                        -- Prozess hängt von clk ab
  begin
    if (clk='1') and clk'event         -- warten auf Taktflanke
    then
      qint <= qint+1;                  -- Zählerstand hochzählen
    end if;
  end process;

  q <= qint;                           -- Nebenläufige Ausgabe

end;
```

Abbildung 18: Zuweisung außerhalb des Prozesses

Die beschriebene Funktion ist prinzipiell dieselbe, nur wird hier das asynchrone Rücksetzsignal 'reset' außerhalb der Prozessumgebung abgefragt. Das erscheint eigentlich als sinnvoll, denn der Rücksetzvorgang ist ja nicht an den Takt gebunden. Leider kommt es dabei aber zu zwei nebenläufigen Zuweisungen zu dem Signal 'qint', einmal außerhalb des Prozesses und einmal innerhalb des Prozesses. Damit ist diese Beschreibung nicht möglich.

4.2.3 Zeitpunkt der Signalaktualisierung

Weiterhin müssen Sie in einer sequentiellen Umgebung beachten, dass Signalzuweisungen, die innerhalb eines Prozesses vorgenommen werden (in Abbildung 17 die Zeile 'qint <= qint +1') erst mit der nächsten Taktflanke wirksam werden. Wenn Sie also ein Signal in einem Prozess abfragen, dann wird der Wert **vor** der Taktflanke verwendet und nicht der gerade eben erst neu berechnete Wert. Sie können sich auch dies erklären, wenn Sie an den Speicher des Automaten denken. Auch hier wird der eben mit der Zustandsübergangsfunktion berechnete neue Zustand erst mit der nächsten Taktflanke in den Zustandsspeicher übernommen. Bis dahin steht im Speicher der alte (also bisherige) Wert zur Auswertung zur Verfügung.

4.2.4 Unerwünschte Latches

Die letzte Falle, die in einer sequentiellen Umgebung lauert, ist die Abfrage eines Signals vor einer Zuweisung. In einer nebenläufigen Umgebung hatte das ja keine weiteren Folgen, da die Reihenfolge der Anweisungen ohnehin keine Bedeutung hat. Wenn aber in einer sequentiellen Umgebung auf ein Signal zugegriffen wird, dem bisher kein Wert zugewiesen wurde, dann muss offensichtlich auf einen älteren, d.h. gespeicherten, Wert zurückgegriffen werden. Das Synthesewerkzeug fügt deshalb ein speicherndes Element, meist ein Latch, ein. Genau das wünschen Sie aber in der Regel nicht. Achten Sie also darauf, dass Sie in einer sequentiellen Umgebung einem Signal zuerst einen Wert zuweisen, ehe Sie es verwenden.

4.2.5 Variablen im Prozess

Wie in Kapitel 4.2.3 dargelegt, werden Signale grundsätzlich erst mit dem nächsten Prozessdurchlauf aktualisiert. Da es manchmal wünschenswert ist, auf berechnete Ergebnisse sofort zugreifen zu können, können innerhalb eines Prozesses Variablen deklariert werden. Für Variablen gelten die gleichen Konventionen (Typen, Vektoren) wie für Signale, nur wird statt des Schlüsselwortes 'signal' das Schlüsselwort 'variable' verwendet und Zuweisungen werden nicht mit '<=' gekennzeichnet sondern mit ':='.

```
14  architecture verhalten of ZAEHLER is
15    signal qint: std_logic_vector(3 downto 0);
16
17  begin
18
19    process (reset, clk)                -- Prozess hängt von reset und clk ab
20      variable V: std_logic_vector(3 downto 0);
21    begin
22      if reset='0'                      -- Asynchrones Rücksetzen auf Null
23      then
24        qint <= x"0";
25        V := x"0";
26      elsif (clk='1') and clk'event     -- warten auf Taktflanke
27      then
28        qint <= qint+1;                 -- Zählerstand hochzählen
29        V := V+1;                       -- Hilfsvariable hochzählen
30      end if;
31    end process;
32
33    q <= qint;                          -- Nebenläufige Ausgabe
34
35  end;
```

Abbildung 19: Variablen in Prozessen

In Abbildung 19 ist der bereits bekannte Zähler um die Deklaration und Verwendung einer Variablen 'V' erweitert. Beachten Sie zunächst die lokale Deklaration in Zeile 20. Die Variable ist damit nur innerhalb des Prozesses sichtbar.
In Zeile 25 wird die Variable bei einem Reset ebenfalls auf Null gesetzt; beachten Sie hier die geänderte Syntax für die Zuweisung.
In Zeile 29 wird die Variable synchron mit dem Signal 'qint' um eins erhöht. Der Unterschied ist aber jetzt, dass in weiteren Anweisungen im 'then'-Zweig (Zeilen 27 bis 30) bereits der neue Wert zur Verfügung steht. Wenn also vor der Taktflanke sowohl 'qint' als auch 'V' die Hexadezimalzahl 8 enthalten haben sollten, dann enthält nach Zeile 29 das Signal 'qint' **weiterhin** die Zahl 8 während die Variable 'V' **bereits** die Zahl 9 enthält.
Erst mit der nächsten Taktflanke wird dem Signal 'qint' die neue Zahl 9 zugewiesen, so dass Signal und Variable wieder synchron laufen.

5 Anweisungen

5.1 Einfache Verknüpfungen

Signale und Variablen können miteinander verknüpft werden und das Ergebnis kann einem Signal oder einer Variablen zugewiesen werden. Die Verknüpfung und Zuweisung ist sowohl in nebenläufigen als auch sequentiellen Umgebungen zulässig. Die verfügbaren Verknüpfungen sind in Abbildung 20 gezeigt.

Schlüsselwort	Operation	Bemerkung
not	Negation	hat Vorrang vor allen anderen Operatoren
and	und	Bei „Serienschaltung“ kann die Klammerung entfallen, ’a **and** b **and** c’ ist zulässig
or	oder	
nand	negiertes und	„Serienschaltung“ ohne Klammerung ist nicht zulässig, ’a **nor** b **nor** c’ ist also verboten!
nor	negiertes oder	
xor	Antivalenz (exklusiv oder)	
xnor	Äquivalenz (neg. xor)	

Abbildung 20: Boolesche Operatoren

Grundsätzlich ist es empfehlenswert, Klammern zu setzen, um die Reihenfolge der Verknüpfungen unmittelbar einsichtig zu machen.
Damit können Sie den Multiplexer aus Abbildung 1 in VHDL wie in Abbildung 21 gezeigt beschreiben.

```
library ieee;                          -- die verwendete Bibliothek
use ieee.std_logic_1164.all;           -- die Definition der Grundtypen und Werte


entity MULTIPLEXER is
port
(
  A0, A1, A2, A3: in bit;      -- Datenwort A
  B0, B1, B2, B3: in bit;      -- Datenwort B
  S:              in bit;      -- Steuereingang zur Auswahl (0: A, 1: B)
  Y0, Y1, Y2, Y3: out bit      -- Datenausgang
);
end;


architecture verhalten of MULTIPLEXER is
begin
  y0 <= (a0 and not s) or (b0 and s);  -- Multiplexergleichung
  y1 <= (a1 and not s) or (b1 and s);
  y2 <= (a2 and not s) or (b2 and s);
  y3 <= (a3 and not s) or (b3 and s);
end verhalten;
```

Abbildung 21: Beispiel für Verknüpfung

5.2 Arithmetische Operatoren

In VHDL sind auch komplexere Operationen möglich. Für den Schaltungsentwurf kommen dabei aber meist nur Addition und Subtraktion in Frage, da Operationen wie Multiplikation oder Division in der Regel zu unverhältnismäßig aufwendigen Schaltungen führen würden. Für die Anwendung arithmetischer Operationen muss bekannt sein, auf welchem Typ die Operation arbeitet und wie eine Dualzahl interpretiert wird. Dies wird durch das verwendete Package (das ist die 'use'-Anweisung im Header) festgelegt. Leider stehen nicht in allen Entwurfswerkzeugen alle Möglichkeiten zur Verfügung.
Am sichersten fahren Sie, wenn Sie die im Standard definierte unsigned-Arithmetik auf den Typ std_logic_vector verwenden. Dabei müssen die Signale oder Variablen vom Typ std_logic_vector sein und die Interpretation des Vektors erfolgt als vorzeichenlose Dualzahl.
Falls Sie einen anderen Typ bearbeiten wollen, müssen Sie eine explizite Typkonversion vorsehen. Ebenso können Sie eine vorzeichenbehaftete Interpretation der Zahl, z.B. im Zweierkomplement, selbst ableiten.

Schlüsselwort	Operation	Bemerkung
+	Addition	
-	Subtraktion	
=	gleich	unabhängig vom Typ, geht ohne Arithmetikpackage
/=	ungleich	erzeugt, wie der Vergleich auf Gleichheit, einen booleschen Wert (Typ **boolean**) Anwendung in Abfragen wie **if/elsif/when**
<	kleiner	
<=	kleiner gleich	
>	größer	
>=	größer gleich	

Abbildung 22: Vergleiche und Arithmetik

Sehen Sie sich den Code in Abbildung 23 an. Es handelt sich um eine vollständige Beschreibung des Zählers aus Abbildung 1.

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  use ieee.std_logic_unsigned.all;
 4
 5  entity ZAEHLER is
 6  port
 7  (
 8    CLK:   in bit;                          -- Taktsignal
 9    RESET: in bit;                          -- asynchroner Reset auf Null
10    Q:     out bit_vector(3 downto 0)       -- Ausgabe
11  );
12  end;
13
14  architecture verhalten of ZAEHLER is
15    signal qint: std_logic_vector(3 downto 0); -- internes Hilfssignal
16
17  begin
18
19    process (reset, clk)                    -- Prozess hängt von reset und clk ab
20    begin
21      if (reset='0') then qint <= x"0";     -- Asynchrones Rücksetzen auf Null
22      elsif (clk='1') and clk'event         -- warten auf Taktflanke
23      then
24        if (qint > x"8")                    -- Abfrage auf >8, da Signal noch den
25        then                                -- vorherigen Zählerstand enthält
26          qint <= x"0";
27        else
28          qint <= qint+1;                   -- Zählerstand hochzählen
29        end if;
30      end if;
31    end process;
32
33    q <= to_bitvector(qint);                -- Nebenläufige Ausgabe mit Typkonversion
34  end;
35
```

Abbildung 23: Arithmetik und Vergleich im Zähler

Zunächst einmal wird in Zeile 3 das Package für vorzeichenlose Arithmetik auf den Typ 'std_logic_vector' eingebunden. Sie benötigen dies sowohl für den Vergleich in Zeile 24 als auch die Addition in Zeile 28.
In Zeile 15 wird ein lokales Hilfssignal von diesem Typ deklariert. Selbst wenn das Ausgabesignal (hier 'Q' in Zeile 10) bereits als 'std_logic_vector' deklariert worden wäre, hätten Sie dennoch ein internes Hilfssignal deklarieren müssen. Der Grund ist, dass das Signal 'Q' ja ein reines Ausgabesignal mit der Richtungsangabe 'out' ist . Diese Signale können aber nicht gelesen werden! Also könnten Sie auch nicht (wie in Zeile 28) auf den letzten Zählerstand zugreifen und damit letztlich nicht zählen.
Da der Zähler nur bis 9 zählen soll, wird in Zeile 24 wird auf Zählerüberlauf abgefragt. Es wäre natürlich auch möglich, auf 'qint = 9' abzufragen. In diesem Fall würde aber der Zähler bei den fehlerhaften bzw. beim Einschalten zufällig entstandenen Zuständen 10 bis 15 nicht sofort zurückgesetzt.
Die Abfrage erfolgt auf größer als 8, da das Signal 'qint' ja wie in Abschnitt 4.2.3 beschrieben um einen Takt „nachhinkt". Wenn der Zähler vor dem Takt auf 8 stand, dann wird zwar mit der Zuweisung in Zeile 28 der Zählerstand auf 9 erhöht. Für Abfragen wird dies aber erst **nach dem nächsten Takt** wirksam.
Zuletzt wird der aktuelle Zählerstand in Zeile 33 ausgegeben. Hier ist eine explizite Typkonversion erforderlich, da ja das interne Signal einen anderen Typ als das Ausgabesignal hat.

5.3 with/select

Mit dieser **nebenläufigen** Anweisung kann **ein** Auswahlsignal **mehrfach** abgefragt werden und in Abhängigkeit von dem gewählten Zweig können **einem** Signal beliebige Verknüpfungen zugewiesen werden. Die Anweisung führt in der Schaltung häufig zu einem Multiplexer, bei dem das Auswahlsignal an den Steuereingängen anliegt und die einzelnen Verknüpfungen an den Dateneingängen des Multiplexers anliegen. Das Ergebnis kann dann am Ausgang des Multiplexers abgeholt werden.

with <auswahlsignal> **select**
ergebnis <= <Verknüpfung_1> **when** <auswahlwert_1>,
<Verknüpfung_2> **when** <auswahlwert_2>,
<Verknüpfung_n> **when others**;

Abbildung 24: Syntax with/select

Die Syntax der Anweisung zeigt Abbildung 24. Zu beachten ist zunächst, dass das Auswahlsignal zwar ein Vektor sein, der aber bereits vorher aus Einzelsignalen zusammengesetzt worden sein muss. Deklarieren Sie, wenn notwendig, einfach ein lokales Hilfssignal. Natürlich muss der Typ des Ergebnisses mit dem Typ der einzelnen Verknüpfungen jeweils übereinstimmen. Ebenso muss der Typ des Auswahlsignals mit den Typen der Auswahlwerte übereinstimmen.
Wenn Sie nicht alle Möglichkeiten erschöpfend behandeln, dann sollten Sie noch eine Defaultzuweisung mit der Anweisung 'when others' vornehmen. Speziell beim Typ 'std_logic' haben Sie ja neun Werte. Bei einer erschöpfenden Aufzählung müssten Sie auch alle Werte außer '0' und '1' abfragen – was Sie natürlich nicht tun. Statt dessen verwenden Sie für die letzte Auswahl einfach 'others'. Der Code für den Dekoder aus Abbildung 1 in Abbildung 25 ist damit selbsterklärend.

```
library ieee;
use ieee.std_logic_1164.all;

entity DEKODER is
port
(
  X0,X1,X2,X3:   in  bit;
  EN:            in  bit;
  a,b,c,d,e,f,g: out bit
);
end;

architecture behaviour of DEKODER is
signal y: bit_vector(0 to 6);
signal sel: bit_vector(3 downto 0);

begin
  sel <= (x3,x2,x1,x0);                -- Zusammensetzung erforderlich
  with sel select                      -- Umsetzung Dualzahl in
  y <= "1111110" when x"0",            -- Siebensegmentcode
       "0110000" when x"1",
       "1101101" when x"2",
       "1111001" when x"3",
       "0110011" when x"4",
       "1011011" when x"5",
       "1011111" when x"6",
       "1110000" when x"7",
       "1111111" when x"8",
       "1111011" when x"9",
       "0000000" when others;          -- hier bleibt die Anzeige dunkel

  a <= y(0) and en; b <= y(1) and en;  -- Enable wirkt auf alle Ausgänge
  c <= y(2) and en; d <= y(3) and en;
  e <= y(4) and en; f <= y(5) and en;
  g <= y(6) and en;

end;
```

Abbildung 25: Dekoder mit with/select

5.4 when/else

Auch diese Anweisung ist wie with/select nur in **nebenläufigen** Umgebungen zulässig. Mit dieser Anweisung können jedoch wechselnde Signale auf zutreffende Bedingungen geprüft werden und in Abhängigkeit davon einem Ergebnissignal verschiedene Verknüpfungen zugewiesen werden. Außerdem ist es möglich, dass sich die Bedingungen überlappen – in diesem Fall wird die erste zutreffende Bedingung ausgewählt. Dies war bei with/select nicht möglich, da dort alle Verzweigungen disjunkt sein mussten.

```
<ergebnis>   <=      < Verknüpfung_1> when <Bedingung_1>
                 else  < Verknüpfung_2> when <Bedingung_2>
                 else  <Verknüpfung_n>;
```

Abbildung 26: Syntax when/else

Die in Abbildung 26 gezeigte Syntax zeigt, dass hier beliebige Bedingungen verwendet werden können. Der letzte Zweig ohne Bedingung entspricht der Defaultzuweisung ’when others’.

```
library ieee;
use ieee.std_logic_1164.all;

entity DEKODER is
port
(
  X0,X1,X2,X3:   in  bit;
  EN:            in  bit;
  a,b,c,d,e,f,g: out bit
);
end;

architecture behaviour of DEKODER is
signal y: bit_vector(0 to 6);
signal sel: bit_vector(3 downto 0);

begin

  sel <= "1111" when (EN='0')          -- Enable hier am Eingang
              else (x3,x2,x1,x0);

  with sel select                      -- Umsetzung Dualzahl in
  y <= "1111110" when x"0",            -- Siebensegmentcode
       "0110000" when x"1",
       "1101101" when x"2",
       "1111001" when x"3",
       "0110011" when x"4",
       "1011011" when x"5",
       "1011111" when x"6",
       "1110000" when x"7",
       "1111111" when x"8",
       "1111011" when x"9",
       "0000000" when others;          -- hier bleibt die Anzeige dunkel

  a <= y(0); b <= y(1); c <= y(2); d <= y(3);
  e <= y(4); f <= y(5); g <= y(6);

end;
```

Abbildung 27: Beispiel when/else

In diesem Beispiel wird die Enable-Funktion dadurch realisiert, dass in einer when/else-Anweisung das interne Auswahlsignal ’sel’ entweder mit dem nicht vorkommenden Wert ’1111’“ belegt wird oder eben mit dem anzuzeigenden Wert. Die nachfolgende Codeumsetzung sorgt dafür, dass bei der Auswahl ’1111’ die Anzeige dunkel bleibt.

5.5 if/then

Diese Anweisung entspricht der when/else-Anweisung, ist jedoch nur in **sequentiellen** Umgebung zulässig. Mit dieser Anweisungen können beliebige Blöcke von sequentiellen Anweisungen in Abhängigkeit von beliebigen Bedingungen ausgeführt werden.
Die Syntax zeigt Abbildung 28.

```
if      <Bedingung_1>     then  <Sequentielle Anweisungen 1>;
elsif   <Bedingung_2>     then  <Sequentielle Anweisungen 2>;
elsif   <Bedingung_2>     then  <Sequentielle Anweisungen 3>;
else                            <Sequentielle Anweisungen n>;
end if;
```

Abbildung 28: Syntax if/then

Beachten Sie die unterschiedliche Schreibweise von 'elsif' und 'end if'. Die Bedingungen müssen vom Typ boolean sein. Das können entweder Signale/Variablen von diesem Typ sein oder aber Vergleiche zweier Ausdrücke gleichen Typs.
Da in jedem Zweig ein ganzer Block von Anweisungen stehen kann, können if/then-Anweisungen beliebig tief geschachtelt werden.

```
process (reset, clk)                    -- Prozess hängt von reset und clk ab
begin
  if (reset='0') then qint <= x"0";     -- Asynchrones Rücksetzen auf Null
  elsif (clk='1') and clk'event         -- warten auf Taktflanke
  then
    if (qint > x"8")                    -- Abfrage auf >8, da Signal noch den
    then                                -- vorherigen Zählerstand enthält
      qint <= x"0";
    else
      qint <= qint+1;                   -- Zählerstand hochzählen
    end if;
  end if;
end process;
```

Abbildung 29: Beispiel if/then

Der Code in Abbildung 29 ist ein Ausschnitt aus dem Code für den Zähler (Abbildung 23). Dabei wird eine geschachtelte if/then-Konstruktion verwendet, denn in dem then-Zweig der ersten elsif-Bedingung ist eine weitere if/then-Anweisung enthalten.
An diesem Beispiel wird auch deutlich, dass mit der if/then-Anweisung eine Priorisierung bei mehreren gleichzeitig wahren Bedingungen erreicht wird.
Angenommen, eine steigende Taktflanke trifft ein während das Signal 'reset' den Wert '0' hat. Dann wird dennoch nur der then-Zweig der ersten if/then-Anweisung ausgeführt, d.h. der Zähler bleibt zurückgesetzt. So soll es bei einem asynchronen Reset auch sein.

5.6 case/is

Diese Anweisung entspricht der with/select-Anweisung, ist jedoch nur in **sequentiellen** Umgebung zulässig. Mit dieser Anweisungen können ebenfalls beliebige Blöcke von sequentiellen Anweisungen in Abhängigkeit von **einem** Testsignal ausgewählt werden.

```
case    <testsignal> is
        when  <Wert_1>     => <Sequentielle Anweisungen 1>;
        when  <Wert_2>     => <Sequentielle Anweisungen 2>;
        when  <Wert_2>     => <Sequentielle Anweisungen 3>;
        when others        => <Sequentielle Anweisungen n>;
end case;
```

Abbildung 30: Syntax case/is

Das zu testende Signal und die Testwerte müssen natürlich vom gleichen Typ sein. Wie bei der with/select-Anweisung sollten Sie eine Defaultzuweisung in einem Zweig 'when others' vorsehen. Dies ist hier besonders wichtig, weil die case/is-Anweisung fast immer bei der Beschreibung einer Zustandsübergangsfunktion benutzt wird. Ist diese Funktion aber nicht vollständig definiert, dann besteht die Gefahr, dass der Automat in einen ungewollten Zustand übergeht und evtl. dort bleibt.
Da ganze Blöcke aufgrund einer Bedingung ausgeführt werden können, sind auch hier beliebig tiefe Schachtelungen möglich.
Ein ausführliches Beispiel zur case/is-Anweisung folgt in der Automatenbeschreibung.

```
architecture verhalten of MULTIPLEXER is
  signal a,b,y: bit_vector(0 to 3);   -- Hilfsvektoren

begin
  a <= (a0,a1,a2,a3);                 -- Kopieraktionen
  b <= (b0,b1,b2,b3);
  y0 <= y(0); y1 <= y(1); y2 <= y(2); y3 <= y(3);

  process (s,a,b)                     -- Empfindlichkeitsliste
  begin
    case s is
      when '0' => y <= a;
      when '1' => y <= b;
    end case;
  end process;

end;
```

Abbildung 31: Beispiel case/is

Im Code aus Abbildung 31 für den Multiplexer musste zunächst mit dem Prozess eine sequentielle Umgebung geschaffen werden, da case/is in nebenläufigen Umgebungen unzulässig ist. Als Besonderheit kann hier die Defaultzuweisung entfallen, weil das Testsignal vom Typ 'bit' ist und damit tatsächlich nur die Werte '0' und x'1' annehmen kann. Diese beiden Möglichkeiten werden auch explizit abgefragt, so dass keine weitere Möglichkeit mehr übrig bleibt.

6 Automatenbeschreibung

Automaten können in VHDL auf vielfältige Weise beschrieben werden. Die Art der Beschreibung hat teilweise Einfluss auf das Syntheseergebnis; manche Entwurfsprogramme können auch nicht alle möglichen Beschreibungsformen umsetzen. Aus diesem Grund wird hier nur eine einzige Möglichkeit zur Beschreibung eines Moore-Automaten vorgestellt. Die Beschreibung kann von allen bekannten Entwurfswerkzeugen umgesetzt werden und ist gleichzeitig übersichtlich. Die Erweiterung auf den Mealy-Automaten ist trivial.
Ein Moore-Automat enthält drei voneinander getrennte Blöcke:

- Zustandsspeicher
- Zustandsübergangsfunktion
- Ausgabefunktion

Im diesem „Schema F“-Entwurf werden diese drei Blöcke in einer nebenläufigen Umgebung als zwei parallel ausgeführte Prozesse modelliert. Der erste Prozess beschreibt lediglich den Zustandsspeicher, während der zweite Prozess sowohl die Zustandsübergangsfunktion als auch die Ausgabefunktion beschreibt.
Für die Zustandsübergangsfunktion und die Ausgabefunktion wären auch nebenläufige Anweisungen denkbar. Die Darstellung als Prozess hat aber den Vorteil, dass die case/is-Anweisung verwendet werden kann, die für jede Bedingung wieder einen ganzen Block mit Anweisungen ermöglicht.
Das Grundgerüst hat in diesem Schema die in Abbildung 32 gezeigte Struktur:

```
architecture verhalten of automat is
  signal z_alt, z_neu: zustaende;                  -- symbolische zustaende
begin

  process (clk, init)                              --  zustandsspeicher
  begin
    if (init) then zustand <= resetzustand;        -- hier die asynchrone initialisierung
    elsif (clk = '1') and clk'event                -- warten auf steigende taktflanke
    then
      z_alt <= z_neu;                              -- zustandsaktualisierung
    end if;
  end process;

  process (z_alt, eingaenge)                       -- ausgabe-/uebergangsfunktion
  begin
    case z_alt is
      when z0 => z_neu <= f_u(z0,eingaenge);       -- uebergaenge vom zustand z0
                 aus   <= f_a(z0, eingaenge);      -- ausgabe im zustand z_0
      when z1 => z_neu <= f_u(z1,eingaenge);       -- uebergaenge vom zustand z1
                 aus   <= f_a(z1, eingaenge);      -- ausgabe im zustand z_1
      ....
     end case;
  end process;

end architecture;
```

Abbildung 32: Schema für den Moore-Automat

6.1 Spezielle Konstrukte

Für eine gut lesbare und erfolgreich synthetisierbare Beschreibung nach diesem Schema benötigen Sie noch zwei bisher nicht vermittelte Kenntnisse.

6.1.1 Symbolische Zustandskodierung

Bei Ihrem Papierautomaten haben Sie sinnvollerweise symbolische Zustandsnamen wie ein, aus, vorwaerts und rueckwaerts gewählt. Dafür gibt es in VHDL keinen geeigneten Typ, obwohl es sicherlich günstig wäre, diese Namen zunächst beizubehalten. Mit der in Abbildung 33 gezeigten Anweisungsfolge können Sie sich aber in VHDL einen eigenen Aufzählungstyp definieren.

Syntax:
type <typname> **is** (wert1, wert2, ..., wert_n)**;**

Beispiel:
type gaenge **is** (leerlauf, vorwaerts, rueckwaerts);
signal z_alt, z_neu: gaenge;

Abbildung 33: Syntax Definition eines Aufzählungstyps

Sie haben damit zwei Signale ’z_alt’ und ’z_neu’ zur Verfügung, die jeweils die Werte ’leerlauf’, ’vorwaerts’ und ’rueckwaerts’ annehmen können. Sie können mit diesen Werten zwar keine Verknüpfungen bilden – Sie können aber die symbolischen Namen einem geeigneten Signal zuweisen und ein Signal mit einem Wert auf Gleichheit prüfen. Mehr benötigen Sie in der Automatenbeschreibung auch nicht!
Wenn Sie den fertig beschriebenen Automaten synthetisieren wollen, dann müssen Sie natürlich den symbolischen Namen Dualzahlen zuordnen. Wenn Sie nichts weiter tun, dann wird das Entwurfswerkzeug automatisch eine Zuordnung durchführen. Bei manchen Werkzeugen können Sie auch noch als Option auswählen, nach welchen Verfahren das geschehen soll. Übliche Möglichkeiten sind binär, one-hot und gray.
Sie können aber auch selbst eine Kodierung vornehmen, und zwar ohne dass Sie in der gesamten VHDL-Beschreibung die symbolischen Namen durch Dualzahlen ersetzen. Erweitern Sie die Definition aus Abbildung 33 um eine Zuordnung nach Abbildung 34.

Syntax:
type <typname> **is** (wert1, wert2, ..., wert_n)**;**
attribute enum_encoding: string**;**
attribute enum_encoding of <typname> **: type is “**zahl_fuer_wert1,zahl_fuer_wert2,…”;

Beispiel:
type gaenge **is** (leerlauf, vorwaerts, rueckwaerts);
attribute enum_encoding: string;
attribute enum_encoding of gaenge: type is “00 11 10”;
signal z_alt, z_neu: gaenge;

Abbildung 34: Beispiel Zustandskodierung

In diesem Beispiel haben Sie festgelegt, dass der Zustand ’leerlauf’ mit “00“ kodiert wird, der Zustand ’vorwaerts’ mit “11“ und der Zustand ’rueckwaerts’ mit ’10’.

6.1.2 Prozessstruktur

Die meisten Synthesewerkzeuge können eine sequentielle Schaltung nur dann fehlerfrei abbilden, wenn sie in einer „Normaldarstellung“ beschrieben sind. Alle bekannten Synthesewerkzeuge können Beschreibung nach Abbildung 35 erkennen und erfolgreich synthetisieren.

```
process (<empfindlichkeitsliste>)
-- hier lokale Signale und Variablen deklarieren
begin
  if (<asynchrone bedingung>)
    then
      <z_neu> <= initialzustand;
    elsif <takt> = ’wert’ and <takt>’event
      then
        -- hier beginnt die sequentielle umgebung
  end if;
end process;
```

Abbildung 35: Synthetisierbarer taktsynchroner Prozess

Beachten Sie zunächst, dass Sie alle notwendigen Signale auch wirklich in die Empfindlichkeitsliste des Prozesses eintragen.
Falls Ihr Automat mit einem asynchronen Signal in einen initialen Zustand gebracht werden kann (Beispiel: Reset), dann beginnen Sie den Prozess sofort mit einer entsprechenden if/then-Anweisung.
In den elsif-Bedingung dieser Anweisung (bzw. die if-Bedingung, wenn Sie keine asynchrone Initialisierung benötigen) schreiben Sie für einen Übergang bei

- **steigender** Taktflanke: (takt = ’1’) and takt’event
- **fallender** Taktflanke: (takt = ’0’) and takt’event

Versuchen Sie nicht, einen Automaten zu kreieren, der auf beide Taktflanken reagiert.
Versuchen Sie nicht, die Taktflankenabfrage an eine andere Stelle zu verschieben.
Versuchen Sie nicht, die asynchrone Bedingungsabfrage an eine andere Stelle zu verschieben.
Versuchen Sie nicht, mehrere Taktflankenabfragen einzubauen.

Sollten Sie in einer nicht von Ihnen verfassten Beschreibung eine ’wait’-Anweisung am Ende des Prozesses sehen, dann denken Sie sich statt dieser Anweisung die entsprechende Taktflankenabfrage an den Beginn des Prozesses (wie in Abbildung 35). Dies ist eine weitere übliche Methode, die jedoch (Stand 2002) an Bedeutung verliert.

6.2 Beispiel Frag-O-Mat

In diesem Kapitel wird mit den bisherigen Kenntnissen ein Moore-Automat in VHDL entworfen. Die bisherige Erfahrung hat gezeigt, dass sich professorales Frageverhalten (leider) einigermaßen zuverlässig mit einem Automaten namens „Frag-O-Mat“ nachbilden lässt. Es seien in der Vorlesung noch ein harter Kern von vier Hörern A,E,L und S übriggeblieben. Diese werden normalerweise der Reihe nach befragt. Kann die Frage beantwortet werden (Bedingung w=weiß es), dann kommt als nächstes der alphabetisch folgende Kandidat dran. Dies ist im Zustandsübergangsgraph (Abbildung 36) die rechte Schleife A -> E -> L -> S -> A.

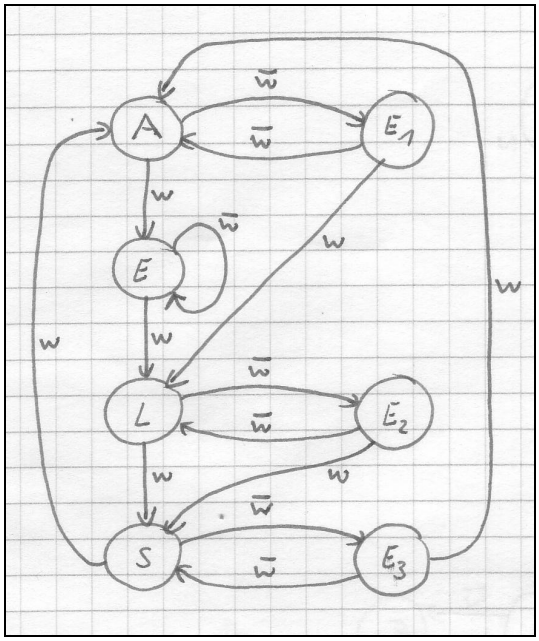


Abbildung 36: Zustandsübergänge „Frag-O-Mat“

Wenn die Frage nicht beantwortet werden kann (w_bar = weiß es nicht), dann kommt zunächst E an die Reihe, da hier die größte Wahrscheinlichkeit besteht, eine zufriedenstellende Antwort zu erhalten. Im Zustandsübergangsgraph sind dies die Übergänge nach rechts in die Zustände E1, E2 und E3 sowie der Sonderfall E -> E.
Weiß es E auch nicht, dann war die Frage zu schwer oder schlecht formuliert. In diesem Fall erhält der zuerst befragte Hörer eine weitere Chance mit einer besser angepassten Frage (Übergänge von E1, E2 und E3 nach links sowie der Sonderfall E -> E).
Konnte E dagegen die Frage beantworten, dann kommt als nächstes der ohnehin als „normaler“ Nachfolger vorgesehene Hörer an die Reihe (Übergänge von E1 und E2 nach links unten bzw. von E3 nach A).
Eine kurze Prüfung ergibt, dass der Graph sowohl vollständig als auch widerspruchsfrei ist.
Die Ausgabefunktion ist besonders einfach, es wird einfach der einem Zustand zugeordnete Hörer angezeigt (Abbildung 37)

Zustand	A	L	S	E	E1	E2	E3
Ausgabe	A	L	S	E			

Abbildung 37: Ausgabefunktion Frag-O-Mat

Der Automat ist, zumindest nach außen, sehr genügsam (Abbildung 38)

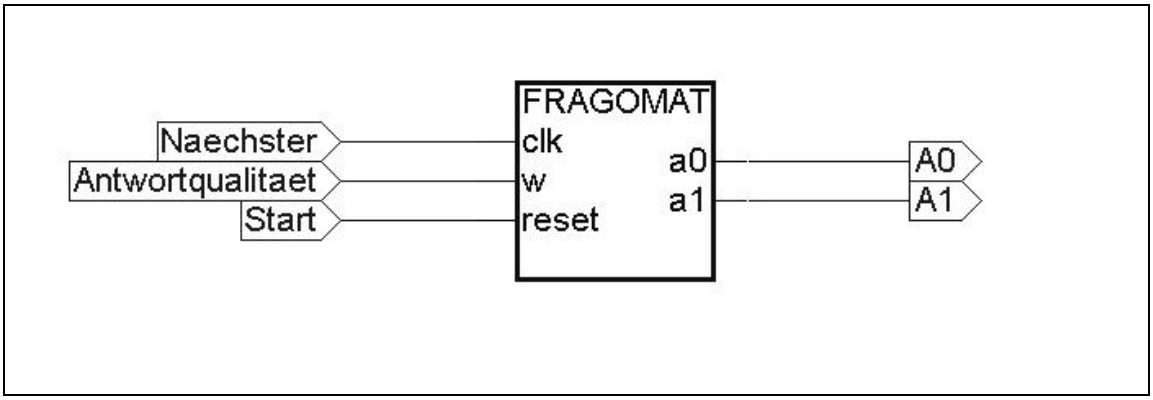


Abbildung 38: Blocksymbol des Frag-O-Mat

Die ersten beiden Abschnitte der dazupassenden VHDL-Beschreibung sind daher kaum mehr als eine Fingerübung (Abbildung 39).

```
 1
 2  library ieee;
 3  use ieee.std_logic_1164.all;
 4
 5
 6  entity fragomat is
 7  port
 8  (
 9    clk, w, reset: in std_logic;
10    a0, a1:        out std_logic
11  );
12  end;
13
14
```

Abbildung 39: Header und Entity des Frag-O-Mat

```
15  architecture verhalten of fragomat is
16   type hoerer is (Z_A,Z_E,Z_L,Z_S,Z_E1,Z_E2,Z_E3);
17   attribute enum_encoding: string;
18   attribute enum_encoding of hoerer: type is "000 100 001 010 101 110 111";
19   signal zalt, zneu: hoerer;
20   signal ausgabe: std_logic_vector(0 to 1);
21
22  begin
23
24   process (clk, reset)
25   begin
26     if (reset = '1')
27       then zalt <= Z_A;
28       elsif (clk = '1') and clk'event
29         then zalt <= zneu;
30    end if;
31    end process;
32
```

Abbildung 40: Deklarationen und Speicherprozess

In Zeile 15 beginnt die Funktionsbeschreibung. Zunächst wird ein eigener Aufzählungstyp ’hoerer’ definiert, damit die symbolischen Zustandsnamen weiterverwendet werden können. Außerdem wird - nur zu Demonstrationszwecken - eine Zustandskodierung vorgegeben.
Der erste Prozess für den Zustandsspeicher beginnt in Zeile 24. Bei einem Reset wird asynchron der Zustand Z_A eingestellt, ansonsten mit der steigenden Flanke am Takt der neue Zustand übernommen.
Die Beschreibung hält sich genau an das in Abbildung 35 angegebene Schema!

```
33
34  process (w, zalt)
35    variable x: boolean;
36  begin
37    if (w = '1')
38      then x := true;
39      else x := false;
40    end if;
41
42    case zalt is
43      when Z_A  => if x
44                     then zneu <= Z_E;
45                     else zneu <= Z_E1;
46                   end if;
47                   ausgabe <= "00";
48      when Z_E  => if x
49                     then zneu <= Z_L;
50                     else zneu <= Z_E;
51                   end if;
52                   ausgabe <= "11";
53      when Z_L  => if x
54                     then zneu <= Z_S;
55                     else zneu <= Z_E2;
56                   end if;
57                   ausgabe <= "01";
58      when Z_S  => if x
59                     then zneu <= Z_A;
60                     else zneu <= Z_E3;
61                   end if;
62                   ausgabe <= "10";
63      when Z_E1 => if x
64                     then zneu <= Z_L;
65                     else zneu <= Z_A;
66                   end if;
67                   ausgabe <= "11";
68      when Z_E2 => if x
69                     then zneu <= Z_S;
70                     else zneu <= Z_L;
71                   end if;
72                   ausgabe <= "11";
73      when Z_E3 => if x
74                     then zneu <= Z_A;
75                     else zneu <= Z_S;
76                   end if;
77                   ausgabe <= "11";
78    end case;
79  end process;
```

Abbildung 41: Zustandsübergangsfunktion und Ausgabefunktion

Die Beschreibung wird jetzt mit dem zweiten Prozess für die beiden Schaltnetze fortgesetzt. Beachten Sie zunächst, dass die Empfindlichkeitsliste dieses Prozesses ganz anders als die des ersten ist: Der Prozess hängt vom Zustand und den Eingangssignalen ab, nicht aber vom Takt oder dem Reset.
Nur zu Demonstrationszwecken wird ein lokales boolesches Signal 'x' deklariert, das mit der if/then-Anweisung in den Zeilen 37 bis 40 mit einem später direkt abfragbaren Wert für die Übergangsbedingungen „w" und „w_bar" belegt wird.
In den Zeilen 42 bis 78 sind nun in einer einzigen case/is-Anweisung alle notwendigen Übergänge und Ausgaben beschrieben. Beachten Sie, dass zunächst mit der case/is-Anweisung die einzelnen Zustände unterschieden werden können. Die Zeilen 53 bis 57 gelten zum Beispiel, wenn sich der Automat gerade im Zustand Z_L befindet. Die if/then-Anweisung für diesen Fall (Zeilen 53 bis 56) berechnet die beiden hier möglichen Folgezustände. Dabei wird schlicht aus dem Papierentwurf abgeschrieben!
Ebenso einfach erfolgt in jedem Zustand die direkte Zuweisung an den Ausgabevektor in Zeile 57.
Die Beschreibung endet mit dem Umkopieren an die Ausgangssignale (Abbildung 42).

```
80
81
82   a0 <= ausgabe(0);
83   a1 <= ausgabe(1);
84  end;
85
```

Abbildung 42: Ausgabe des internen Ausgabesignals

13.5 DE0 Board

Chapter 2
Altera DE0 Board

This chapter presents the features and design characteristics of the DE0 board.

2.1 Layout and Components

A photograph of the DE0 board is shown in Figure 2-1. It depicts the layout of the board and indicates the location of the connectors and key components.

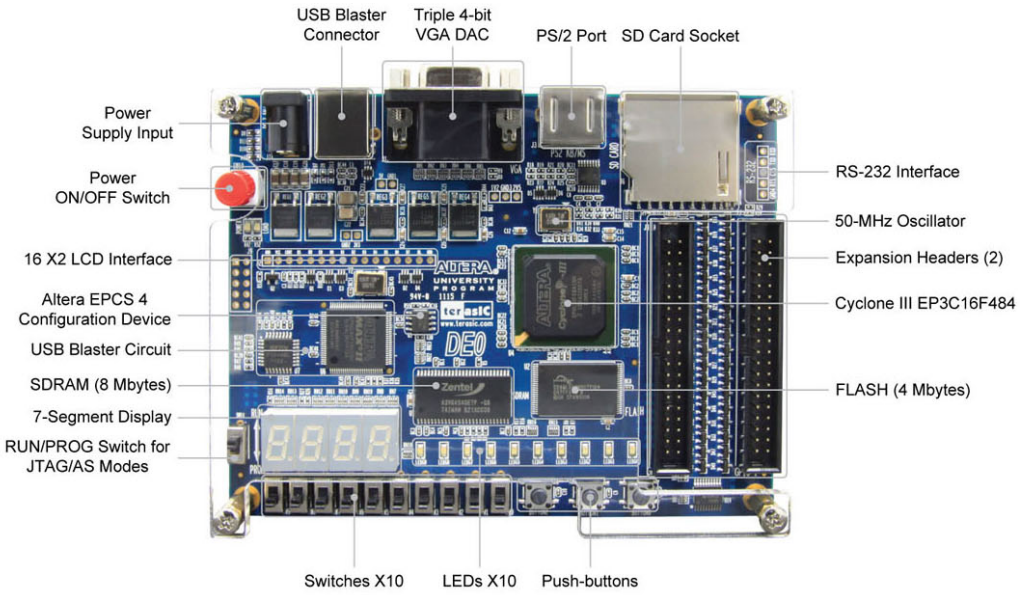


Figure 2-1 The DE0 board.

The DE0 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware is provided on the DE0 board:

- ®Altera Cyclone III 3C16 FPGA device
- Altera Serial Configuration device – EPCS4
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory
- SD Card socket
- 3 pushbutton switches
- 10 toggle switches
- 10 green user LEDs
- 50-MHz oscillator for clock sources
- VGA DAC (4-bit resistor network) with VGA-out connector
- RS-232 transceiver
- PS/2 mouse/keyboard connector
- Two 40-pin Expansion Headers

2.2 Block Diagram of the DE0 Board
Figure 2-2 gives the block diagram of the DE0 board. To provide maximum flexibility for the user, all connections are made through the Cyclone IIII FPGA device. Thus, the user can configure the FPGA to implement any system design.

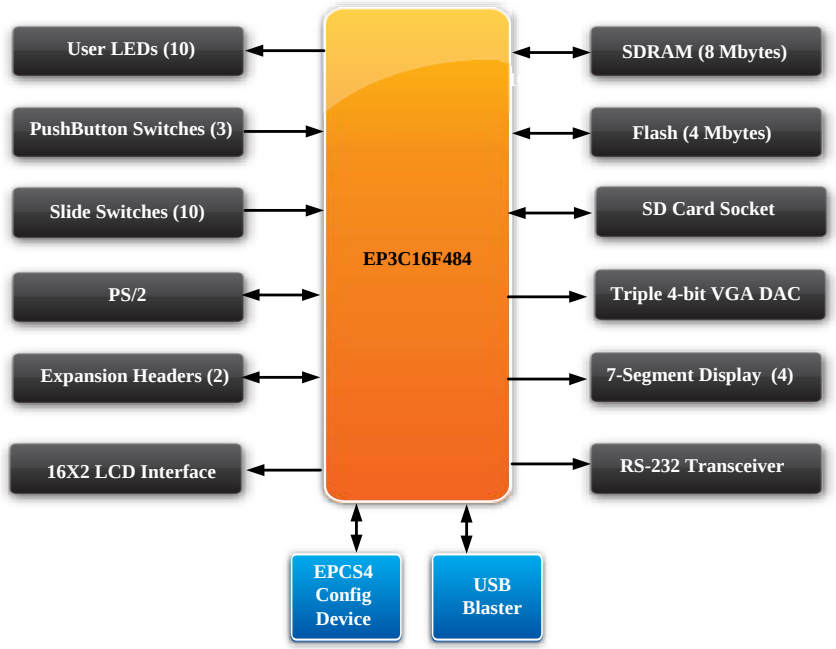


Figure 2-2 Block diagram of the DE0 board.

Following is more detailed information about the blocks in Figure 2-2:

Cyclone IIII 3C16 FPGA

- 15,408 LEs
- 56 M9K Embedded Memory Blocks
- 504K total RAM bits
- 56 embedded multipliers
- 4 PLLs
- 346 user I/O pins
- FineLine BGA 484-pin package

Built-in USB Blaster circuit

- On-board USB Blaster for programming and user API (Application programming interface) control
- Using the Altera EPM240 CPLD

SDRAM

- One **8**-Mbyte Single Data Rate Synchronous Dynamic RAM memory chip
- Supports 16-bits data bus

Flash memory

- 4-Mbyte NOR Flash memory
- Support Byte (8-bits)/Word (16-bits) mode

SD card socket

- Provides both SPI and SD 1-bit mod SD Card access

Pushbutton switches

- 3 pushbutton switches
- Normally high; generates one active-low pulse when the switch is pressed

Slide switches

- 10 Slide switches
- A switch causes logic 0 when in the DOWN position and logic 1 when in the UP position

General User Interfaces

- 10 Green color LEDs (Active high)
- 4 seven-segment displays (Active low)
- 16x2 LCD Interface (Not include LCD module)

Clock inputs

- 50-MHz oscillator

VGA output

- Uses a 4-bit resistor-network DAC
- With 15-pin high-density D-sub connector
- Supports up to 1280x1024 at 60-Hz refresh rate

Serial ports

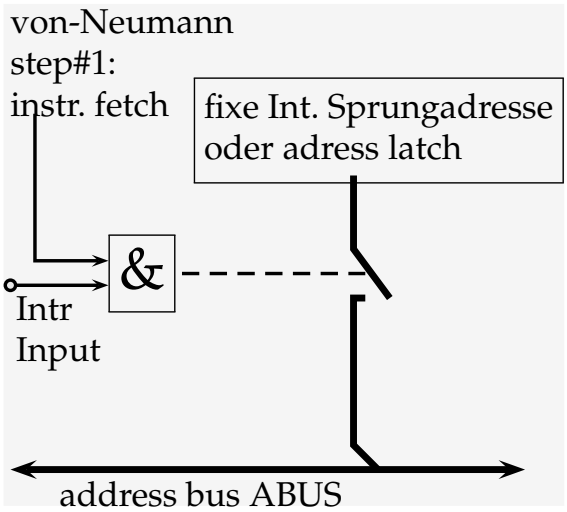
- One RS-232 port (Without DB-9 serial connector)
- One PS/2 port (Can be used through a PS/2 Y Cable to allow you to connect a keyboard and mouse to one port)

Two 40-pin expansion headers

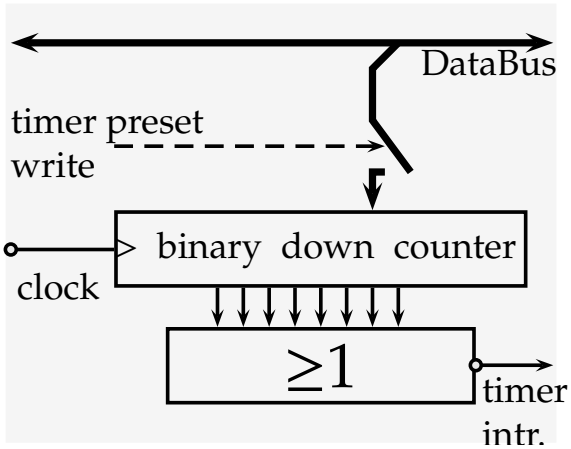
- 72 Cyclone III I/O pins, as well as 8 power and ground lines, are brought out to two 40-pin expansion connectors
- 40-pin header is designed to accept a standard 40-pin ribbon cable used for IDE hard drives

Teil V
Peripheriekomponenten

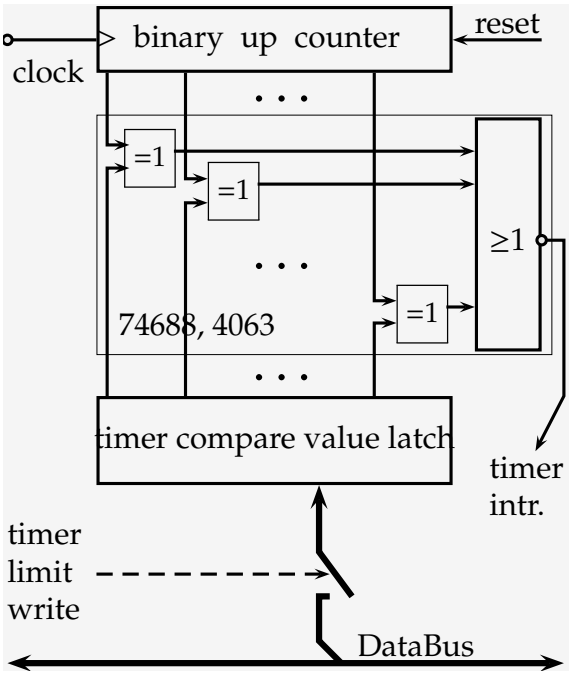
14.1 Interrupt am Prozessor



14.2 Timer, down counting



14.3 Timer, up counting



14.4 DMA Direct Memory Access

= Zusatzhardware zur Beschleuniging von RAM- Kopierprozessen
durch Einsparung der von-Neumann-Prozessor-Zyklen und Programmspeicherzugriffe

ohne DMA:

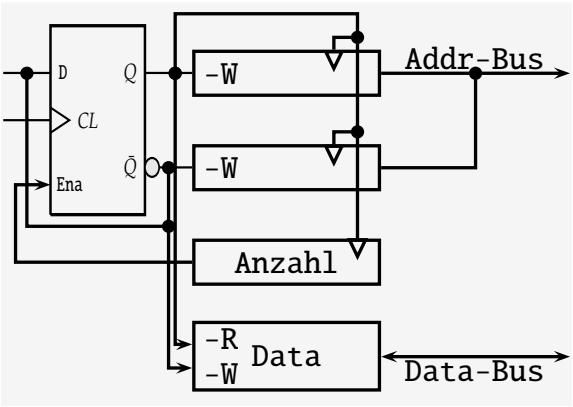
fetch
decode
load
execute
store

```
LOAD C,Anz
LOAD IX,*Quelle
LOAD IY,*Ziel
LOOP:
LOAD A,M(IX)
STORE A,M(IY)
INC IX
INC IY
DEC C
JNZ LOOP
```

Z80
code

mit DMA:

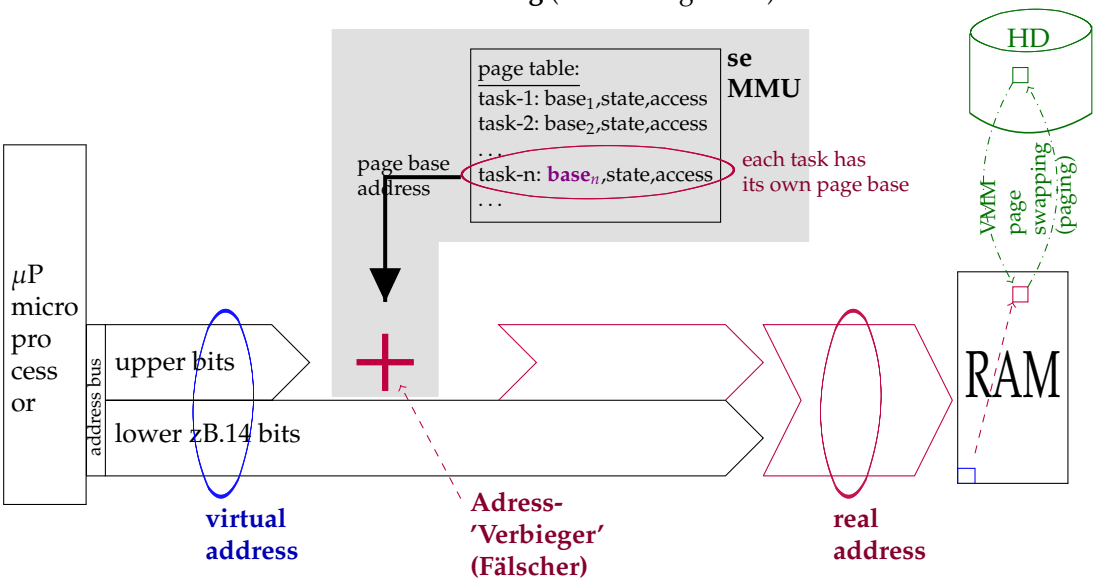
```
COUNTER1 <- Quelladresse
COUNTER2 <- Zieladresse
COUNTER3 <- Anzahl
DATAREG.
GO
```



DMA Hardware

14.5 MMU Memory Management Unit

= Zusatzhardware für **VMM** (Virtual Memory Management) und multitask-**Scheduling** (task management)



14.6 VMM Virtual Memory Management

virtual memory ist der Speicher *aus Sicht des Prozessors*, der vom Prozessor ansprechbare Daten- und Programmspeicher

real memory ist der *physisch* vorhandene Daten- und Programmspeicher

VMM

a) trennt (mithilfe der MMU) die Speicherbereiche verschiedener (parallel-)Prozesse,
indem sie unterschiedliche *page base* Adressen verwenden, welche beim Task-Wechsel
ausgetauscht werden.

b) lagert Daten- und Programmbereiche inaktiver Prozesse auf externe Speichermedien (HDs, 'Auslagerungsdatei', 'swap partition') aus (bzw. bedarfsfalls wieder ein), um scheinbar (=virtuell) mehr Speicherkapazität vorzutäuschen. Bei Multiprozessor-Systemen ist das gar nicht so trivial

cache memory ist *Zwischenspeicher* (zwischen RAM und Prozessor) zu Beschleunigungszwecken

wiki:

de.wikipedia.org/wiki/Arbeitsspeicher (26Okt'21):

Der Arbeitsspeicher oder Hauptspeicher (englisch core, main store, main memory, primary memory, RAM = Random Access Memory) eines Computers ist die Bezeichnung für den Speicher, der die gerade auszuführenden Programme oder Programmteile und die dabei benötigten Daten enthält. Der Hauptspeicher ist eine Komponente der Zentraleinheit. Da der Prozessor unmittelbar auf den Hauptspeicher zugreift, beeinflussen dessen Leistungsfähigkeit und Größe in wesentlichem Maße die Leistungsfähigkeit der gesamten Rechenanlage.

Arbeitsspeicher wird charakterisiert durch die Zugriffszeit bzw. Zugriffsgeschwindigkeit und (damit verbunden) die Datenübertragungsrate sowie die Speicherkapazität. Die Zugriffsgeschwindigkeit beschreibt die Dauer, bis angefragte Daten gelesen werden können. Die Datenübertragungsrate gibt an, welche Datenmenge pro Zeit gelesen werden kann. Es können getrennte Angaben für Schreib- und Lesevorgang existieren. Zur Benennung der Arbeitsspeichergröße existieren zwei unterschiedliche Notationsformen, die sich aus der verwendeten Zahlenbasis ergeben. Entweder wird die Größe zur Basis 10 angegeben (als Dezimalpräfix; 1 kByte oder kB = 103 Bytes = 1000 Bytes, SI-Notation) oder zur Basis 2 (als Binärpräfix; 1 KiB = 210 Bytes = 1024 Bytes, IEC-Notation). Aufgrund der binärbasierten Struktur und Adressierung von Arbeitsspeichern (Byte-adressiert bei 8-Bit-Aufteilung, wortadressiert bei 16-Bit-Aufteilung, doppelwortadressiert bei 32-Bit-Aufteilung usw.) ist letztere Variante die üblichere Form, die zudem ohne Brüche auskommt.

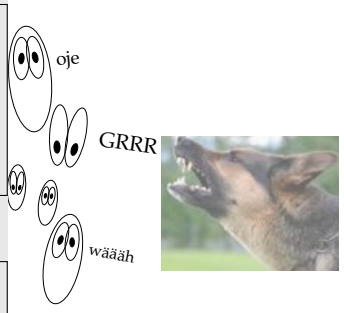
Soweit Arbeitsspeicher über den Adressbus des Prozessors angesprochen wird oder direkt im Prozessor integriert ist, spricht man von physischem Speicher. Modernere Prozessoren und Betriebssysteme können durch virtuelle Speicherverwaltung mehr Arbeitsspeicher bereitstellen, als physischer Speicher vorhanden ist, indem sie Teile des Adressraums mit anderen Speichermedien hinterlegen (etwa mit einer Auslagerungsdatei, pagefile oder swap u. a.). Dieser zusätzliche Speicher wird virtueller Speicher genannt. Zur Beschleunigung des Speicherzugriffs - physisch oder virtuell - kommen heute zusätzliche Pufferspeicher zum Einsatz.

14.7 MAC Operation mit Aze

= Zusatzhardware für **M**ultiply and **AC**umulate (zammzählen)

in VHDL:

```
constant PtrWidth    :integer := 16;
constant CtrWidth    :integer := 16;
constant ResultWidth :integer := 32;

signal aPTR, bPTR: std_logic_vector(PtrWidth downto 0);
signal anz,
       ctr2:       std_locig_vector(CtrWidth downto 0);
signal prod, sum:  std_logic_vector(ResutlWidth downto 0);

MAC: PROCESS (clock) BEGIN

  IF( anz > 0 ) THEN       /*check ob MAC fertig*/

    enable_cpu_clock <= '0';  /*der CU die clock abdrehen*/

    IF(ctr2=0) THEN

      IF (rising_edge(clock)) THEN        /*''''''''''*/
        ABUS  <= aPtr;                    /* a[anz]   */
        Read  <= '0';   --Lo-aktiv)       /* aus RAM  */
      ELSE                                /* holen    */
        RegA  <= DBUS;                    /*.........*/
        Read  <= '1';   --Lo-aktiv)
      END IF;

    ELSIF(ctr2=1) THEN

      IF (rising_edge(clock)) THEN        /*''''''''''*/
        ABUS  <= bPtr;                    /* b[anz]   */
        Read  <= '0';   --Lo-aktiv)       /* aus RAM  */
      ELSE                                /* holen    */
        RegB  <= DBUS;                    /*.........*/
        Read  <= '1';   --Lo-aktiv)
      END IF;

    ELSIF(ctr2=2) THEN

      IF(rising_edge(clock)) THEN
        prod  <= aReg*bReg;               /* MULtiplizieren */}
        sum   <= sum+prod;                /* ACCumulieren   */}MAC
      ELSE
        anz  <= anz-1;                       --zaehler decrement
        aPtr <= aPtr+1;                      --Pointer increment
        bPtr <= bPtr+1;                      --''--
        ctr2 <= 0;                           --MAC-Ausfuehrungsphase ruecksetzen
      END IF;
    END IF;

  ELSE
    enable_cpu_clock <= '1';            /*CU-clock wieder einschalten*/
  END IF;
END PROCESS MAC;
```

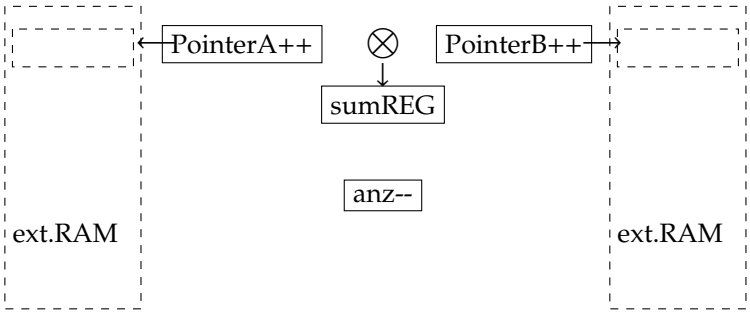


[1] enable_cpu_clock: Während Ausführung einer MAC-Operation darf die CU keinen neuen von-Neumann-Zyklus beginnen. Wir erreichen das zB., indem wir der CU die clock 'absperren' (das wäre in der CU mittels AND-Gatters nachzuimplementieren ('CUclock <= clock AND enable_cpu_clock').

[2] die MAC-Operation beginnt, sobald das Register 'anz' > 0

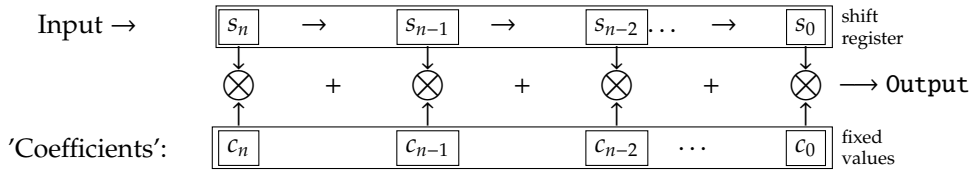
[3] MAC bezeichnet teilweise nur $sum+=a_i * b_i$ meist aber das ganze Skalarprodukt.

DSP MAC Unit:



mit VHDL-'FOR':

```
FOR i in 0 to 1023 loop
  sum <= sum + sample(i)*coeff(i);
end loop
```



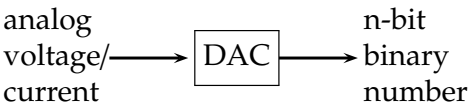
Des erzeugt allerdings ein komplettes, aber unflexibles FIR mit fix 1024 Schieberegister-Zellen. So wird es in einem DSP (Digital SignalProcessor) nicht gemacht! Ein DSP hat nur 1 Stk Multiplizierwerk und hat Samples sowie Koeffizienten in Pointer-addressierbaren RAM-Speichern ('PointerA', 'PointerB') s.o.

14.8 ADC u.DAC

14.8.1 Begriffe

analog	stufenlos
digital	nur zulässige Stufen
Digitalisierung	Rasterung, Abtastung und/oder Quantisierung
Rasterung der t-Zeitachse	Abtastung (sampling)
Rasterung der y-Größenachse	Quantisierung (quantization), Auflösung (resolution)
DAC	digital to analog converter:
	viele Bits/Leitungen → 1 Größe/Leitung mit ('fast' stufenlos) vielen Zuständen
ADC	analog to digal converter:
	stufenlose Größe → binäres Wort
Kennwerte:	
	Wertebereich (unipolar, +/-), 'multiplying DAC',
	Einstellzeit und Abtastrate
	Monotonie
	Linearität (differentielle u. integrale)
	missing Codes (ADC)
	Offset Error
	Gain Error
	Einschwingen(Settling) / Überschwingen
Bsp:	PC-Sound-Karte,
	DigitalStorage-Oscilloscope (DSO)

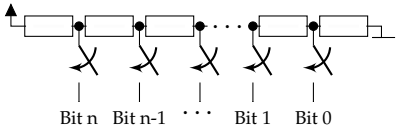
14.8.2 DAC digital to analog converter



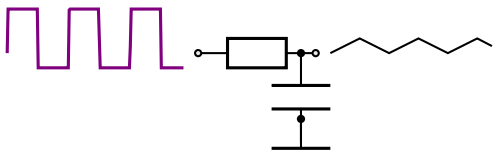
'analog' bedeutet hier *potentiell analog*, d.h. es kann jeder reelle Wert aus einem Interval $[min, max]$ vorkommen;
$\min \leq \text{Wert } \epsilon \mathbb{R} \leq \max$

DAC Ausführung:

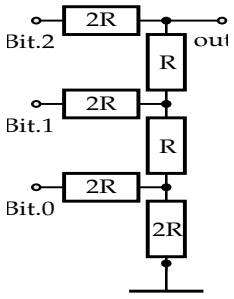
- Widerstandskette und Auswahl-Schalter mit n Stellungen:
  zB. MAX533



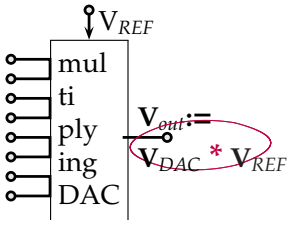
- PWM mit Tiefpass-Filter (typ. mit uC)



- R-2R Netzwerk

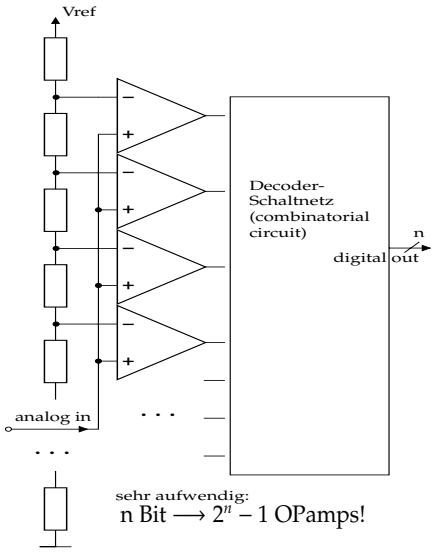


- 'multiplying' DAC:



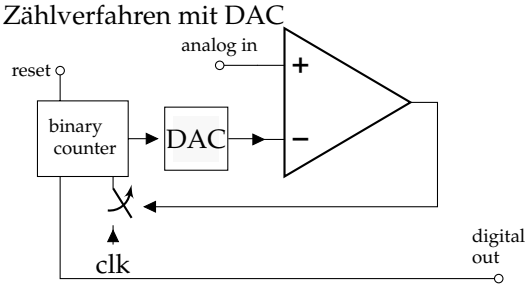
ADC Ausführungen:

- Flash-ADC



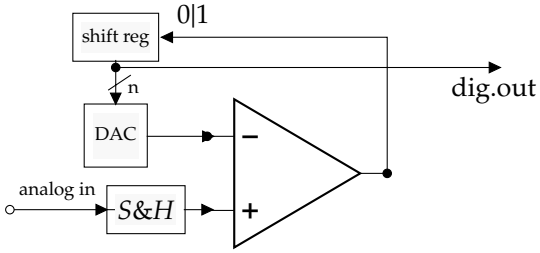
Flash-ADCs werden dort verwendet, wo es schnell, aber nicht so feinstufig zugehen muss, zB. bei Videobild-Farbauszügen mit nachfolgendem Dithering oder als Bestandteil von cyclic- und pipeline-ADC.

- Zählverfahren mit DAC



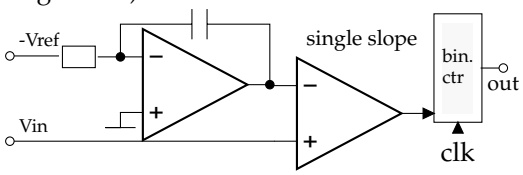
Ein Komparator vergleicht die Eingangsspannung mit der eines Binärzähler-gesteuerten DAC und stoppt bei Gleichstand. Die Binärzahl ist das Ergebnis.

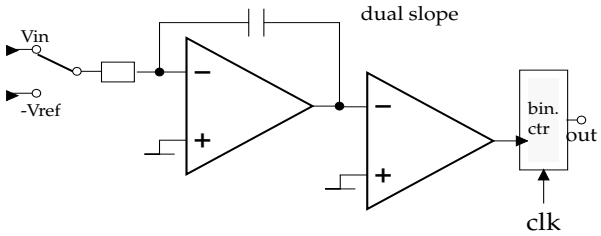
- *Successive Approximation Register 'SAR' ADC*



auch 'Wägeverfahren' – die 'binär-suchen'-Methode

- *Single-, Dual-, Multi- Slope* Verfahren (integrating ADCs)





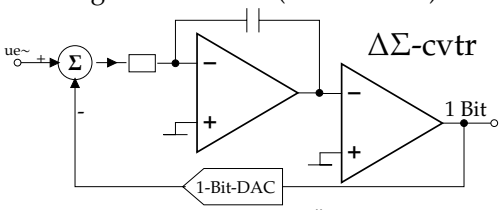
1. Vin auf-integrieren bis Ctr-Overflow: *the input is integrated on to a capacitor for 2^N cycles of the counter*
2. -Vref mit derselben Zeitkonstante RC ab-integrieren bis 0: *the counter is reset and the reference is integrated to discharge the capacitor*
3. Zählerstand ∝ Vin: *The value of the counter when the capacitor is totally discharged is the digitized output*

Dual-, Quad- und Multislope-Umsetzer bestehen im Wesentlichen aus einem Integrator, mehreren Zählern und elektronischen Schaltern. Der Integrator arbeitet mit einem externen, hochwertigen Kondensator, der in zwei oder mehr Zyklen geladen und entladen wird. Beim Zweirampenverfahren (Dual-Slope) wird zunächst der Integratoreingang mit der unbekannten ADU-Eingangsspannung verbunden, und es erfolgt die Ladung über ein fest vorgegebenes Zeitintervall. Für die anschließende Entladung wird der Integrator mit einer bekannten Referenzspannung entgegengesetzter Polarität verbunden. Die benötigte Entladezeit bis zum Erreichen der Spannung null am Integratorausgang wird durch einen Zähler ermittelt; der Zählerstand steht bei geeigneter Dimensionierung unmittelbar für die Eingangsspannung. Die Größe der Kapazität kürzt sich bei diesem Verfahren aus dem Ergebnis heraus. Zur Korrektur des Nullpunktfehlers des ADU wird beim Vierrampenverfahren noch ein weiterer Lade-/Entladezyklus bei kurzgeschlossenem Integratoreingang durchgeführt. Die Referenzspannung ist die bestimmende Größe für die Genauigkeit; das heißt beispielsweise, dass thermisch bedingte Schwankungen vermieden werden müssen. Mehrrampen-Umsetzer

- sind langsam,
- benötigen keine Abtast-Halte-Schaltung,
- bieten hohe Auflösung
- sowie gute differentielle Linearität und
- Störsignaldämpfung (Rauschen, Netzbrumm).

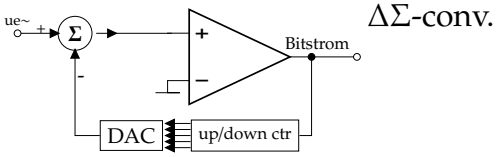
Das typische Einsatzgebiet sind anzeigende Messgeräte (Digitalmultimeter), Umsetzzeiten > 500 ms, die (bei geeigneter Integrationsdauer) 50-Hz-Brumm eliminieren.
(aus → https://de.wikipedia.org/wiki/Analog-Digital-Umsetzer 14Dez'22)

- Delta-Sigma-Converter (AD und DA)

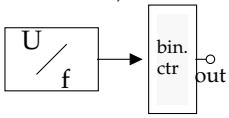


➢ Der Integrator speichert große Änderungen, denen erst nach längeren Bitketten gefolgt werden kann.
➢ 1-Bit-DAC: '1' → positiver Output, '0' → negativer Output
➢ Gleichspannung → '1' und '0' aufeinanderfolgend

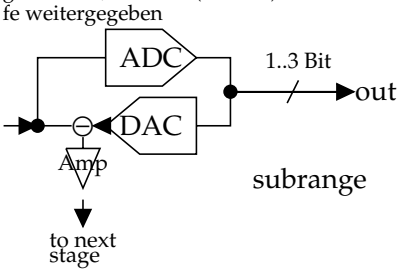
oder:



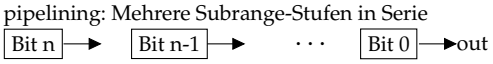
- AD mit Spannungs/Frequenz-Wandler (V/f converter)



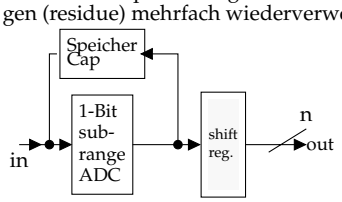
- pipelining-, subranging- u. cyclic ADCs
  sind *essentially the same* – grundsätzlich dasselbe.
  - subranging: Nur ein grober Teil des Wertebereichs wird gewandelt, der Rest (residue) verstärkt an die Folgestufe weitergegeben



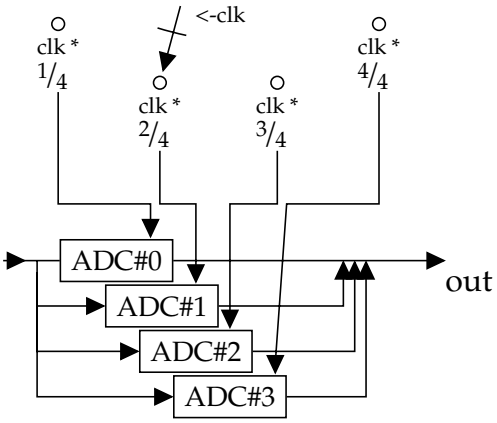
  - pipelining: Mehrere Subrange-Stufen in Serie



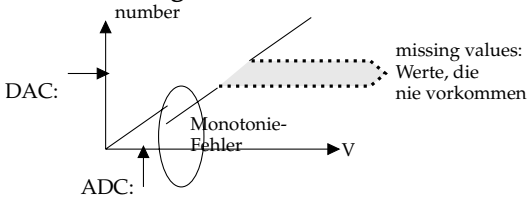
  - cyclic: Anstelle mehrer Stufen wird dieselbe Stufe unter Kondensatorspeicherung der analogen Restspannungen (residue) mehrfach wiederverwendet.



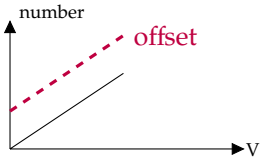
- multiplexing (time interleaved) ADCs
  mehrere identische ADCs werden **zeitversetzt** gestartet. ADC#1 wandelt Sample#1, ADC#2 wandelt Sample#2, . . . nach ADC#n wiederholt sich die Abfolge.



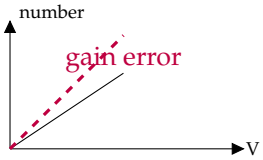
14.8.3 Monotonic error und missing values



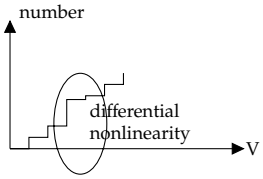
14.8.4 Offset Error



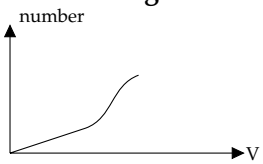
14.8.5 Gain Error



14.8.6 differential non-linearity



14.8.7 integral non-linearity



14.8.8 Wiederkehrgenauigkeit

Streuung bei wiederholten, identischen Wandlungen

14.8.9 Dithering

→ Absichtliche, der Steigerung der Auflösung dienende Störsignal-Beimischung

14.9 UART

14.9.1 serial communication

'parallel'::= Jedes Bit eine **eigene Leitung**.

'serial'(seriell)::= Jedes Bit zeitlich getrennt hintereinander **auf derselben Leitung**.
(dabei meist B0-B1-B2-B3- . . . -Bn = *'LSB first'*)

Mischformen: zB.
'mehrstufige Codierung':
Prinzip — (teilweise) analog/digital conversion:

binary parallel → [DAC] —*serial*/*multilevel*→ [ADC] → binary parallel

oder:
256 tone modulation/demodulation:

binary parallel → [Mod] —*as many tones*/*as numbers*→ [Demod] → binary parallel

oder:
8 tone modulation/demodulation:

binary parallel → [Mod] —*as many tones*/*as Bits*→ [Demod] → binary parallel

oder:
DTMF: 2 Töne aus je einer 4ergruppe (4Bit):

binary parallel → [Mod] —*2 tones of*/*dif ferent choice*→ [Demod] → binary parallel

man kann auch die Amplitude der Töne mehrstufig machen, zB. 0 - 33 -67 - 100% in einem *Schritt* 2 Bit codieren

Tonhöhe und Amplitude zugleich variieren nennt sich QAM Quadratur-Amplituden-Modulation:

4 tones @ 4 levels → 2 x 2 Bit = 4 Bit (per Schritt):

binary parallel → [Mod] —*as many tones*/*as numbers*→ [Demod] → binary parallel

Eine andere **Mischform** ist die Auftrennung in parallelgeschaltete serielle Wege wie

Bit Block → [MUX] —*serial* / . . . / *serial*→ [DEMUX] → Bit Block

Wenn ein Teil der Daten einen parallelen Weg nimmt spricht man auch von *'multiplex'-'demultiplex'*und *'load balancing' (Lastausgleich)*.
Das TCP/IP Protokoll macht das nicht, es sieht dann eine Daten-*'loop'* und trennt alle Parallelwege – man muss TCP/IP *unterlaufen*.

14.9.2 wosisa UART

U niversal:
'universal', weils viele *Baudraten* = Bitraten kann: 45.45, 50, 75, 150, 300, 600, 1200, 2400, 3600, 4800, 7200, 9600, 14400, 19200, 33400, 38400, 56k, ... 115k2 [Baud] = [Bd] (=[Bit / s]), und neuerdings bis > 1Mb/s

A synchronous: Bei jedem Byte wird die Synchronisation neu hergestellt

R eceiver and

T ransmitter

Man baut

- ein Schieberegister, 8-11 Bit lang, parallel loadable
- einen Baudrate-Generator aus:
  Clock-Oscillator
  \+ Clock-Divider, programmable

14.9.3 Fernschreiber

Entstanden ist die asynchrone serielle Übertragung mit dem mechanischen deutschen LorenzAG-Fernschreiber an Telefonleitungen (Telex-Netz)

→ https://de.wikipedia.org/wiki/Fernschreiber
→ https://de.wikipedia.org/wiki/Baudot-Code

→ Tempo: 50 Bit/[s] ('Baud')
→ Leitungs-Ruhezustand ist '1': 40mA Stomfluss.
→ Zeichen beginnen mit einer 0 (Strom-Stop)
→ dann folgen 5 Datenbits und
→ eine Mindestpause von 1.5 Bit
→ LSB first

Baudot-Murray-Code (CCITT-2):
→ 26 Kleinbuchstaben
→ CR (Wagenrücklauf), LF (Zeilenvorschub), Blank (Leerzeichen)
→ Ziffernumschaltung, Buchstabenumschaltung
→ Klingel, wer da?
→ oder:
→ Ziffern 0-9
→ -, =, +, *, /, ?, :, (,), ., „, '

De Ami (die ASA) haben dann 1963 den 7-Bit-ASCII-Code definiert, wo Groß- und Kleinbuchstaben, die Ziffern und Sonderzeichen zugleich enthalten sind.

→ https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange

14.9.4 RS232

definiert die Kommuikation von Computer (DTE data terminal equipment) und Leitungsmodem (DCE data communications equipment)

RS-232, RS-422, RS-485 siehe obelix- Link 'Can/Eth/Usb' (w12aEsCom*.pdf).

'0' ϵ {+3..+15V} nominal +12V
'1' ϵ {-3..-15V} nominal -12V
bei 9600[Bd] (Baudrate) dauert ein Bit 104,2[μs]

- es gibt keine eigene Takt/Clock- Leitung
- es gibt auch kein mitcodiertes Taktsignal
- Start of Transmission wird mit einem *'start bit' (0≡3.0...15.0[V])* gekennzeichnet
- LSB zuerst (Bit0)
- MSB zuletzt (Bit5, Bit7 oder Bit8)
- ein Parity- Bit kann, muss aber nicht, folgen, wahlweise *'even'(gerade)* oder *'odd'(ungerade) Parity(Parität)*
- eine *'stop bit'* genannte Mindestpause≡-3.0...-15.0[V] von 1, 1.5 oder 2 Bit Dauer

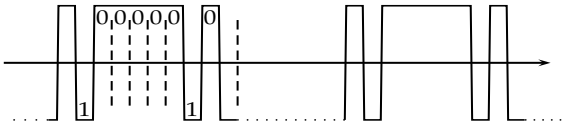
kürzeste Variante:

start bit	data bit0	data bit1	data bit2	data bit3	data bit4	stop bit

längste Variante:

start bit	data bit0	data bit1	data bit2	data bit3	data bit4	data bit5	data bit6	data bit7	pari ty	stop bit1	stop bit2

Ein 'A' im erweiterten 8-Bit-ASCII code $65_D \equiv 41_H \equiv 01000001_B$:



14.9.5 slow RS232 mit rPi3 GPIO

RS232 RX mit einigen Hundert Baud unter Verwendung von GPIO Digitalports und 'parallel threads':

Listing 3: rPi 'rs232rx1.c' @GPIO

```
/*
 * program "rPi232rx1.c"  @ linux-ARM(RaspberryPi)
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <fcntl.h>
#include <pthread.h>
#include <poll.h>
#include <signal.h>
#include <unistd.h>
#include <stdint.h>
#include <sys/timerfd.h>
#include <float.h>

int     fd_MISO, fd_RX;

#define GPIO_DIR          "/sys/class/gpio"
#define GPIO_EXPORT       "/sys/class/gpio/export"
#define GPIO_UNEXPORT     "/sys/class/gpio/unexport"
#define PIN_FILE(a)       "/sys/class/gpio/gpio" a "/value"
#define PIN_DIRECTION(a)  "/sys/class/gpio/gpio" a "/direction"

#define MISO_NR           "9"
#define MISO_VAL          (get_MISO_0_1())
int get_MISO_0_1(){
  int r; char v=0;
  r=pread(fd_MISO, &v, 1, 0L);
  return((v>0x030)? 1:0);
}
#define MOSI_NR           "10"
#define RPI_MOSI_PIN      "/sys/class/gpio/gpio10/value"
#define MOSI_DIRECTION    "/sys/class/gpio/gpio10/direction"

#define SCK_NR            "11"
#define RPI_SCK_PIN       "/sys/class/gpio/gpio11/value"
#define SCK_DIRECTION     "/sys/class/gpio/gpio11/direction"

#define RESET_NR          "27"
#define RPI_RESET_PIN     "/sys/class/gpio/gpio27/value"
#define RESET_DIRECTION   "/sys/class/gpio/gpio27/direction"

#define XTAL_NR           "17"
#define RPI_XTAL_PIN      "/sys/class/gpio/gpio17/value"
#define XTAL_DIRECTION    "/sys/class/gpio/gpio17/direction"

#define PWR_NR            "22"
#define RPI_PWR_PIN       "/sys/class/gpio/gpio22/value"
#define PWR_DIRECTION     "/sys/class/gpio/gpio22/direction"

#define clrPin(a)         write_dev((a), 0)
#define setPin(a)         write_dev((a), 1)
#define MISO              (fd_MISO)

/*************************************
 * open/close device files
 *************************************/
void psystem(char *s){
  struct timespec ts={0L, 50*1000000L};
  printf("issue [%s]\n", s);
  nanosleep(&ts, NULL);
  system(s);
}
int open_devs(){
  int or;
  psystem("echo \"" MISO_NR "\">" GPIO_EXPORT);
  psystem("echo \"in\" >" PIN_DIRECTION(MISO_NR));
  if(0> (fd_MISO = open( PIN_FILE(MISO_NR), O_RDONLY)))  perror("MISO  pin " PIN_FILE(MISO_NR))
      ;
  or=fd_MISO;
  printf("opened. %02X\n", or);
  if(0>or){ perror( PIN_FILE(MISO_NR) " MISO failed"); abort();}
  fd_RX=fd_MISO;
}
int close_devs(){
  close(fd_MISO);
  psystem("echo \"" MISO_NR "\">" GPIO_UNEXPORT);
  printf( PIN_FILE(MISO_NR) " closed.\n");
```

```
}



  /*geordnetes Programm-Ende mit 'q' und Strg+C*/
void exitCode(int snum, siginfo_t *sinfo, void *nix){
  printf(">");fflush(stdout);getchar();
  close_devs();
  printf("*** ->server terminated on signal %i.\n", snum);
  exit(EXIT_SUCCESS);
}
void *exit_on_Q(void*nix){
  while( NULL == strchr("QqCc\003",getchar()) );
  exitCode(-12345, NULL, NULL);
}
#define INSTALL_EXIT_FUNCTION() {\
    struct sigaction sa;          \
    sa.sa_handler       = NULL; \
    sa.sa_sigaction     = exitCode;     \
    sa.sa_flags         = SA_SIGINFO;   \
    sigemptyset(&sa.sa_mask);   \
    if(sigaction(SIGABRT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGALRM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGHUP , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGINT , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGQUIT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTERM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTSTP, &sa, NULL)) perror("setting sigaction() failed.");    \
    pthread_t   THx;    \
    pthread_create( &THx, NULL, exit_on_Q, NULL);       \
    alarm(3000); /*stop after 3000[s] !!! */    \
  }




unsigned char rxcode;   //global!
int dmode=0,smode=0;            //global!
int pipefd1[2], pipefd2[2];     //global!

void *printThread01(void *parm){
  char c;
  for(;;){
    read(pipefd1[0],&c,1);
    if(!smode)printf("\tRX waiting...\n");
  }
}
void *printThread02(void *parm){
  char c;
  for(;;){
    read(pipefd2[0],&c,1);
    if(!smode)printf("\t%02X", rxcode);
  }
  //pthread_exit(NULL);
}

/*************************************
 * async ser.RX  //ex:MISO_'interrupt'
 *************************************/
pthread_t isrPtr;
void *isr_RX(void *parm){
  int i,n,timerfd1,timerfd8;
  unsigned char c1,samp[1000];
  double dt, du,duMin=DBL_MAX,duMax=0.0;
  pthread_t thp1, thp2;
  struct timespec t1,t2,t3,t4,
         ts1={0L,100000L},
         ts4={0L,2500L*1000L};
  struct itimerspec its0={{0L,0L}, {0L, 0L}},
         its1={{0L,1667000L}, {0L,2400000L}},
         its8={{0L,208000L}, {0L,208000L}};
  char   tmrbuf[10];
  uint64_t timerrv;
  timerfd1=timerfd_create(CLOCK_REALTIME,0);
  timerfd8=timerfd_create(CLOCK_REALTIME,0);
  pipe(pipefd1);
  pipe(pipefd2);
  pthread_create(&thp1,NULL,printThread01,NULL);
  pthread_create(&thp2,NULL,printThread02,NULL);
  for(;;){
    clock_gettime(CLOCK_REALTIME, &t1);
    dprintf(pipefd1[1], "1"); //signal to print01()-thread
    for(;;){
      n=pread( fd_RX, &c1, 1, 0L);
```

```
      if(c1>0x030) break;
      nanosleep(&ts1,NULL);//Stopbit polling
    }
    for(;;){
      n=pread( fd_RX, &c1, 1, 0L);
      if(c1<0x031) break;
      nanosleep(&ts1,NULL);//Startbit polling
    }
    if(dmode){
      timerfd_settime(timerfd8,0,&its8,NULL);
      for(i=0;i<88;i++){
        n=pread( fd_RX, &c1, 1, 0L);
        samp[i]= c1;
        if(i>11 && i<72 && (!((i+4)%8))){rxcode= ((rxcode>>1) | ((c1>'0') ? 0x080:0));}
        read(timerfd8,&timerrv,sizeof(uint64_t));
      }
      timerfd_settime(timerfd8,0,&its0,NULL);
      for(i=0;i<88;i++) putchar(samp[i]);
      printf("\n     %8c%8c%8c%8c%8c%8c%8c%8c",
                samp[12],samp[20],samp[28],samp[36],samp[44],samp[52],samp[60],samp[68]);
      duMin=duMax=0.0;
    }else{
      timerfd_settime(timerfd1,0,&its1,NULL);
      clock_gettime(CLOCK_REALTIME, &t1);
      read(timerfd1,&timerrv,sizeof(uint64_t));
      clock_gettime(CLOCK_REALTIME, &t4);
      for(i=0;i<8;i++){
        n=pread( fd_RX, &c1, 1, 0L);
        rxcode= ((rxcode>>1) | ((c1>'0') ? 0x080:0));//0x30=='0', 0x31=='1'
        dprintf(pipefd2[1], "1"); //signal to print02()-thread
        read(timerfd1,&timerrv,sizeof(uint64_t));
        memcpy(&t3,&t4,sizeof(t3));
        clock_gettime(CLOCK_REALTIME, &t4);
        du = (t4.tv_sec-t3.tv_sec) + (t4.tv_nsec-t3.tv_nsec)/1000000000.0;
        if(du<duMin)duMin=du; if(du>duMax)duMax=du;
      }
    }
    timerfd_settime(timerfd1,0,&its0,NULL);
    if(smode){
      printf(", %02X", rxcode); fflush(stdout);
    }else{
      clock_gettime(CLOCK_REALTIME, &t2);
      dt = (t2.tv_sec-t1.tv_sec) + (t2.tv_nsec-t1.tv_nsec)/1000000000.0;
      printf("\n%02X==RXcode t2-t1=%lf[s] t4-t3=%lf[s] min=%lf max=%lf\n-->",
                rxcode, dt, du, duMin, duMax);
    }
    nanosleep(&ts4,NULL);//Ruhepause(Stopbit
  }
}



//C++++++++||++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//         ||
//       \  ||  /
//        \ || /
//         \  /
//          \/
//C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//C+++   M A I N       Procedure
//C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
main(int argc, char *argv[]){
  int   i;

  if(argc<2){
    printf("usage: %s <'-x' | '-d'>\n"
              " -d = debug\n -x=exec\n q...quit\n", argv[0]);
    abort();
  }
  for(i=0; i<argc; i++){
    printf("argv[%i]=[%s]\n", i, argv[i]);
  }
  if(!strcmp("-d",argv[1]))dmode=1;
  if(!strcmp("-s",argv[1]))smode=1;

  open_devs();
  INSTALL_EXIT_FUNCTION();
  pthread_create( &isrPtr, NULL, isr_RX, NULL);

l_ende:
  printf("Return=terminate>");fflush(stdout);getchar();
  close_devs();
  return (0);
}
```

```
/*
RfIc0: rPi232rx1.c new, Umbau
        RfIck-RfIi0,
RfIjf-RfIkf,
RfJ73-RfJ80,
RfKi0: RxSelfTrigger wieder aussa
       RfKe0-f0,RfKi0-nk,
RfN60-80,
RfO8p-9f,RfOa3-ff: ATN13-'OSCCAL':=$4D RCoscillator trim => COMM funktioniert!!!
RfOh0-np: codeCosmetics
RfP80-a5: codeCosmetics
*/
```

14.10 SPI

Das Serial Peripheral Interface (SPI) ist ein im Jahr 1987 von Susan C. Hill Et al., damals bei dem Halbleiterhersteller Motorola (heute NXP Semiconductors), entwickeltes Bus-System und stellt einen „lockeren“ Standard für einen synchronen seriellen Datenbus (Synchronous Serial Port) dar, mit dem digitale Schaltungen nach dem Master-Slave-Prinzip miteinander verbunden werden können.

Ein ähnliches Bus-System unter der Bezeichnung Microwire existiert von National Semiconductor.

Die drei gemeinsamen Leitungen, an denen jeder Teilnehmer angeschlossen ist, sind:

SCLK (englisch Serial Clock) auch SCK, wird vom Master zur Synchronisation ausgegeben

MOSI oder SIMO (englisch Master Output, Slave Input)

MISO oder SOMI (englisch Master Input, Slave Output)

Die Datenleitungen werden manchmal auch SDO (englisch Serial Data Out) und SDI (englisch Serial Data In) genannt, wobei die Benennung meistens aus der Sicht des jeweiligen Busteilnehmers erfolgt, so dass hier die Leitungen über Kreuz verbunden werden müssen. Statt SDI mit SDI und SDO mit SDO zu verbinden, müssen jeweils SDI mit SDO der Gegenstelle verbunden werden.

Eine oder mehrere mit logisch-0 aktive Chip-Select-Leitungen, welche alle vom Master gesteuert werden und von denen je eine Leitung pro Slave vorgesehen ist. Diese Leitungen werden je nach Anwendung unterschiedlich mit Bezeichnungen wie SS, CS, STE oder CE für Slave Select, Chip Select, Slave Transmit Enable oder Chip Enabler bezeichnet, oft noch in Kombination mit einer Indexnummer zur Unterscheidung. Es gibt auch spezielle Anwendungen, bei denen sich mehrere Slaves eine Leitung teilen.

Vollduplexfähig
Viele Einstellmöglichkeiten, wie mit welcher Taktflanke ausgegeben oder eingelesen wird
Wortlänge Übertragung: MSB oder LSB zuerst
Unterschiedliche Taktfrequenzen bis in den MHz-Bereich sind zulässig. Vielfältige Einsatzmöglichkeiten in Audio- und Messanwendungen, zur Datenübertragung zwischen Mikrocontrollern.

Viele Einstellungsmöglichkeiten sind unter anderem deshalb erforderlich, weil die Spezifikation für den SPI-Bus in vielen Eigenschaften nicht festgelegt ist, wodurch verschiedene, zueinander inkompatible Geräte existieren. Häufig ist beispielsweise für jeden angeschlossenen Schaltkreis eine eigene Konfiguration des steuernden Mikrocontrollers (Master des SPI-Bus) erforderlich.

Viele Mikrocontroller, wie die meisten AVR, erlauben über den SPI-Bus eine In-System-Programmierung (kurz auch ISP).

Protokollablauf und Einstellmöglichkeiten

An den Bus können so viele Teilnehmer angeschlossen werden, wie Slave-Select-Leitungen vorhanden sind, zuzüglich des genau einen Masters, der seinerseits das Clock-Signal an SCK erzeugt. Der Master legt mit der Leitung „Slave Select“ fest, mit welchem Slave er kommunizieren will. Wird sie gegen Masse gezogen, ist der jeweilige Slave aktiv und „lauscht“ an MOSI, bzw. legt er seine Daten im Takt von SCK an MISO. Es wird ein Wort vom Master zum Slave und ein anderes Wort vom Slave zum Master transportiert.

Ein Protokoll für die Datenübertragung wurde von Motorola zwar nicht festgelegt, doch haben sich in der Praxis vier verschiedene „Modi“ durchgesetzt. Diese werden durch die Parameter "Clock Polarity"(CPOL) und "Clock Phase"(CPHA) festgelegt. Bei CPOL=0 ist der Clock Idle Low, bei CPOL=1 ist der Clock Idle High. CPHA gibt nun an, bei der wievielten Flanke die Daten übernommen werden sollen. Bei CPHA=0 werden sie bei der ersten Flanke übernommen, nachdem SS auf Low gezogen wurde, bei CPHA=1 bei der zweiten. Somit werden die Daten bei CPOL=0 und CPHA=0 mit der ersten Flanke übernommen, die nur eine steigende Flanke sein kann. Bei CPHA=1 wäre es die zweite, also eine fallende Flanke. Bei CPOL=1 ist es folglich genau andersherum, bei CPHA=0 fallende Flanke und bei CPHA=1 steigende Flanke.

Der Slave legt bei CPHA=0 seine Daten schon beim Runterziehen von SS an MISO an, damit der Master sie beim ersten Flankenwechsel übernehmen kann. Bei CPHA=1 werden die Daten vom Slave erst beim ersten Flankenwechsel an MISO gelegt, damit sie beim zweiten Flankenwechsel vom Master übernommen werden können. Der Master hingegen legt seine Daten immer zum gleichen Zeitpunkt an, meist kurz nach der fallenden Flanke von SCK.

Mit jeder Taktperiode wird ein Bit übertragen. Beim üblichen Bytetransfer sind also acht Taktperioden für eine vollständige Übertragung nötig. Es können auch mehrere Worte hintereinander übertragen werden, wobei in der Spezifikation nicht festgelegt ist, ob zwischen jedem Wort das SS-Signal kurz wieder auf High gezogen werden muss. Eine Übertragung ist beendet, wenn das Slave-Select-Signal endgültig auf High gesetzt wird.

(aus https://de.wikipedia.org/wiki/Serial_Peripheral_Interface)

Das Serial Peripheral Interface, kurz SPI oder auch Microwire genannt, ist ein Bussystem bestehend aus drei Leitungen für eine serielle synchrone Datenübertragung zwischen verschiedenen ICs.

Der Bus besteht aus folgenden Leitungen
MOSI (Master Out -> Slave In) auch SDO (Serial Data Out) oder DO

MISO (Master In <- Slave Out) auch SDI (Serial Data In) oder DI
SCK (Serial Clock) - Schiebetakt

Zusätzlich zu diesen drei Leitungen wird für jeden Slave eine Slave Select (SS) oder auch Chip Select (CS) genannte Leitung benötigt, durch die der Master den Slave zur aktuellen Kommunikation selektiert. Dies geschieht dadurch, dass der Master die SS/CS-Leitung von High nach Low zieht. Oft ist mit dieser Aktivierung durch den Master auch eine Benachrichtigung für den Slave verbunden mit der ihm mitgeteilt wird, dass jetzt eine Nachricht beginnt, das nächste Byte also zum Beispiel als Kommando aufzufassen ist.

Die Übertragung geschieht so, dass der Master seine Datenleitung (MOSI) auf den Pegel des nächsten Bits bringt und dann an der SCK Leitung einen Puls ausgibt. Gleichzeitig wird vom Master der Pegel an der Datenleitung vom Slave zum Master überwacht und ihr Zustand als nächstes einzulesendes Bit aufgefasst. Üblicherweise gibt es zumindest beim Master mehrere Einstellungen, die festlegen, welches der Grundzustand dieser SCK Leitung sein soll und welche Flanke des Taktes zur Datenübernahme herzunehmen ist (die steigende oder die fallende). Bei einigen Slaves ist diese Einstellung ebenfalls möglich, oft ist es aber so, dass per SPI anzusprechende IC eine feste Einstellung benutzen, an die sich der Master anpassen muss.

Für den SPI-Bus gibt es kein festgelegtes Protokoll. Die Taktpolarität (CPOL) und Phase (CPHA) können ebenfalls von Slave zu Slave unterschiedlich sein. Der SPI-Bus kann mit einer Taktfrequenz von vielen Megahertz betrieben werden. Es gibt viele verschiedene ICs die als Slave an dem SPI-Bus betrieben werden können, diese gehen von einfachen Schieberegistern bis hin zu RTCs oder Displaytreibern mit vorgegebenem Protokoll. Unter anderem werden die meisten AVR-Microcontroller von Atmel über SPI programmiert, siehe dazu AVR In System Programmer.

Übertragungsprinzip

Bei SPI verlieren die Begriffe 'Sender' und 'Empfänger' ihre Bedeutung. Das Übertragungsprinzip funktioniert so, dass gleichzeitig 1 Bit vom Master zum Slave und 1 Bit vom Slave zum Master übertragen wird. Die Übertragung ist eher mit dem Begriff Äustausch von Bitsßu beschreiben, als dass es sich um ein Senden bzw. Empfangen handelt.

> dieses Gefasel ist ein Blödsinn
> so etwas heisst fachlich **richtig**
> ganz einfach **vollduplex**

Je nach Sichtweise könnte man auch sagen, dass sowohl Master als auch Slave jederzeit sowohl Sender als auch Empfänger sind. Der einzige Unterschied zwischen den Busteilnehmern besteht darin, dass beim SPI Bus jegliche Kommunikation immer vom Master ausgeht. Er hat die Verwaltung der SCK Leitung inne und wenn der Master keine SCK Pulse generiert, dann kann ein Slave auch nichts übertragen. Die Generierung der SCK Pulse wird dabei üblicherweise dadurch angestossen, dass der Master einfach angewiesen wird, 1 Byte über SPI auszugeben. Des Weiteren kann ein Slave auch eine SPI Übertragung nicht hinauszögern, so wie es beim I2C-Bus der Fall ist. Wenn vom Master die SCK Pulse kommen, dann muss ein Slave die MISO Leitung bedienen. Er hat keine Möglichkeit dies nicht zu tun, selbst wenn er noch gar kein Ergebnis für den Master vorliegen hat. Selbst wenn der Slave die MISO Leitung nicht bedient, der Master wird zum per Modus eingestellten Zeitpunkt die Polarität der MISO Leitung auswerten und sich daraus das nächste Bit bestimmen.

Erfolgt eine Abfrage eines Masters an einen Slave dergestalt, dass der Master ein Kommando an den Slave sendet, auf das der Slave mit einem Ergebnis zu antworten hat, dann sind dazu immer mindestens 2 Übertragungen notwendig. Im ersten Byte-Austausch übermittelt der Master sein Kommando an den Slave (der zu diesem Zeitpunkt logischerweise noch kein Ergebnis vorliegen haben wird, er kennt ja das Kommando noch nicht) und im zweiten Byte-Austausch überträgt der Master einfach nur 1 Byte (meistens genügt da einfach irgendein Bytewert) um dem Slave seinerseits Gelegenheit zu geben, das zuvor angefragte Ergebnis zum Master zu übertragen. Beachten sollte man, dass ein Slave auch des öfteren etwas Zeit benötigen wird, um ein gerade erhaltenes Kommando auszuwerten und die entsprechenden Ergebnisse bereit zu stellen. Ein Master wird also gut daran tun, nach dem Byteaustausch zum Zwecke der Kommandoübermittlung eine kleine Pause einzulegen, ehe er dann mit dem zweiten Byteaustausch das Ergebnis abholt.

SPI-Modi

Wie schon angesprochen gibt es für das SPI verschiedene Möglichkeiten Polarität und Phase des Taktes einzustellen. Folgende vier Modi sind möglich:

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

CPOL (Clock Polarity)
0: Takt ist in Ruhe LOW, ein Wechsel auf HIGH zählt als steigende Taktflanke
1: Takt ist invertiert: in Ruhe HIGH, ein Wechsel auf LOW zählt als steigende Taktflanke
CPHA (Clock Phase)
0: Daten werden bei steigender Taktflanke (=abh. von CPOL) eingelesen, bei fallender ausgegeben
1: Daten werden bei fallender Taktflanke eingelesen, bei steigender ausgegeben

Man sieht, dass Mode 0 und Mode 3 bzw. Mode 1 und Mode 2 jeweils fast identisch sind. Der einzige Unterschied ist der Pegel des Taktes in

Ruhe. In der Regel sind diese Modi deshalb austauschbar.

CPOL und CPHA lassen sich in den Konfigurationsregistern des Controllers einstellen. Beim AT91SAM hat sich Atmel ein kleines Extra einfallen lassen: hier heißt das Bit zur Einstellung der Clock Phase "NCPHAünd entspricht genau dem invertierten Wert von CPHA.

Es ist problemlos möglich ICs mit verschiedenen SPI-Modi an einem Bus zu betreiben, man muss nur vor dem Aktivieren des Chip Select den jeweils richtigen Modus einstellen.

Falle bei AVR Prozessoren

Bei den AVR-Prozessoren, welche eine SPI Einheit besitzen, ist es zwingend nötig, den SS-Pin der SPI Einheit auf Ausgang zu setzen, wenn man den Prozessor als Master benutzen möchte. Dies geschieht nicht automatisch, da ein auf Eingang geschalteter Pin im Multimaster Modus benötigt wird. Da dieser Fall aber recht selten vorkommt, wird er hier ignoriert. Schaltet man den Pin nicht auf Ausgang, dann hängt es vom extern angelegten Pegel ab, ob die Master-SPI Einheit arbeitet oder nicht. Daher muss man den Pin immer vorher auf Ausgang schalten. In diesem Fall ist der Pegel der SS Leitung dann irrelevant, die SPI Einheit arbeitet immer, sobald sie als Master konfiguriert wurde.

Fehlersuche

Immer wieder kommt es zu Problemen bei der Nutzung von SPI, obwohl diese Schnittstelle recht einfach aufgebaut ist. Oft kommt es u.a. zu dem Effekt, dass nur bestimmte SPI-Frequenzen funktionieren, etwas höher oder niedrigere nicht. Das ist ein klares Zeichen für Fehler. Folgende Punkte sollte man mit dem Oszilloskop prüfen, dabei sollte man die Tastköpfe richtig benutzen.

* Stimmt jeweils der SPI-Modus?
* Gilt nur für AVRs: Ist das SS-Pin vorher als Ausgang geschaltet, wenn der Master Modus verwendet wird?
* Sind die Select-Signale vorhanden?
* Sind die erreichten Pegel ausreichend?
* Sind die Taktflanken sauber genug? Siehe Wellenwiderstand.
* Sind die Daten während der Datenübernahme stabil?
* Ist die Zeit zwischen Select-Signal und Übertragungsbeginn ausreichend?
* Ist das SPI-Modul im Power Reduction Register (PRR) versehentlich deaktiviert?

(aus https://www.mikrocontroller.net/articles/Serial_Peripheral_Interface)

14.11 IIC

I²C, für englisch Inter-Integrated Circuit, im Deutschen gesprochen als I-Quadrat-C oder englisch I-Squared-C, ist ein 1982 von Philips Semiconductors (heute NXP Semiconductors) entwickelter serieller Datenbus.

Er wird hauptsächlich geräteintern für die Kommunikation zwischen verschiedenen Schaltungsteilen benutzt, z. B. zwischen einem Controller und Peripherie-ICs. Das ursprüngliche System wurde von Philips in den frühen 1980er Jahren entwickelt, um verschiedene Chips in Fernsehgeräten einfach steuern zu können. Seit Mitte der 1990er Jahre wird I2C auch von einigen Wettbewerbern zur Bezeichnung von Philips-kompatiblen I2C-Systemen verwendet, darunter Siemens AG (später Infineon Technologies AG), NEC, STMicroelectronics, Motorola (später Freescale), Intersil etc. Es gibt insgesamt über 1000 verschiedene ICs von über 50 Herstellern (Stand 2014).

Atmel führte aus lizenzrechtlichen Gründen die heute auch von einigen anderen Herstellern verwendete Bezeichnung TWI (Two-Wire-Interface, englisch für Zweidraht-Schnittstelle) ein; technisch sind TWI und I²C praktisch identisch. Allerdings ist das ursprüngliche Patent am 1. Oktober 2006 ausgelaufen, so dass keine Lizenzgebühren für die Benutzung von I2C mehr anfallen. I2C ist auch kein eingetragenes Markenzeichen von NXP Semiconductors, Markenschutz besteht lediglich für das Logo.

Geschichte

Der Bus wurde 1982 von Philips eingeführt zur geräteinternen Kommunikation zwischen ICs in z. B. CD-Spielern und Fernsehgeräten. Dazu wurde die Mikrocontroller-Familie MAB8400 entwickelt, die einen I²C-Bus-Controller enthielt. Die erste standardisierte Spezifikation 1.0 wurde 1992 veröffentlicht. Diese ergänzte den ursprünglichen Standard mit 100 kbit/s um einen neuen „schnellen“ Modus (Fast-mode) mit 400 kbit/s und erweiterte den Adressraum um einen 10-Bit-Modus, so dass statt der ursprünglichen 112 Knoten seitdem bis zu 1136 unterstützt werden.

Mit Version 2.0 aus dem Jahr 1998 kam ein „Hochgeschwindigkeitsmodus“ (Hs-mode) mit max. 3,4 Mbit/s dazu, wobei die Strom- und Spannungsanforderungen in diesem Modus gesenkt wurden. Version 3.0 von 2007 führte einen weiteren Modus „Fast-mode Plus“ (Fm+) mit bis zu 1 Mbit/s ein, der im Gegensatz zum Hs-mode dasselbe Protokoll verwendet wie die 100- und 400-kbit/s-Modi.

Im Jahr 2012 wurde mit der Spezifikation V.4 ein noch schnellerer Modus „Ultra Fast-mode“ (Ufm) eingeführt, der unidirektionale Übertragungsraten bis zu 5 Mbit/s unterstützt. Im selben Jahr wurden mit der aktuellen V.5 einige Fehler der Vorgängerversion korrigiert. Im April 2014 erschien V.6, die erneut Fehler korrigierte.

Bussystem

I2C ist als Master-Slave-Bus konzipiert. Ein Datentransfer wird immer durch einen Master initiiert; der über eine Adresse angesprochene Slave reagiert darauf. Mehrere Master sind möglich (Multimaster-Betrieb). Wenn im Multimaster-Betrieb ein Master-Baustein auch als Slave arbeitet, kann ein anderer Master direkt mit ihm kommunizieren, indem er ihn als Slave anspricht.

Elektrische Definition

I2C benötigt zwei Signalleitungen: Takt- (SCL = Serial Clock) und Datenleitung (SDA = Serial Data). Beide liegen mit den Pull-up-Widerständen RP an der Versorgungsspannung VDD. Sämtliche daran angeschlossene Geräte haben Open-Collector-Ausgänge, was zusammen mit den Pull-up-Widerständen eine Wired-AND-Schaltung ergibt. Der High-Pegel soll mindestens 0,7 × VDD betragen, und der Low-Pegel soll bei höchstens 0,3 × VDD liegen. Die im Bild nicht eingezeichneten Serienwiderstände RS an den Eingängen der Geräte sind optional und werden als Schutzwiderstände verwendet. Der I2C-Bus arbeitet mit positiver Logik, d. h. ein High-Pegel auf der Datenleitung entspricht einer logischen „1“, der Low-Pegel einer „0“.

Takt und Zustände des Busses

Zeitverhalten am I²C-Bus: Zwischen dem Start-Signal (S) und dem Stopp-Signal (P) werden die Datenbits B1 bis BN übertragen.

Der Bustakt wird immer vom Master ausgegeben. Für die verschiedenen Modi ist jeweils ein maximal erlaubter Bustakt vorgegeben. In der Regel können aber auch beliebig langsamere Taktraten verwendet werden, falls diese vom Master-Interface unterstützt werden. Einige ICs (z. B. Analog-Digital-Umsetzer) benötigen jedoch eine bestimmte minimale Taktfrequenz, um ordnungsgemäß zu funktionieren.

	Max.Bitrate	Richtung
StandardMode(Sm)	0,1 Mbit/s	bi
FastMode(Fm)	0,4 Mbit/s	
FastModePlus(Fm+)	1,0 Mbit/s	
HighSpeedMode(Hs)	3,4 Mbit/s	
UltraFastMode(UFm)	5,0 Mbit/s	uni

Wenn der Slave mehr Zeit benötigt, als durch den Takt des Masters vorgegeben ist, kann er zwischen der Übertragung einzelner Bytes die Taktleitung auf „low“ halten (Clock-Stretching) und so den Master bremsen. In der Spezifikation eini-

ger Slave-Bausteine wird explizit erklärt, dass sie kein Clock Stretching anwenden. Dementsprechend gibt es auch Bustreiber-Bausteine, die so ausgelegt sind, dass sie das Taktsignal nur in eine Richtung übertragen können.

Daten (Einzelbits) sind nur gültig, wenn sich ihr logischer Pegel während einer Clock-High-Phase nicht ändert. Ausnahmen sind das Start-, Stop- und Repeated-Start-Signal. Das Start-Signal ist eine fallende Flanke auf SDA, während SCL high ist, das Stop-Signal ist eine steigende Flanke auf SDA, während SCL high ist. Repeated-Start sieht genauso aus wie das Start-Signal.

Eine Dateneinheit besteht aus 8 Datenbits = 1 Oktett (welche protokollbedingt entweder als Wert oder als Adresse interpretiert werden) und einem ACK-Bit. Dieses Bestätigungsbit (Acknowledge) wird vom Slave durch einen Low-Pegel auf der Datenleitung während der neunten Takt-High-Phase (welche nach wie vor vom Master generiert wird) und als NACK (für engl. not acknowledge) durch einen High-Pegel signalisiert. Der Slave muss den Low-Pegel an der Datenleitung anlegen, bevor SCL auf high geht, andernfalls lesen weitere eventuelle Teilnehmer ein Start-Signal.

Adressierung

Eine Standard-I2C-Adresse ist das erste vom Master gesendete Byte, wobei die ersten sieben Bit die eigentliche Adresse darstellen und das achte Bit (R/W-Bit) dem Slave mitteilt, ob er Daten vom Master empfangen soll (Low: Schreibzugriff) oder Daten an den Master zu übertragen hat (High: Lesezugriff). I^2C nutzt daher einen Adressraum von 7 Bit, was bis zu 112 Knoten auf einem Bus erlaubt (16 der 128 möglichen Adressen sind für Sonderzwecke reserviert).

Jedes I2C-fähige IC hat eine (üblicherweise vom Hersteller) festgelegte Adresse, von der in der Regel eine modellabhängige Anzahl der untersten Bits (LSB) über spezielle Eingangspins des ICs individuell konfiguriert werden können. Hierdurch wird es möglich, mehrere ICs dieses Typs am selben I2C-Bus zu betreiben, ohne dass es zu Adresskonflikten kommt. Lassen sich Adresskonflikte nicht vermeiden, so müssen die entsprechenden ICs mit getrennten I2C-Bussen angesteuert oder temporär vom Bus getrennt werden.

Wegen Adressknappheit wurde später eine 10-Bit-Adressierung eingeführt. Sie ist abwärtskompatibel zum 7-Bit-Standard durch Nutzung von 4 der 16 reservierten Adressen. Beide Adressierungsarten sind gleichzeitig verwendbar, was bis zu 1136 Knoten auf einem Bus erlaubt.

Übertragungsprotokoll

Der Beginn einer Übertragung wird mit dem Start-Signal vom Master angezeigt, dann folgt die Adresse. Diese wird durch das ACK-Bit vom entsprechenden Slave bestätigt. Abhängig vom R/W-Bit werden nun Daten byteweise geschrieben (Daten an Slave) oder gelesen (Daten vom Slave). Das ACK beim Schreiben wird vom Slave gesendet und beim Lesen vom Master. Das letzte Byte eines Lesezugriffs wird vom Master mit einem NACK quittiert, um das Ende der Übertragung anzuzeigen. Eine Übertragung wird durch das Stop-Signal beendet. Alternativ kann auch ein Repeated-Start am Beginn einer erneuten Übertragung gesendet werden, ohne die vorhergehende Übertragung mit einem Stop-Signal zu beenden.

Alle Bytes werden dabei „Most Significant Bit First“ übertragen.

Für den High-Speed-Mode wird zuerst im Fast oder Standard Mode ein Master-Code geschickt, bevor auf die erhöhte Frequenz umgeschaltet wird. Dadurch wird zum einen der High-Speed-Mode signalisiert, zum anderen haben nicht High-Speed-taugliche Busteilnehmer die Chance, innerhalb ihrer Spezifikation zu erkennen, dass sie nicht angesprochen wurden. Im Multimasterbetrieb muss jeder Busmaster einen eigenen Master-Code benutzen. So ist sichergestellt, dass die Busarbitrierung (s. u.) abgeschlossen ist, bevor in den High-Speed-Mode gewechselt wird.

Arbitrierung im Multimaster-Betrieb

Die Arbitrierung (Zugriffsregelung auf den Bus) ist durch die Spezifikation geregelt: Der Bus ist zwischen Start- und Stoppsignal belegt. Busmaster müssen daher immer auf Start- und Stoppsignale achten, um den Überblick über den Busstatus zu behalten. So können sie warten, bis der Bus frei ist, sollte (evtl. unvorhergesehen) eine Übertragung anstehen.

Sollten mehrere Busmaster gleichzeitig mit einer Transaktion starten wollen, so sehen sie den Bus als frei an und starten gleichzeitig mit der Übertragung. Sind die Master unterschiedlich schnell, erfolgt die Übertragung nun zunächst so schnell, wie der langsamste der beteiligten Busmaster arbeitet, da das Taktsignal eines langsameren Busmasters per Clock-Stretching die schnelleren ausbremst. Alle Busmaster lauschen auf die von ihnen selbst gesendeten Daten. In dem Augenblick, wenn ein Busmaster eine „0“ und ein anderer eine „1“ übertragen will, nimmt die Bus-Leitung (aufgrund der Wired-And-Schaltung aller Busteilnehmer) „0“-Pegel an. Gemäß dem I^2C-Protokoll verlieren in diesem Augenblick die Busmaster mit der „1“ den Bus, ziehen sich zurück und warten auf das Stoppsignal, um dann ihr Glück erneut zu versuchen. Die anderen Busmaster machen weiter, bis schließlich nur noch einer übrigbleibt. Sollte ein unterlegener Busmaster-

Baustein auch Slave-Dienste anbieten, muss er allerdings gleichzeitig darauf achten, ob der gewinnende Busmaster ihn gerade ansprechen will und daher gerade dabei ist, ihn zu adressieren.

Das Verfahren geht so weit, dass gar keine Arbitrierung stattfindet, wenn mehrere Busmaster zufällig – über mehrere Bytes hinweg von Anfang bis zum Abschluss ihrer jeweiligen Transaktionen hinweg – identische Daten an denselben Slave-Baustein senden: Die betreffenden Busmaster merken nichts voneinander – eventuelles Clock-Stretching durch einen langsameren Master ist gemäß Protokoll nicht von Clock-Stretching durch den Slave zu unterscheiden; der angesprochene Slave-Baustein kommuniziert mit den betreffenden Busmastern gleichzeitig, ohne dass es von den Beteiligten bemerkt wird. Diese Tatsache ist zu berücksichtigen, und ihr muss, sofern sie sich störend auswirken könnte, anderweitig Abhilfe geschaffen werden.

Verwendung

Serielles EEPROM mit I2C-Bus von STMicroelectronics

Eine Eigenschaft von I2C ist die Tatsache, dass ein Mikrocontroller ein ganzes Netzwerk an integrierten Schaltungen mit nur zwei I/O-Pins und einfacher Software kontrollieren kann. Busse dieses Typs wurden realisiert, da ein nicht unerheblicher Teil der Kosten einer integrierten Schaltung und der verwendeten Leiterplatte von der Größe des Gehäuses und der Anzahl der Pins abhängt. Ein großes IC-Gehäuse hat mehr Pins, braucht mehr Platz auf der Leiterplatte und hat mehr Verbindungen, die versagen können. All das steigert die Produktions- und Testkosten.

Obwohl langsamer als neuere Bus-Systeme, ist I2C wegen des geringen Aufwands vorteilhaft für Peripheriegeräte, die nicht schnell zu sein brauchen. Häufig wird er für die Übertragung von Steuer- und Konfigurationsdaten verwendet. Beispiele sind Lautstärkeregler, Analog-Digital- oder Digital-Analog-Wandler mit niedriger Abtastrate, Echtzeituhren, kleine, nichtflüchtige Speicher oder bidirektionale Schalter und Multiplexer. Auch elektronische Sensoren haben oft einen Analog-Digital-Wandler mit I2C-Schnittstelle integriert.

Während des Betriebes können Chips zum Bus hinzugefügt oder entfernt werden (Hot-Plugging).

I2C wird auch als Basis für ACCESS.bus und VESAs Monitordaten-Interface (Display Data Channel, kurz DDC) benutzt. Der vom Prozessorhersteller Intel für die Kommunikation von Mainboard-Komponenten definierte SMBus ist dem I2C-Bus sehr ähnlich, die meisten ICs erlauben einen Betrieb an beiden Bussen.

Große Bedeutung hatte das I2C-Protokoll in der Vergangenheit im Chipkartenbereich. Die in Deutschland bis Ende 2014 verwendete Krankenversichertenkarte war eine I2C-Karte, d. h. unter den goldenen Kontaktflächen der Chipkarte befand sich ein einfaches I2C-EEPROM, das vom Kartenleser über das I^2C-Protokoll ausgelesen und beschrieben werden konnte.

Stabilität

Das Protokoll des I2C-Bus ist von der Definition her recht einfach, aber auch recht störanfällig. Diese Tatsache schränkt die Verwendung auf störarme Umgebungen ein, wo weder mit Übersprechen, Rauschen, EMV-Problemen noch mit Kontaktproblemen (Stecker, Buchsen) zu rechnen ist. Auch ist er ungeeignet zur Überbrückung größerer Entfernungen, wie es beispielsweise für Feldbusse typisch ist.

Der Bus kann jedoch mit speziellen Treibern auf einen höheren Strom- oder Spannungspegel umgesetzt werden, wodurch der Störabstand und die mögliche Leitungslänge steigen. Ein noch größerer Störabstand ist durch eine Umsetzung auf den physikalischen Layer des CAN-Busses möglich, der mit differentiellen Open-Collector-Signalen arbeitet. Störungen sowohl des SDA- als auch des SCL-Signals resultieren in fehlerhaft übertragenen Daten, die vor allem bei Störungen auf SDA oft nicht erkannt werden können. Lediglich bei geringen, zeitlich begrenzten Störungen, z. B. weit oberhalb der Signalfrequenz, kann das System mittels Signalverarbeitung stabiler gemacht werden.

Die eigentliche I2C-Spezifikation beinhaltet (anders als die SMBus-Spezifikation) keinen Timeout; dadurch kann es unter bestimmten Umständen dazu kommen, dass Busteilnehmer den Bus blockieren. Falls ein Slave-Chip gerade die Datenleitung auf „0“ zieht, während der Master den Transfer (beispielsweise durch einen Reset) abbricht, bleibt die Datenleitung für unbestimmte Zeit auf „0“. Somit bleibt der gesamte I2C-Bus mit allen angeschlossenen Teilnehmern (Chips) blockiert. Daher sollen im Falle eines Resets auch alle Busteilnehmer zurückgesetzt werden, ggf. durch Unterbrechen der Spannungsversorgung. Alternativ wird ein Bus clear durchgeführt: Der Master generiert bis zu neun Taktimpulse; spätestens dann sollte die Datenleitung freigegeben sein. Selbst wenn sich der Slave-Baustein noch mitten in einer Übertragung befindet und die Datenleitung nur freigegeben ist, weil er gerade eine „1“ ausgibt, wird er (bzw. dessen I2C-Komponente) durch das nächste Start-Signal zurückgesetzt. Analog Devices empfiehlt, zunächst ein Stopp-Signal zu senden.

Erweiterungen

Die MIPI Alliance hat 2014 ein I3C genanntes, zu I2C abwärtskompatibles Interface vorgestellt. Es wird als Nachfolger von I2C propagiert. Es enthält Erweiterungen wie eine höhere Übertragungsgeschwindigkeit (High Data Rate – HDR) und kann damit bei mittleren Geschwindigkeiten auch das Serial Peripheral Interface (SPI) ersetzen.

(aus https://de.wikipedia.org/wiki/I2C)

Übertragungsraten

Die genormte Übertragungsrate beträgt beim sogenannten ”standard mode” 100 kbit/s, beim ”fast mode” 400 kbit/s und beim ”fast mode+” 1000kbit/sec. Im sogenannten high-speed mode, der mit etwas anderen Spannungs- und Stromrandbedingungen arbeitet sind es 3,4 MBit/s. Die Raten beziehen sich auf die festgelegten Taktraten. Andere sind möglich, jedoch nicht genormt.

Die reale maximale Datenrate ist infolge des asynchronen Betriebs und der Pausen meistens etwas niedriger. Zudem kann - falls die Taktrate für einen Slave zu hoch ist - die Clock-Leitung auf Null durch ihn auf GND gezogen- und damit die Übertragung verlangsamt werden, was als sog. ”Clock Stretching” bezeichnet wird. Dies ist auf Bit- wie auf Byte-Ebene möglich; ersteres allerdings nicht im high-speed mode.

Reale Beispiele

Um sich an längere Übertragungswege anzupassen, kann man die Taktrate theoretisch beliebig vermindern, jedoch erzeugen einige Bausteine aber irgendwann ein Time-Out. Manche interpretieren einen lange niedergehaltenen Takt auch als Reset. Daher sind Taktraten von unter 1kbps in der Regel nicht denkbar.

Als Beispiel können jedoch mit einer Taktfrequenz von nur 5 kbit/s durchaus mehrere Meter überbrückt werden:

Erfahrungen mit den Atmegas (gemessene I2C-Taktfrequenzen):

16,000MHz - schafft sauber bis zu 700KHz
18,432MHz - schafft sauber bis zu 950KHz

Selbst unter schlechten Bedingungen wie...

50cm 6-adriges Spiralkabel, frei schwingend verbaut in einem Rennwagen als Verbindung zwischen Lenkraddisplay und Amaturenbrett
Nähe zu nicht geschirmten Leitungen mit 12V, GND und 5V PWM zum Dimmen von LEDs
grausamen Umgebungsbedingungen, Fahrten im Regen, Schnee oder bei glühender Sonne und gefühlten 100° C ;-) auf der Rennstrecke.

... hat der Bus ohne Probleme funktioniert.

Um anders herum hohe Bandbreiten zu erreichen, lassen sich krumme I2C-Frequenzen ausserhalb der SPEC verwenden, damit sie besser zu den Controllern und deren Taktfrequenzen passen.

Bausteine

Neben Mikrocontrollern gibt es eine Reihe von Peripheriebausteinen, die per I^2C angeschlossen werden können. Eine gute Anlaufstelle bei der Suche ist die unten angegebene Seite des ’Erfinders’ Philips, heute als NXP bekannt.

serielle EEPROMs

24Cxx

PCF85xx von NXP, 256-2048 Byte

I/O-Portexpander
PCF8574
MCP23008 (8-bit) von Microchip
MCP23017(16-bit) von Microchip

I2C MUX, zum Anschluss von ICs mit gleicher, fester Adresse
PCA9545A

AD-Wandler
12x12 Bit ADC MAX1238
12x10 Bit ADC MAX1138
MAX127 und MAX128 von Maxim, 12bit x8, PDIP24+SSOP28
12x8 Bit ADC MAX1038

DA-Wandler
TDA8444, 8x6Bit

Uhrenbausteine
MCP7940N
PCF8583, mit 256 Bytes RAM
DS1307

LCD-Treiber
PCF8577, 2x32 Segmente

LED-Treiber
HT16K33 (RAM Mapping 16*8 LED) von Holtek
SAA1064 (seit 2005 abgekündigt! Restposten sehr teuer)

Temperatursensoren
DS1621
LM75
SHT21
TMP101 von TexasInstruments
TMP175 von TI (mehr als 8 Bausteine im gleichen Bus möglich)

Drucksensoren
SMD500
BMP085

Beschleunigungssensor
BMA020

Audioverstärker
TPA2016
TPA2026
TDA8594

TPMs (Trusted Platform Module)
SLB 9645

(aus https://www.mikrocontroller.net/articles/I2C)

14.12 CAN

Controller Area Network

Der CAN-Bus (Controller Area Network) ist ein serielles Bussystem und gehört zu den Feldbussen. Er wurde 1983 vom Unternehmen Bosch entwickelt und 1986 zusammen mit Intel vorgestellt. Sein Zweck ist es, Kabelbäume zu reduzieren und hiermit Kosten und Gewicht zu sparen. Zur damaligen Zeit konnte die Gesamtlänge aller Kabel im Kraftfahrzeug ohne CAN bis zu 2 km betragen.

CAN ist als ISO 11898 international standardisiert und definiert die Layer 1 (physische Schicht) und 2 (Datensicherungsschicht) im ISO/OSI-Referenzmodell. Die beiden gängigsten Realisierungen der physischen Schichten sind nach **ISO 11898-2 (Highspeed-CAN)** und **ISO 11898-3 (Lowspeed-CAN).** Sie unterscheiden sich in zahlreichen Eigenschaften und sind nicht zueinander kompatibel.

Übertragungsverfahren

Der CAN-Bus arbeitet nach dem **"Multi-Master-Prinzip"** d. h., er verbindet mehrere gleichberechtigte Steuergeräte. Ein CSMA/CR-Verfahren löst Kollisionen (gleichzeitiger Buszugriff) auf, ohne dass die gewinnende, höher priorisierte Nachricht beschädigt wird. Dazu sind die Bits – je nach Zustand – dominant bzw. rezessiv (ein dominantes Bit überschreibt ein rezessives). Die logische 1 ist rezessiv, kann sich auf dem Bus also nur durchsetzen, solange kein Teilnehmer logisch 0 sendet, logisch entspricht dies einer UND-Verknüpfung, obwohl bei Betrachtung einer der Leitungen für die Spannungspegel eine Wired-OR-Verknüpfung gilt. Die Daten sind NRZ-codiert, mit Bitstopfen zur fortlaufenden Synchronisierung auch von Busteilnehmern mit wenig stabilem Oszillator. Zur Datensicherung kommt zyklische Redundanzprüfung zum Einsatz. Der Bus ist entweder mit Kupferleitungen oder über Glasfaser ausgeführt.

Spannungspegel im Highspeed-CAN-Bus

Im Falle von Kupferleitungen arbeitet der CAN-Bus mit zwei verdrillten Adern, **CAN_HIGH** und **CAN_LOW** (symmetrische Signalübertragung). **CAN_GND (Masse)** als dritte Ader ist optional, jedoch oft zusammen mit einer vierten Ader zur **5-V-Stromversorgung** vorhanden.

Bei höheren Datenraten (Highspeed-CAN) ist der Spannungshub zwischen den beiden Zuständen relativ gering: Im rezessiven Ruhezustand ist die Differenzspannung null **(beide Adern etwa 2,5 V über Masse)**, im dominanten Zustand beträgt sie mindestens 2 V (CAN_HIGH > 3,5 V, CAN_LOW < 1,5 V).

Beim für größere Distanzen geeigneten Lowspeed-CAN kommt ein Spannungshub von 7 V zum Einsatz, indem die rezessiven Ruhepegel auf 5 V (CAN_LOW) und 0 V (CAN_HIGH) gelegt sind. Bei Ausfall einer der beiden Leitungen kann die Spannung der anderen Leitung gegen Masse ausgewertet werden. Bei langsameren Bussen ("Komfort-Bus" z.B. zur Betätigung von Elementen durch den Benutzer) kann ein Eindrahtsystem mit der Karosserie als Masse deshalb reichen. Praktisch wird es meistens doch als Zweidrahtsystem ausgeführt, verwendet aber im Fall eines Aderbruchs den Eindrahtbetrieb als Rückfallebene, um den Betrieb weiterführen zu können. Das nennt sich dann "Limp-Home-Modus" (Deutsch: "nach-Hause-humpeln-Modus").

Topologie

Linearer CAN-Bus

Das CAN-Netzwerk wird als Linienstruktur aufgebaut. Stichleitungen sind in eingeschränktem Umfang zulässig. Auch ein sternförmiger Bus (z. B. bei der Zentralverriegelung im Auto) ist möglich. Diese Varianten haben allerdings im Vergleich zum linienförmigen Bus Nachteile:

Der sternförmige Bus wird meist von einem Zentralrechner gesteuert, da diesen alle Informationen passieren müssen, mit der Folge, dass bei einem Ausfall des Zentralrechners keine Informationen weitergeleitet werden können. Beim Ausfall eines einzelnen Steuergeräts funktioniert der Bus weiter.

Für Stichleitungen und sternförmige Busarchitektur ist der Leitungswellenwiderstand etwas aufwendiger zu bestimmen. Die Anzahl der Stichleitungen und ihre Gesamtlänge wird durch empirische Richtformeln abgeschätzt. Der lineare Bus hat den Vorteil, dass alle Steuergeräte parallel an einer zentralen Leitung liegen. Nur wenn diese ausfällt, funktioniert der Bus nicht mehr. Diese Topologie wird häufig in Kraftfahrzeugen eingesetzt.

An jedem Leitungsende sollte sich ein Abschlusswiderstand von 120 Ohm befinden. Für einen einzelnen CAN-Bus-Teilnehmer an einer Stichleitung wirkt dies genauso wie ein einzelner 60-Ohm-Widerstand, der am Ort der Abzweigung eingefügt ist. Dieser Wert ist die zentrale Impedanz einer Sternarchitektur.

Synchronisierung und Zeitquanten

Die nominale Datenübertragungsrate im Netzwerk muss allen Teilnehmern bekannt sein, ggf. durch automatische Detektion – CAN in Automation hat dazu eine Application-Note herausgegeben, CiA 801. Die Synchronisation auf den genauen Beginn einer Nachricht erfolgt mit dem Wechsel vom rezessiven Idle-Pegel des Busses zum dominanten Synchronisations-Bit, mit dem jede Nachricht beginnt. Jeder weitere Pegelwechsel von rezessiv zu dominant kann zur dynamischen Nachsynchronisierung der Empfänger verwendet werden. Die Nachsynchronisierung gleicht Phasenrauschen und -drift zwischen den lokalen Oszillatoren aus. Eine Nachsynchronisierung findet auch während der Arbitrierungsphase statt, wenn ein Sender eine Nachricht mit höherer Priorität zu senden beginnt. Dies bewirkt

meist ebenfalls einen Phasensprung in Bezug zur vorherigen Nachricht.

Maximale Übertragungsrate und Leitungslänge

Es wird zwischen einem Highspeed-Bus mit einer Datenrate von bis zu 1 Mbit/s und einem Lowspeed-Bus mit bis zu 125 kbit/s unterschieden. Diese Raten gelten jedoch nur bei Leitungslängen bis zu **40m**. Darüber hängt die maximal zulässige Datenrate von der Leitungslänge ab. Mit niedrigeren Datenraten sind längere Leitungen möglich: Bei 500 kbit/s bis zu 100 m und bei 125 kbit/s bis zu 500 m.

Diese Maximalwerte beruhen darauf, dass die Zeit, die ein Signal am Bus anliegt (Bitzeit, Sekunde/Bit), umso kürzer ist, je höher die Übertragungsrate ist. Mit zunehmender Leitungslänge steigt jedoch die Zeit, die ein Signal braucht, bis es am anderen Ende des Busses angekommen ist (Ausbreitungsgeschwindigkeit). Zu beachten ist, dass sich das Signal nicht nur ausbreitet, sondern auch der Empfänger auch innerhalb einer begrenzten Zeit auf den Sender reagieren muss (siehe ACK). Der Sender muss wiederum die eventuelle Buspegeländerung des oder der Empfänger mitbekommen (siehe auch Arbitrierung). Deshalb ist die maximale Leitungslänge etwas komplexer zu berechnen. Es müssen Verzögerungszeiten auf der Leitung, des Transceivers (Sender und Empfänger), des Controllers (Sender und Empfänger), Oszillatortoleranzen und der gesetzte Abtastzeitpunkt (Sender und Empfänger) berücksichtigt werden.

Der weiter entwickelte CAN FD Standard erlaubt es die Datenrate nach der Verbindungsaushandlung zu erhöhen. Damit kann die Übertragungsgeschwindigkeit des Datenabschnitts um den Faktor 10 oder mehr gesteigert werden.

Als Busmedium werden nach ISO 11898-2 (High-Speed Medium Access Unit) Twisted-Pair-Kabel ursprünglich mit einem Wellenwiderstand von 108–132 Ohm empfohlen. In der derzeit gültigen Ausgabe der ISO 11898-2 aus dem Jahr 2003 ist die Toleranz mit 95–140 Ohm spezifiziert.

Die maximale Teilnehmeranzahl auf physischer Ebene hängt von den verwendeten Bustreiberbausteinen (Transceiver, physische Anschaltung an den Bus) ab. Mit gängigen Bausteinen sind 32, 64 oder bis zu 110 (mit Einschränkungen bis zu 128) Teilnehmer pro Leitung möglich (Erweiterungsmöglichkeit über Repeater oder Bridge).

Objekt-Identifier

Der Objekt-Identifier kennzeichnet den Inhalt der Nachricht, nicht das Gerät. Zum Beispiel kann in einem Messsystem den Parametern Temperatur, Spannung und Druck jeweils ein eigener Identifier zugewiesen sein. Es können mehrere Parameter unter einem Identifier vereint sein solange die Summe der Daten die maximal mögliche Länge des Datenfeldes nicht überschreitet. Die Empfänger entscheiden anhand des Identifiers, ob die Nachricht für sie relevant ist oder nicht. Zudem dient der Objekt-Identifier auch der Priorisierung der Nachrichten.

Die Spezifikation definiert zwei Identifier-Formate:

- 11-Bit-Identifier, auch „Base frame format“ genannt (CAN 2.0A)
- 29-Bit-Identifier, auch „Extended frame format“ genannt (CAN 2.0B).

Ein Teilnehmer kann Empfänger und Sender von Nachrichten mit beliebig vielen Identifiern sein, aber umgekehrt darf es zu einem Identifier immer nur maximal einen Sender geben, damit die Arbitrierung funktioniert.

Der 29-Bit-Identifier ist in erster Linie für das Umfeld von Nutzfahrzeugen, Schiffen, Schienenfahrzeugen und Landmaschinen definiert. Der CAN-Standard fordert, dass eine Implementierung das „Base frame format“ akzeptieren muss, dagegen das „Extended frame format“ akzeptieren kann, es aber zumindest tolerieren muss.

Die Liste der Objekt-Identifier einschließlich Sender und Empfänger ist Bestandteil der sog. Kommunikationsmatrix oder K-Matrix.

Arbitrierung, Priorität

Der Buszugriff wird verlustfrei mittels der bitweisen Arbitrierung auf Basis der Identifier der zu sendenden Nachrichten aufgelöst. Dazu überwacht jeder Sender den Bus, während er gerade den Identifier sendet. Senden zwei Teilnehmer gleichzeitig, so überschreibt das erste dominante Bit eines der beiden das entsprechend rezessive des anderen, was dieser erkennt und seinen Übertragungsversuch beendet. Verwenden beide Teilnehmer den gleichen Identifier, wird nicht sofort ein Error-Frame erzeugt (siehe Frame-Aufbau), sondern erst bei einer Kollision innerhalb der restlichen Bits, was durch die Arbitrierung ausgeschlossen sein sollte. Daher empfiehlt der Standard, dass ein Identifier auch nur von maximal einem Teilnehmer verwendet werden soll.

Durch dieses Verfahren ist auch eine Hierarchie der Nachrichten untereinander gegeben. Die Nachricht mit dem niedrigsten Identifier darf immer übertragen werden. Für die Übertragung von zeitkritischen Nachrichten kann also ein Identifier hoher Priorität (= niedrige ID, z. B. 0x001; 0x000 für Netzmanagement – NMT) vergeben werden, um ihnen so Vorrang bei der Übertragung zu gewähren. Dennoch kann selbst bei hochprioren Botschaften der Sendezeitpunkt zeitlich nicht genau vorher bestimmt werden, da gerade in Übertragung befindliche Nachrichten nicht unterbrochen werden können und den Startzeitpunkt einer Sendung so bis zur maximalen Nachrichtenlänge verzögern können (nichtdeterministisches Verhalten). Lediglich die maximale Sendeverzögerung für die höchstpriore Nachricht kann bei bekannter maximaler Nachrichtenlänge errechnet werden. Für niederpriore Nachrichten ist im Allgemeinen keine Aussage über den Sendezeitpunkt möglich.

Sollte ein Teilnehmer kontinuierlich Nachrichten mit einer hohen Priorität versenden, kann dies zur Blockade des Busses führen, da die Nachrichten der anderen Teilnehmer jeweils die Arbitrierung verlieren. Dieses Verhalten wird als Babbling idiot beschrieben. Sollte dieses Verhalten auf einer Fehlfunktion basieren, kann es nur durch zusätzliche Hardware – sogenannte Buswächter (Bus Guardians) – gelöst werden.

Frame-Aufbau

CAN-Daten-Frame mit elektrischen Pegeln ohne Stuffbits

CAN-Datentelegramm im Base Frame Format

CAN-Datentelegramm im Extended-Frame-Format

Die Kommunikation erfolgt mit Telegrammen. Innerhalb eines Telegramms gibt es Steuerbits und Nutzbits. Der genormte Aufbau eines solchen Telegrammrahmens wird als Frame bezeichnet.

Es gibt vier verschiedene Arten von Frames:

Daten-Frame, dient dem Transport von Daten

Remote-Frame, dient der Anforderung eines Daten-Frames von einem anderen Teilnehmer

Error-Frame, signalisiert allen Teilnehmern eine erkannte Fehlerbedingung in der Übertragung

Overload-Frame, dient als Zwangspause zwischen Daten- und Remote-Frames

Daten-Frame

Ein Daten-Frame ist logisch wie folgt aufgebaut:

Start of Frame (SOF) = ein dominantes Bit

Arbitrierungsfeld, bestehend aus einem Identifier-Segment (11 Bit oder 29+2 Bit) plus einem RTR-Bit (Remote Transmission Request, siehe unten)

Kontrollfeld (CTRL) = 6 Bit

Identifier Extension (IDE) = 1 Bit

reserved = 1 Bit

Data Length Code (DLC) = 4 Bit (Anzahl der Bytes im Datenfeld, 0 bis 8 Bytes, Werte 9 bis 15 werden nicht unterstützt)

Datenfeld (DATA) = 0 bis 8 mal 8 Bit

Prüfsummenfeld (CRC) = 15 Bit (Generatorpolynom x 15 + x 14 + x 10 + x 8 + x 7 + x 4 + x 3 + 1
$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$) gefolgt von einem rezessiven CRC-Delimiter-Bit

Bestätigungsfeld (ACK) = 2 Bit, bestehend aus einem ACK-Slot (siehe untenstehende Erläuterung) plus einem rezessiven ACK-Delimiter

End of Frame (EOF) = 7 Bit (rezessiv)

Intermission (IFS – Intermission Frame Space) = 3 Bit (= min. Anzahl der Bits, die aufeinanderfolgende Botschaften trennt)

Remote-Frame

Ein gesetztes RTR-Bit (Remote Transmission Request) kennzeichnet einen Remote-Frame (rezessiv). Mit Hilfe eines Remote-Frames kann ein Teilnehmer einen anderen auffordern, seine Daten zu senden.

Im Falle eines „Extended Identifiers“ (siehe oben) wird das RTR-Bit durch das SRR-Bit (Substitute Remote Request) ersetzt und ebenfalls rezessiv gesendet. In diesem Fall wird das nachfolgende IDE-Bit ebenfalls rezessiv gesendet, wodurch ein „Extended Identifier“ signalisiert wird. Im Anschluss werden die restlichen 18 Bit des Identifiers und anschließend das eigentliche RTR-Bit gesendet. Das IDE-Bit zählt dabei logisch zum ”Arbitrierungsfeld”, wobei das Kontrollfeld aber weiterhin aus 6 Bit besteht.

Die Datenlänge muss entsprechend der zu erwartenden Datenlänge gesetzt werden (Fehlerquelle: Viele Entwickler setzen die Datenlänge = 0 – dies ist falsch; ebenso sind CAN-Controller am Markt, welche RTR-Frames nur mit der Datenlänge 0 senden können). Der Objektidentifier ist derselbe wie der der angeforderten Nachricht.

Error-Frame

Der Error-Frame besteht aus zwei Feldern:

Das erste Feld wird bestimmt durch die Überlagerung von ERROR FLAGS, die von den verschiedenen Stationen erzeugt werden können. Das folgende Feld ist der ERROR DELIMITER (8 rezessive Bits) .

Es gibt zwei Typen von Error Flags:

Active Error Flag
6 dominante Bits, gesendet von einem Knoten, der einen Fehler im Netzwerk entdeckt hat und im Fehler-Status ”error active” ist.
Passive Error Flag
6 rezessive Bits, gesendet von einem Knoten, der einen Fehler im Netzwerk entdeckt hat und im Fehler-Status ”error passive” ist.

Overload-Frame

Der Overload-Frame ist eine Zwangspause zwischen Daten- und Remote-Frames. Er beinhaltet zwei Felder: Overload Flag und Overload Delimiter.

Es gibt zwei Arten von Überlastung, die zur Generierung des Overload-Flag führen:

Die Elektronik des Empfängers erfordert eine Verzögerung der Übertragung des nächsten Datenframes oder Remoteframes (bspw. aufgrund eines vollen Empfangspuffers).
Erkennung eines dominanten Bits auf dem Bus während einer Übertragungspause des eigenen Sendevorganges.

Ein Overload-Frame, verursacht aufgrund des ersten Falls, darf nur im ersten Bitintervall einer erwarteten Sendepause erzeugt werden, während ein Overload-Frame, bedingt durch Fall 2, einen Takt nach der Erkennung des dominanten Bits gesendet wird.

Das Overload-Flag besteht aus sechs dominanten Bits. Die allgemeine Form korrespondiert zu der des Active-Error-Flags: Die Form des Overload-Flags zerstört die festgelegte Übertragungsform, da das Bitstuffing verletzt wird. Als Konsequenz erkennen alle anderen Geräte ebenfalls die Überlastung und generieren selber wiederum auch ein Overload-Flag.

Der Overload-Delimiter besteht aus acht rezessiven Bits und entspricht der Form des Error-Delimiters.

ACK-Slot

Der Acknowledge-Slot wird verwendet, um den Empfang eines korrekten CAN-Frames zu quittieren. Jeder Empfänger, der keinen Fehler feststellen konnte, setzt einen dominanten Pegel an der Stelle des ACK-Slots und überschreibt somit den rezessiven Pegel des Senders. Im Falle einer negativen Quittung (rezessiver Pegel) muss der fehlererkennende Knoten nach dem ACK-Delimiter ein Error-Flag auflegen, damit erstens der Sender vom Übertragungsfehler in Kenntnis gesetzt wird und zweitens, um netzweite Datenkonsistenz sicherzustellen. Wird der rezessive Pegel von einem Empfänger durch einen dominanten überschrieben, kann der Absender jedoch nicht davon ausgehen, dass das Telegramm von allen anderen Empfängern erhalten wurde.

Bit Stuffing

Bitfolgen mit mehr als fünf gleichen Bits werden im CAN-Protokoll für Steuerungszwecke z. B. 'End of Frame' benutzt. Es dürfen also innerhalb des CAN-Frames nicht mehr als fünf Bits mit dem gleichen Pegel hintereinander vorkommen. Um dies zu verhindern, wird nach fünf Bits mit dem gleichen Pegel ein Bit mit dem inversen Pegel eingefügt. Dieses Bit nennt man 'Stopf-Bit' oder 'stuff bit'. Bitstopfen (bit stuffing) kann die physische Länge eines Frames vergrößern. Bit stuffing wirkt auf Start of frame (SOF) bis einschließlich Prüfsummenfeld (CRC) von Daten- sowie Remote-Frames und dient der Nachsynchronisation der Teilnehmer innerhalb eines Frames.

Datensicherung

Erkennt ein Empfänger eine Fehlerbedingung, sendet er einen Error-Frame und veranlasst so alle Teilnehmer, den Frame zu verwerfen. Sollten andere Teilnehmer diese Fehlerbedingung erkannt haben, senden sie ihrerseits direkt im Anschluss ein weiteres Error-Frame. Damit wird eine weitere Sicherheitsfunktion des CAN-Protokolls möglich. Um zu vermeiden, dass einzelne Teilnehmer durch irrtümlich erkannte Fehlerbedingungen dauerhaft den Nachrichtentransport blockieren, enthält jeder Teilnehmer Fehlerzähler. Diese Zähler erlauben nach den Regeln der Spezifikation, einen fehlerhaft arbeitenden Teilnehmer in zwei Stufen des Betriebszustands vom Bus zu trennen, wenn er wiederholt Fehler erkennt, die andere Teilnehmer nicht erkennen, oder wiederholt fehlerhafte Frames versendet. Die Zustände nennen sich error active (normal), error passive (Teilnehmer darf nur noch passive - das heißt rezessive - Error-Frames senden) und bus off (Teilnehmer darf nicht mehr senden).

Der Sender wiederholt nach dem Error-Frame seine Datenübertragung. Auch der Sender kann durch die zuvor erwähnten Fehlerzähler vom Bus getrennt werden, wenn die Datenübertragung dauerhaft fehlschlägt. Verschiedene Fehlerfälle führen zu einer unterschiedlich großen Erhöhung des Fehlerzählers.

Standards

ISO 11898-1:2015 Road vehicles - Controller area network - Part 1: Data link layer and physical signalling
ISO 11898-2:2016 Road vehicles - Controller area network - Part 2: High-speed medium access unit
11898-3:2006 Road vehicles - Controller area network - Part 3: Low-speed, fault-tolerant, medium dependent interface
ISO 11898-4:2004 Road vehicles - Controller area network - Part 4: Time-triggered communication
ISO 11898-5:2007 Road vehicles - Controller area network - Part 5: High-speed medium access unit with low-power mode
ISO 11898-6:2013 Road vehicles - Controller area network - Part 6: High-speed medium access unit with selective wake-up functionality
SAE J2284-1:2016 High Speed CAN for Vehicle Applications at 125 kbps
SAE J2284-2:2016 High Speed CAN for Vehicle Applications at 250 kbps
SAE J2284-3:2016 High Speed CAN for Vehicle Applications at 500 kbps
SAE J2284-4:2016 High Speed CAN for Vehicle Applications at 500 kbps with CAN FD Data at 2 Mbps
SAE J2284-5:2016 High Speed CAN for Vehicle Applications at 500 kbps with CAN FD Data at 5 Mbps

Weiterentwicklung

2012 wurde von Bosch ein Vorschlag zur Erhöhung der verfügbaren Bandbreite namens CAN FD (Flexible Data Rate) vorgestellt. Dies wird durch Verkürzung der Bit-Zeiten in der Datenphase und Vergrößerung des Datenfeldes auf bis zu 64 Byte erreicht. Insgesamt verspricht man sich zurzeit durch das 'improved CAN' genannte Verfahren einen bis zu 8-fach höheren Datendurchsatz. Das CAN-FD-Protokoll kann wie das Classical-CAN-Protokoll alle einfachen (single) Bitfehler erkennen. Außerdem werden mehrfache (multiple) Bitfehler mit einer noch höheren Wahrscheinlichkeit entdeckt.

CAN FD wurde international standardisiert und ist nun Bestandteil von ISO 11898-1:2015.

Anwendungsbereiche

CAN-Protokolle haben sich in verschiedenen, vor allem sicherheitsrelevanten Bereichen etabliert, bei denen es auf hohe Datensicherheit ankommt. Beispiele:

- Automobilindustrie (Vernetzung unterschiedlicher Steuergeräte, Sensoreinheiten und Multimediaeinheiten)
- Automatisierungstechnik (zeitkritische Sensoren im Feld, Überwachungstechnische Einrichtungen)
- Aufzugsanlagen (Vernetzung der Steuerung mit verschiedenen Sensoren, Aktoren und Aufzugsanlagen untereinander innerhalb einer Aufzugsgruppe)
- Medizintechnik (Magnetresonanz- und Computertomographen, Blutgewinnungsmaschinen, Laborgeräte, Elektro-Rollstühle, Herzlungen-Maschinen)
- Flugzeugtechnik (Vernetzung innerhalb von Kabinen- und Flugführungssystemen)
- Raumfahrttechnik (vermehrte Verwendung in parallelen Busarchitekturen)
- Beschallungsanlage (wird für die Steuerung von digitalen Endstufen verwendet)
- Schienenfahrzeuge
- Schiffbau (Die DGzRS lässt in die neue Generation ihrer Seenotrettungskreuzer Bus-Systeme einbauen.)
- Pyrotechnik (Vernetzung von Zündsystemen)
- Agrartechnik (Vernetzung unterschiedlicher Steuergeräte, Sensoreinheiten und Aktoreinheiten)
- Sicherheitstechnik (Vernetzung intern in Anlagen, extern bei einzelnen Bauelementen)

Höhere Protokolle

- ISO-TP
  ISO 15765-2, auch kurz ISO-TP ermöglicht den Transport von Botschaften, deren Länge die maximal 8 Bytes Nutzdaten eines CAN-Frames überschreiten. Im OSI-Modell deckt es die Schichten 3 (Network Layer) und 4 (Transport Layer) ab und kann bis zu 4095 Bytes Nutzdaten pro Telegramm transportieren. ISO-TP segmentiert längere Botschaften auf mehrere Frames und ergänzt die Datenpakete um Metadaten, die eine Interpretation der einzelnen Frames durch den Empfänger ermöglichen.
- CANopen
  CANopen ist ein auf CAN basierendes Schicht-7-Kommunikationsprotokoll, welches anfänglich in der Automatisierungstechnik verwendet wurde, mittlerweile aber vorwiegend in Embedded Systemen eingesetzt wird.

  CANopen wurde vorwiegend von deutschen klein- und mittelständischen Firmen initiiert und im Rahmen eines ESPRIT-Projektes unter Leitung von Bosch erarbeitet. Seit 1995 wird es von der CAN in Automation gepflegt und ist inzwischen als Europäische Norm EN 50325-4 standardisiert. Der Einsatz erfolgt vorwiegend in Europa, gefolgt von Asien.
- DeviceNet
  DeviceNet ist ein auf CAN basierendes Schicht-7-Kommunikationsprotokoll, welches hauptsächlich in der Automatisierungstechnik verwendet wird. DeviceNet ist vorwiegend in Amerika verbreitet. Es wurde von Allen-Bradley (gehört zu Rockwell Automation) entwickelt und später als offener Standard an die ODVA (Open DeviceNet Vendor Association) übergeben.
- J1939 sowie die Erweiterungen NMEA2000 und ISOBUS
  J1939 ist ein auf CAN basierendes Protokoll im Nutzfahrzeugbereich. Es wird von der Society of Automotive Engineers (SAE) gepflegt. Eine Einführung in J1939 findet sich in Application Note Introduction J1939

  NMEA 2000 ist eine Erweiterung von SAE J1939 für den maritimen Bereich. Das Protokoll der NMEA-Organisation breitet sich zunehmend aus. Vorgänger ist NMEA 0183. NMEA2000 ist ein IEC Standard: IEC61162-3.

  In der Landwirtschaft und Kommunaltechnik kommt der ISOBUS (ISO 11783), der eine Erweiterung des J1939 darstellt, zur Steuerung und Überwachung von Anbaugeräten zum Einsatz.
- CleANopen
  Eine Arbeitsgruppe der CAN in Automation, die CANopen Special Interest Group (SIG) 'Municipal Vehicles', entwickelt das CANopen-Anwendungsprofil für Abfallsammelfahrzeuge : CleANopen (DIN EN 50325-4).
- CANopen-Lift
  Eine 2001 gegründete Arbeitsgruppe der CAN in Automation, die CANopen Special Interest Group (SIG) 'Lift Control', entwickelt das CANopen-Anwendungsprofil (CANopen CiA-417) für Aufzüge. Die erste Version von CiA 417 wurde im Sommer 2003 veröffentlicht. Die Version 2.0 steht seit Februar 2010 auf der CiA-Webseite frei zur Verfügung. Die Arbeitsgruppe arbeitet an der Erweiterung des CANopen-Lift-Funktionsumfangs, verfeinert technische Inhalte und sorgt um die Einhaltung aktueller, gesetzlich vorgeschriebener Normen für Aufzüge in CiA-417. Die Version 2.1.0 ist im Juli 2012 und die Version 2.2.0 (verfügbar für CiA-Mitglieder) ist im Dezember 2015 als Draft Standard Proposal verabschiedet worden. Im Jahre 2016 wurde an der Version 2.3.0 (verfügbar für CiA-Mitglieder) gearbeitet.

  Jörg Hellmich (ELFIN GmbH) ist der Vorsitzende dieser Arbeitsgruppe und betreibt unabhängig vom CiA ein Wiki der CANopen-Lift-Anwendergemeinschaft mit Inhalten zu CANopen Lift.
- SafetyBUS p
  SafetyBUS p ist ein auf CAN basierendes sicheres Kommunikationsprotokoll, welches hauptsächlich in der Automatisierungstechnik zur Übertragung sicherheitsgerichteter Daten verwendet wird. Alle Busteilnehmer sind zwei-

oder sogar dreikanalig aufgebaut und prüfen die Datenintegrität. Das Übertragungsmedium selbst ist nicht sicher, die Sicherheit wird durch das SafetyBUS p-eigene Datenprotokoll erreicht. Der SafetyBUS p kann bis SIL3 eingesetzt werden.

TTCAN
Time-Triggered Communication on CAN setzt auf dem CAN-Bus auf und ermöglicht über höhere Protokollebenen eine Echtzeitsteuerung. TTCAN ist in ISO 11898-4 genormt.

CANaerospace
CANaerospace ist ein Open-Source-Kommunikationsprotokoll, welches 1998 insbesondere für den Einsatz in der Luftfahrt mit ihren besonderen Zuverlässigkeits- und Leistungsanforderungen konzipiert wurde. Im Jahr 2000 hat die amerikanische NASA CANaerospace als eigenen Standard übernommen. CANaerospace wird in zahlreichen Forschungsflugzeugen weltweit eingesetzt und hat sich als De-facto-Standard in der militärischen Flugsimulationstechnik etabliert.

ARINC 825
ARINC 825 ist ein internationaler Luftfahrt-Kommunikationsstandard, welcher in einer Technischen Arbeitsgruppe (bestehend aus mehreren Luftfahrtunternehmen, darunter Boeing und Airbus) auf der Basis von CANaerospace entwickelt wurde.

EnergyBus
EnergyBus ist ein Kommunikations- und Energieübertragungs-Bus und dazugehöriges Steckersystem für Leicht-Elektrofahrzeuge wie Pedelecs und E-Bikes. EnergyBus wird von einem eingetragenen Verein, dem EnergyBus e.V. mit Sitz in Tanna gemeinsam mit dem CAN in Automation e.V. spezifiziert. Mitglieder sind sowohl Einzelpersonen wie auch Hersteller von Steckern, Batterien, Steuerungen und Antriebseinheiten (darunter Bosch, Panasonic, Sanyo, Deutsche Bahn AG, Philips und Varta). Das Kommunikationsprotokoll ist im CANopen-Applikationsprofil 454 ënergy management systems"definiert.

FireCAN
FireCAN wurde durch Zusammenarbeit österreichischer und deutscher Feuerwehraufbauhersteller im Jahr 2006 gegründet und ist mittlerweile als Norm DIN 14700 vorhanden. Ursprünglich wurde FireCAN als freie Übereinkunft der wesentlichen am Markt befindlichen Hersteller, die redaktionelle Betreuung der gemeinsamen Spezifikation wird dabei durch die Firma Rosenbauer ausgeübt. Die Vorstellung erfolgte im Zuge der DIN-Sitzung des Ausschusses NA 031-02-02 AA 'Elektrische Betriebsmittel' am 29. Oktober 2009 in Berlin. Diese Datenbusfestlegung basiert auf einem vereinfachten CANopen-Standard und regelt sowohl die physischen Eigenschaften (Stecker, Leitungen, Anschlussbelegung), die Art und Anzahl der Teilnehmer, sowie die verwendeten Datenformate und Dateninhalte. Als wesentlicher Vorgänger ist der in der Landwirtschaft erfolgreich eingeführte ISOBUS zu verstehen.

Unified Diagnostic Services
In Personenkraftwagen sehr verbreitet ist mittlerweile Unified Diagnostic Services gemäß der ISO 14229. In älteren Modellen verwendeten viele Hersteller eigene Standards, oft basierend auf der letztlich nicht standardisierten Norm für KWP on CAN (Normentwurf ISO/-DIS 15765).

(aus https://de.wikipedia.org/wiki/Controller_Area_Network 21.Aug'20)

Microcontroller.net:

Das Controller Area Network (CAN) verbindet mehrere gleichberechtigte Komponenten(Knoten, Node) über einen 2-Draht Bus miteinander. Das CAN-Protokoll wurde 1983 von-Bosch für den Einsatz in Kraftfahrzeugen entwickelt. Aufgrund der hohen Störsicherheit, der geringen Kosten und der Echtzeitfähigkeit wird-CAN auch in der Automatisierungstechnik, vor allem in Textilmaschinen,Aufzugssteuerungen und in Landmaschinen eingesetzt. Die Organisation "Can in Automation" (CiA) widmet sich der Weiterentwicklung des CANProtokolls

Physikalische Beschreibung der CAN Schnittstelle

Die physikalischen Gegebenheiten für CAN und viele andere Busprotokolle, wie z.B.Profibus, sind in der ISO 11898 definiert. Zur Umsetzung dieser Spezifikation stehen vieleMicrochips zur Verfügung, wie z.B. der PCA82C250 von Philips.Die elektrische Störsicherheit wird unter anderem dadurch erreicht, dass ein Bit auf zweiLeitungen gleichzeitig mit einer gegensinnigen Potezialänderung abgebildet wird. Man spricht hier auch von einem differentiellen Signal. Auf einer zweiten Leitung wird also eine redundant invertierte Übertragung des logischenSignals vorgenommen. In die Leitung eingestreute Störungen wirken auf beide Leitungen in der gleichenRichtung. Da die beiden differentiellen Leitungen jedoch immer gegensinnige Pegelhaben, bleibt die Differenz der Pegel auch bei Störungen weitgehend erhalten. Dies nenntman Gleichtaktunterdrückung, auf englisch „Common Mode Rejection Ratio" (CMRR). Die Leitung CAN-High und Can-Low. enthalten das invertierte und das nicht invertierte serielleDatensignal. Durch die Ausführung als offener Collector (PNP auf VCC bei CAN-H und NPN auf GNDbei CAN-L) können ausserdem mehrere Teilnehmer auf dem Bus parallelgeschaltetwerden, ohne daß im Konfliktfall elektrische Kurzschlüsse entstehen. Der Zustand mit zwei unterschiedlichen Pegeln auf Can-H und CAN-L wird als derdominante Zustand genannt (Pegeldifferenz > 3,5 Volt); der Zustand mit zwei gleichenPegeln wird als rezessiv bezeichnet

(Pegeldifferenz < 1,5 Volt). Der dominante Zustand entspricht per CAN Definition einer logischen Null: Legt einKnoten eine logische Null auf den Bus, überschreibt er möglicherweise den Zustand einerlogischen Eins eines anderen Knotens. Die Kopplung der Knoten über die Bussleitungstellt eine logische Und-Verknüpfung dar (Wired-And).

Eine weitere Maßnahme zur Erhöhung der Störsicherheit ist das NRZI-Verfahren: Dabeiwird nach maximal fünf Bits gleicher „Polarität“ (rezessiv bzw. dominant) ein Bit der jeweilsanderen Polarität eingefügt.

Prinzip des Datenaustausches im CAN Netzwerk

Bei der Datenübertragung in einem CAN Bus werden keine Knoten adressiert, sondern der Inhalt einer Nachricht (z.B. Drehzahl oder Motortemperatur) wird durch eineneindeutigen Identifier gekennzeichnet. Neben der Inhaltskennzeichnung legt der Identifierauch die Priorität der Nachricht fest. Mit der dann folgenden Akzeptanzprüfung stellen alle Stationen nach korrektem Empfang der Nachricht anhand des Identifiers fest, ob die empfangenen Daten für sie relevant sindoder nicht. Durch die inhaltsbezogene Adressierung wird eine hohe Flexibilität erreicht: Es lassen sich sehr einfach Stationen zum bestehenden CAN-Netz hinzufügen. Außerdem ergibt sich die Möglichkeit des Multicasting: Eine Nachricht kann von mehrerenTeilnehmern gleichzeitig empfangen und ausgewertet werden. Messgrößen, die vonmehreren Steuergeräten als Information benötigt werden, können über das CAN-Netz soverteilt werden, dass nicht jedes Steuergerät einen eigenen Sensor benötigt.

Kollisionsprüfung

Jeder Teilnehmer darf Daten ohne besondere Aufforderung irgend eines Masters verschicken. Wie bei Ethernet kann es hier zu Kollisionen kommen, die allerdings perHardware aufgelöst und durch Wiederholung behoben werden. Eine Kollision wirddadurch erkannt, daß ein Sender den gesendeten Identifier selbst zurückliest undvergleicht. Bei Ungleichheit war ein Teilnehmer mit höherer Priorität da, welcher dieLeitung an irgendeiner Stelle in den dominanten Pegel gezogen hat. Der Identifier mit der niedrigsten Binärzahl hat somit die höchste Priorität. Den Vorgang zur Kollisionsprüfung über den Identifier nennt man „bitweise Arbitrierung“. Entsprechend dem "Wired-and-Mechanismus", bei dem der dominante Zustand (logisch0) den rezessiven Zustand (logisch 1) überschreibt, verlieren all diejenigen Knoten denWettstreit um die Buszuteilung, die rezessiv senden, aber auf dem Bus dominantbeobachten. Alle "Verlierer"werden automatisch zu Empfängern der Nachricht mit derhöchsten Priorität und versuchen erst dann wieder zu senden, wenn der Bus frei wird. Der CANbus ist somit ein Bussystem mit bedarfsabhängiger Buszuteilung. Auch gleichzeitige Buszugriffe mehrerer Knoten müssen immer zu einer eindeutigenBusvergabe führen. Durch das Verfahren der bitweisen Arbitrierung über die Identifier der zur Übertragung anstehenden Botschaften wird jede Kollision nach einer berechenbarenZeit eindeutig aufgelöst: Im CAN Standard Format sind es maximal 13 Bitzeiten, im erweiterten Format sind es maximal 33 Bitzeiten.

Aufbau einer CAN Nachricht

Eine Nachricht wird in einer für den CAN-Bus eigenen Form verpackt. Diese Verpackungwird als ”Frame” bezeichnet. Ein Frame besteht aus 7 Kennfeldern:

- Start-Condition
- Message Identifier
- Steuerbits
- Daten (0-8 Bytes)
- Prüfbits
- Acknowledge-Bit
- Stop-Condition

Man unterscheidet außerdem die Frames nach der Länge des Identifiers:

- Standard Frame (11 Bit Identifier)
- Extendet Frame (29 Bit Identifier)

Nach der Art des Frames unterscheidet man den

- Data Frame (Daten werden ohne spezielle Aufforderung gesendet)
- Remote Data Frame (Daten werden angefordert – Ein Empfänger, der den REMOTE identifiziert, sendet daraufhin seine Nachricht)

(aus www.mikrocontroller.net 21.Aug’20)

14.13 LVDS low voltage differential signalling

- differenzielle Spannungspegel
- relativ geringe Spannungspegel (englisch low voltage)
- die Signale werden mit einer Konstantstromquelle erzeugt

- Year created 1994
- 655 Mbit/s (up to 1-3 Gbit/s possible)
- TIA/EIA-644
- Serial ATA (SATA), PCI Express (PCIe), FireWire, HyperTransport,
- Videoschnittstellen wie DisplayPort
- Feldbusse wie SpaceWire und RapidIO
- Digital Visual Interface (DVI) oder HDMI basieren auf dem prinzipiell ähnlichen Transition-Minimized Differential Signaling (TMDS).
- Spannungshub 350 mV Unterschied
- absolute Spannung gegen Masse: 1200 mV
- Treiberseite: Konstantstromquelle 3,5 mA
- Empfängerseite: Abschlusswiderstand von 100 Ohm
- Empfänger: Spannungsänderung von +350 mV zu −350 mV
- Gegentaktstörung: bis zu 1000 mV tolerant

(frei nach https://de.wikipedia.org/wiki/Low_Voltage_Differential_Signaling 02Jan'22)

14.14 TMDS transition minimized differential signalling

Transition-Minimized Differential Signaling

- Digital Visual Interface (DVI, Spezifikation von 1999)
- High-Definition Multimedia Interface (HDMI).
- bis zu 5,94 Gbit/s
- Leitungslängen von 15 Metern bis zu 165 Megapixel pro Sekunde
- erweiterte Version mit Dual-Link (24 + x Kontakte DVI-Stecker) max. 330 Megapixel pro Sekunde
- spezielle Kanalkodierung zur Minimierung der Signalübergänge
- spezifischer 8b10b-Code,
- Bytes auf 10 Bit Sequenzen erweitern,
- langfristig Mittel gleich viele Einsen wie Nullen,
- von den bei 10 Bit möglichen 1024 Kombinationen 460 verwendet f.d. 256 möglichen des 8 Bit breiten Nettodatenwortes
- (viele der 8 Bit Nettodatenwörter zwei mögliche, gleichwertige 10-Bit Codes),
- 4 Kombinationen spezielle Steuerwörter vergleichbar VSYNC/HSYNC
- 560 Kombinationen nicht verwendet

(frei nach https://de.wikipedia.org/wiki/Transition-Minimized_Differential_Signaling 02Jan'22)

14.15 openGL

ist ein Standard (API) für Grafikkarten.

OpenGL (Open Graphics Library; deutsch Offene Grafikbibliothek) ist eine **Spezifikation** einer plattform- und programmiersprachenübergreifenden Programmierschnittstelle (API) zur Entwicklung von 2D- und 3D-Computergrafikanwendungen. Der OpenGL-Standard beschreibt etwa 250 Befehle, die die Darstellung komplexer 3D-Szenen in Echtzeit erlauben. Zudem können andere Organisationen (zumeist Hersteller von Grafikkarten) proprietäre Erweiterungen definieren. Die Implementierung des OpenGL-API erfolgt in der Regel durch Systembibliotheken, auf einigen Betriebssystemen auch als Teil der Grafikkarten-Treiber. Diese führen entsprechend Befehle der Grafikkarte aus, insbesondere müssen auf der Grafikkarte nicht vorhandene Funktionen durch die CPU emuliert werden. Der Nachfolger von OpenGL ist Vulkan.

(aus https://de.wikipedia.org/wiki/OpenGL, 09Sep'22)

OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

(aus https://wiki.freepascal.org/OpenGL_Tutorial, 09Sep'22)

```
// Include standard headers
#include <stdio.h>
#include <stdlib.h>

// Include GLEW
#include <GL/glew.h>

// Include GLFW
#include <GLFW/glfw3.h>
GLFWwindow* window;

// Include GLM
#include <glm/glm.hpp>
using namespace glm;

int main( void ){
	// Initialise GLFW
	if( !glfwInit() ){
		fprintf( stderr, "Failed to initialize GLFW\n" );
		getchar();
		return -1;
	}

	glfwWindowHint(GLFW_SAMPLES, 4);
	glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
	glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
	glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE); // To make MacOS happy; should not be needed
	glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

	// Open a window and create its OpenGL context
	window = glfwCreateWindow( 1024, 768, "Tutorial 01", NULL, NULL);
	if( window == NULL ){
		fprintf( stderr, "Failed to open GLFW window. If you have an Intel GPU,\n"
			" they are not 3.3 compatible. Try the 2.1 version of the tutorials.\n" );
		getchar();
		glfwTerminate();
		return -1;
	}
	glfwMakeContextCurrent(window);

	// Initialize GLEW
	if (glewInit() != GLEW_OK) {
		fprintf(stderr, "Failed to initialize GLEW\n");
		getchar();
		glfwTerminate();
		return -1;
	}

	// Ensure we can capture the escape key being pressed below
	glfwSetInputMode(window, GLFW_STICKY_KEYS, GL_TRUE);

	// Dark blue background
	glClearColor(0.0f, 0.0f, 0.4f, 0.0f);

	do{
```

```
                // Clear the screen. It's not mentioned before Tutorial 02, but it can cause flickering,
                // so it's there nonetheless.
                glClear( GL_COLOR_BUFFER_BIT );

                // Draw nothing, see you in tutorial 2 !


                // Swap buffers
                glfwSwapBuffers(window);
                glfwPollEvents();

        } // Check if the ESC key was pressed or the window was closed
        while( glfwGetKey(window, GLFW_KEY_ESCAPE ) != GLFW_PRESS &&
                   glfwWindowShouldClose(window) == 0 );

        // Close OpenGL window and terminate GLFW
        glfwTerminate();

        return 0;
}
```

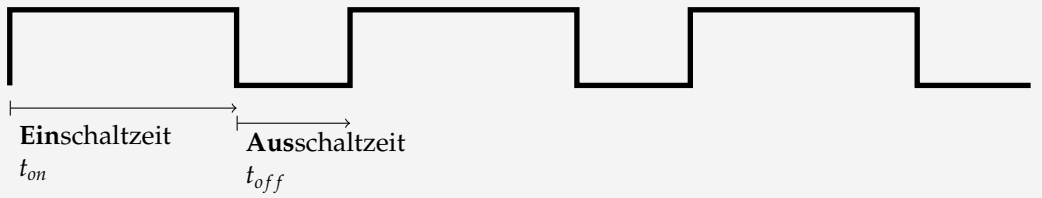
(aus http://www.opengl-tutorial.org/download/, 09Sep'22)

14.16 PWM basics

Wenn ein elektronisches Schaltelement (MosFET, BJT, Thyristor, TRIAC, GTO, Röhrentriode, Röhrenpentode, Relais usw.)

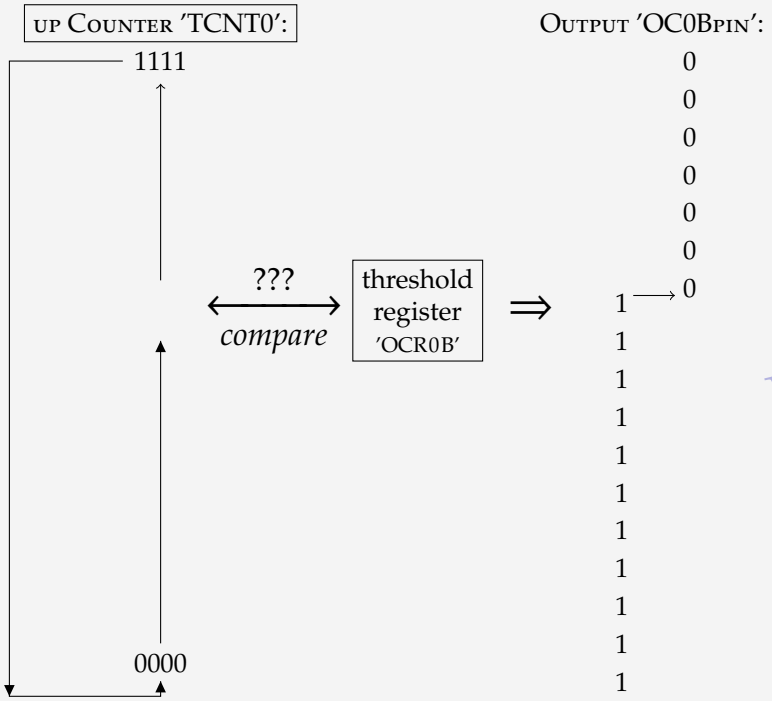
- **ganz AUS** geschaltet ist,
  fällt daran keine Verlustleistung $P_D = I_D * U_D$ ab,
  weil dann die Stromstärke $I_D = 0$ ist. } $P_D = 0$
- **ganz EIN** geschaltet ist,
  fällt daran keine Verlustleistung $P_D = I_D * U_D$ ab,
  weil dann die Spannung am Schaltelement $U_D = 0$ ist. } $P_D = 0$
- der Last (Motor, Leuchte, Trafo, Lautsprecher udgl.)
  wird umso mehr Leistung zugeführt,
  je länger der Strom eingeschaltet ist.

PWM 'Pulse Width Modulation' (eingedeutscht auch 'PulsWeitenModulation')
bedeutet die
Variation der Puls-einschalt-Dauer
eines zyklisch wiederholenden **ganz EIN** und **ganz AUS** Schaltens.



wenig:

viel:



* Counter (TCNT) kann auch abwärts zählend realisiert werden

14.16.1 8-Bit-PWM-Unit in VHDL

```
/* TN13-PWM-Unit, VHDL-Pseudocode:
   Ein Aufwaertszaehler (TCNT0reg) beinnt bei '00000000',
   schaltet den Output (OC0Bpin) Hi, wird CLOCK-getaktet,
   zaehlt bis '11111111' und beginnt wieder bei '00000000'.
   Beim Zaehlerstand eines Vergleichswertes (OCR0Breg)
   schaltet der Output (OC0Bpin) auf Lo
  -->
   je hoeher der Vergleichswert, desto laenger die Hi-Phase */

Entity PwmUnit is port(
        OC0Bpin: out std_logic;
        clock:   in  std_logic;
);

Architecture ToDo of PwmUnit is BEGIN
        TCNT0reg,
        OCR0Breg: signal  std_logic_vector(7 downto 0);
        TN13pwmUnit: process(clock) BEGIN
          if(rising_edge(clock) then
            TCNT0reg <= TCNT0reg+1;     --raufzaehlen
          end if;
          if (falling_edge(clock) then
            if(TCNT0reg=OCR0Breg) then  --abschalten?
                OC0Bpin <= '0';
            elsif (TCNT0reg=='1111 1111') then  --Zaehlbeginn?
                OC0Bpin <= '1';                 --einschalten
                TCNT0 <= '0000 0000';           --Counter:=max
            end if;
          endif
        END TN13pwmUnit;
END Architecture ToDo;
```

VHDL

14.16.2 uC-Programmierung der 'PWM-Unit'

```
;AT-TN13:
APPLICATION_INIT:
        LDI     R_PwmWert, C_Init_Prozent
        OUT     OCR0B, R_PwmWert

        LDI     Rtmp1,  0b00100011
        ;                 !!!!  ++--- WGM01+00: WaveGenMode: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
        ;                 !!++------- Com0B1, Com0B0: 0=off, 3=clear/set, 2=set/clear OC0B(PB.1)
        ;                 ++--------- Com0A1, Com0A0: 0=off, 2=clear/set, 3=set/clear OC0A(PB.0)
        OUT     TCCR0A, Rtmp1

        LDI     Rtmp1,  0b00001001
        ;                 !/  !+++--- CS02..CS00: 0=no 1=clk/1 2=/8 3=/64 4=/256 5=/1024 6+7=extClk
        ;                 !   +------ WGM02: WaveGenMode2: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
        ;                 +---------- FOC0A,FOC0B: ForceOutputCompare
        OUT     TCCR0B, Rtmp1
L_End_INIT:

        .
        .
        .

L_SetPwm:
        OUT     OCR0B, R_PwmWert

        ...
```

Assembler

14.17 Handhabung von Power-HexMosFET

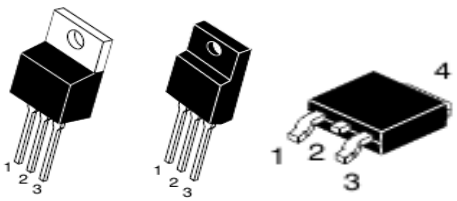
Unsere Impuls-Leistungs-MosFET (*'PwrMosFET'*) sind sog. selbstsperrende Anreicherungs- (enhancemend-) DMOS-HexFET (Weiterentwicklung aus dem VMOS); das *'Hex'* ist von der hexagonalen Sechseck-Wabenstruktur.

Die PwrMosFETs halten erstaunliches aus; zwei Misshandlungen **töten** sie jedoch rasch und dauerhaft:

a) Überschreiten der $U_{GS,max}$ (meist $\pm 20V$)
(auch **nur 1 [ns]** !)
Elektronen oder Metall-Ionen durchschlagen das Gate-Oxid und bleiben z.T. dort 'hängen', sodass diese Isolationsschicht leitend wird — das war's dann.
(besonders gefährlich bei
(selbst-)Induktionsspannungsspitzen)

b) Überhitzung == Betrieb ohne Kühlkörper (heatsink)
Die Atombewegungen werden so heftig, dass die Dotierungs-Atome ihren Platz verlassen (dem E-Feld entsprechend). Dies baut die Dotierungs-Grenzschichten ab/um zu einem falsch dotierten Bauteil, das dann immer schlechtere Verstärkungen und Schaltzeiten aufweist und schließlich dauerleitend wird.
Weiters wandert (*'vergiftet'*) Aluminium aus den Kontaktierungen ins Halbleitermaterial und überbrückt dort die Isolierungen; dies führt zu dauerleitenden Leistungshalbleitern.
Extrem überlasteten Bauteilen können aber auch interne Zuleitungs-Bond-Drähte ab-/durchschmelzen, sodass sie einfach nicht mehr kontaktiert sind (dann sperren sie natürlich).

Die TO-220/TO0247/DPAK Gehäuse (case) führen ohne Kühlkörper kaum Wärme ab und werden schon bei nur 1W Leistung unzulässig warm; bei 3W leben sie nur **wenige Sekunden** (experimentell ermittelt). Schon kleine U- oder Fingerkühlkörper verbessern das erheblich.
Wir messen deshalb, wie im Leistungsbereich üblich, *'gepulst'*, d.h. mit Rechtecksignalen *kurzer* Einschaltzeit. Mit *'kurz'* meinen wir solche Einschaltzeiten, die die spezifizierten, maximalen *'single pulse avalanche energy'* (zB. $E_{AS} = 91[mJ]$) und *'repetitive avalanche energy'* (zB. $E_{AR} = 4.8[mJ]$) -Angaben des Bauteils ausreichend unterschreiten.
(*Energie* $W[Joule] = P[Watt] \times t[Sekunden]$) Die verbreiteten Impuls-Schalt-Leistungs-MOSFET in den Gehäusen TO-220, TO-220FP/ISO, TO-247, DPAK, D2PAK ua:



Das heisst auch für DICH:

a) ausschließlich nur **mit** Kühlkörper! ← nur! ← **niemals!!!** ohne

b) **DU** sorgst dafür, dass die V_{GS} die max.±20V **niemals** überschreiten **kann!**

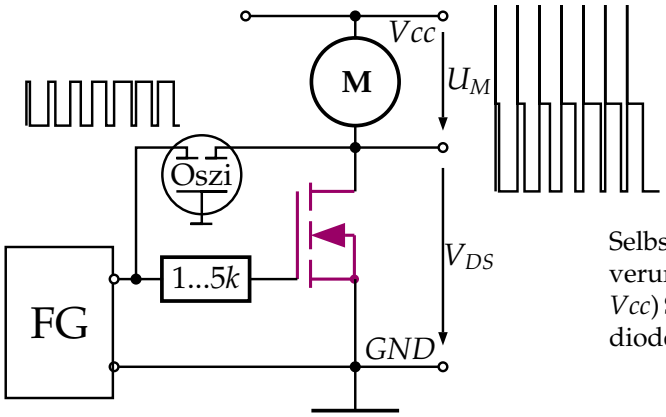
Die Anschlüsse sind (i.d.R.)

1-Gate
2-Drain (mit Kühllasche (4) verbunden)
3-Source

Die neueren SMD Varianten werden oft mit der Kühllasche (4) flach auf die Platine (PCB) gelötet, sodass die Kupferfläche ('pad') zugleich Drain-Anschluss und Kühlkörper (heat sink) darstellt und das Anschlussbein (2) fehlt.
Typisch sind:

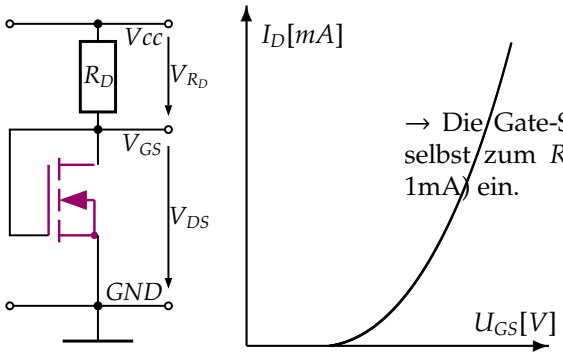
- $P_{V,max} \approx 50W$
- $V_{BR,DSS} \approx 50..800V$
- $I_{D,max} \approx 2..30A$

14.17.1 Primitiv-Schaltung:



Selbst**induktion** ('inductive kickback') verursacht die bremsenden ($U_{spitz} > Vcc$) Spannungsspitzen → mit Freilaufdiode läuft der Motor besser.

14.17.2 $V_{GS,th}$ - Messung:

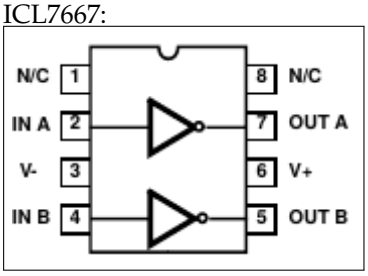


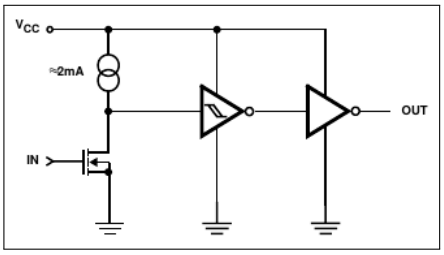
→ Die Gate-Spannung V_{GS} stellt sich selbst zum R_D-Strom passend (meist 1mA) ein.

14.17.3 Gate-Treiber für Power MosFET

- Push/Pull BJT od. MosFET
- Gatterparallelschaltung (zB. alle 6 Inverter eines 4049 od. 4050 parallel)
- spez. ICs zB. TC4427 (< 1.5A), ICL7667 (< 300mA) ua.

ICL7667:



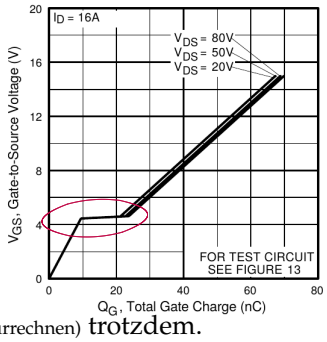


→ Wieviel Strom brauchi (um ein MosFET-Gate *schnell* aufzuladen) ?

Regel-1: **Kuh = Kuh:** **Q = C*U**: [As] = [F] * [V] ⟶ I = C*U/t [s] = [F]*[V]/[s]

Regel-2: **Miller-Effekt:** $C_{in} = C_{GS} + A_V * C_{GD}$ (A_V...Amplification of Voltage)

V(C_{GD}) muss sich nicht um ΔV_{in} erhöhen sondern
- weil gleichzeitig V_D sinkt -
um *viel* mehr (im kontinuierlichen (=analogen) Fall um die Spannungsverstärkung Av.



Im Schaltbetrieb ($A_V = \infty$) rechnet man **einfacher** mit den Ladungen:
Ein Datenblatt (zB. IRF540) zeigt das ausschlaggebende 'Plateau'
bei $V_{GS,th}$ mit dessen Miller-Effekt.
Die Q_G-Beiträge unterhalb und oberhalb
sind relativ bedeutungslos.
Der Datenblattwert 'Q_G'=Q_{GS} + Q_{GD} gilt für
ΔV_G: $0V \rightarrow 10V$
Obwohl eher der Bereich $3V \rightarrow 6V$ anzupeilen wäre,
verwenden wir den Datenblattwert einfachheitshalber (kannma ja korrekturrechnen) trotzdem.
Dann wird es einfach:

Mit Q = I*t ergibt sich | I = 72nCb / t |

Den IRF540 in 100ns einzuschalten erforderte I = 72nCb/100ns =

| 0.72A !!! |

Umgekehrt dauert das IRF540-Einschalten mit I_G=300mA (ICL7667) t = 72nCb/0.3A =

| 240 ns |

Ein BC560 ($I_{C,max} = 100mA$ braucht t = 72nCb/0.1A =

| 720 ns |

14.18 "Usb Statt Uart"

USB siehe obelix- Link 'Can/Eth/Usb' (w12aEsCom*.pdf)
USB Electrical specification:

Die USB-Datenleitungen D+ und D- pendeln zwischen $\leq 0.3V$ und $\geq 2.8 - 3.0V$ sodass sich die Differenzen D^+ - D^-- = +3V bzw. D^- - D^+ = -3V ergeben. Der verwendete NRZI-Code (Non-Return-to-Zero-Inverted) ordnet einem Pegelwechsel eine '1' zu. Ohne Wechsel entsteht eine '0'

USB signals are transmitted using differential signaling on a twisted-pair data cable with 90Ω ± 15% characteristic impedance. Low speed (LS) and Full speed (FS) modes use a single data pair, labelled D+ and D-, in half-duplex. Transmitted signal levels are 0.0-0.3 V for logical low, and 2.8-3.6 V for logical high level. The signal lines are not terminated. High speed (HS) mode uses the same wire pair, but with different electrical conventions. Lower signal voltages of -10 to +10 mV for low and 360 to 440 mV for logical high level, and termination of 45 Ω to ground or 90 Ω differential to match the data cable impedance. SuperSpeed (SS) adds two additional pairs of shielded twisted wire (and new, mostly compatible expanded connectors). These are dedicated to full-duplex SuperSpeed operation. The half-duplex lines are still used for configuration. SuperSpeed+ (SS+) uses increased data rate (Gen 2×1 mode) and/or the additional lane in the Type-C connector (Gen 1×2 and Gen 2×2 mode).
A USB connection is always between a host or hub at the A connector end, and a device or hub's upstream port at the other end.

Signaling state
The host includes 15 kΩ pull-down resistors on each data line. When no device is connected, this pulls both data lines low into the so-called single-ended zero state (SE0 in the USB documentation), and indicates a reset or disconnected connection.

(aus →https://en.wikipedia.org/wiki/USB_(Communications))

The high-speed current driver is used for high-speed data transmission. A current source derived from a positive supply is switched into either the D+ or D- lines to signal a J or a K, respectively. The nominal value of the current source is 17.78 mA. When this current is applied to a data line with a 45Ω termination to ground at each end, the nominal high level voltage (VHSOH) is +400 mV. The nominal differential high-speed voltage (D+ - D-) is thus 400 mV for a J and -400 mV for a K. The current source must comply with the Transmit Eye Pattern Templates specified in Section 7.1.2.2, starting with the first symbol of a packet. One means of achieving this is to leave the current source on continuously when a transceiver is operating in high-speed mode. If this approach is used, the current can be directed to the port ground when the transceiver is not transmitting (the example design in Figure 7-1 shows a control line called HS_Current_Source_Enable to turn the current on, and another called HS_Drive_Enable to direct the current into the data lines.) The penalty of this approach is the 17.78 mA of standing current for every such enabled transceiver in the system.
The preferred design is to fully turn the current source off when the transceiver is not transmitting.

(aus www.usb.org usb_20.pdf)

- jedes USB-Device (Gerät) hat ≥ 1 'Endpoints'
- Endpoits sind entweder nur 'in' oder nur 'out'
- nur der control-Endpoint (#0) ist zugleich 'in' und 'out'
- Geräte-Adressen werden erst nach dem Einstecken vergeben (*enumeration*)
  dh. im ersten Moment wird Adresse '0' verwendet
- der root-hub verwaltet mehrere getrennte Busse mit getrennten Nummernkreisen (dh. Device#0 existiert mehrfach)
- die Initiative zu einer Datenübertragung geht **immer** vom Host (root-Hub) aus, niemals vom Device

  Hosts müssen die Devices im *Polling* abfragen

Data Flow Types (Transfer Modes):

- **Control-Transfer:**
  Adressvergabe *enumeration*, Steuerkommandos, Statusabfragen, Device- und Interface-Descriptoren
  Control data is used by the USB System Software to configure devices when they are first attached. Other driver software can choose to use control transfers in implementation-specific ways. Data delivery is lossless. (USB-IF Implementers Forum, www.usb.org)
- **Isochronous Transfer:**
  aus dem Griechischen: 'iso'≡gleich/gleichartig/gleichmächtig chronos≡Zeit/Uhr/Gott-der-Zeit(Kronos)
  regelmäßig getaktete Transfers
  alle 1ms (Low und Full Speed) bzw. alle 125us (Hi-Speed und SuperSpeed), zB. Soundkarten,

Mikrofone, Kamera-Audio *Isochronous data is continuous and real-time in creation, delivery, and consumption. Timing-related information is implied by the steady rate at which isochronous data is received and transferred. Isochronous data must be delivered at the rate received to maintain its timing. In addition to delivery rate, isochronous data may also be sensitive to delivery delays. For isochronous pipes, the bandwidth required is typically based upon the sampling characteristics of the associated function. The latency required is related to the buffering available at each endpoint. A typical example of isochronous data is voice. If the delivery rate of these data streams is not maintained, drop-outs in the data stream will occur due to buffer or frame underruns or overruns. Even if data is delivered at the appropriate rate by USB hardware, delivery delays introduced by software may degrade applications requiring real-time turn-around, such as telephony-based audio conferencing. The timely delivery of isochronous data is ensured at the expense of potential transient losses in the data stream. In other words, any error in electrical transmission is not corrected by hardware mechanisms such as retries. In practice, the core bit error rate of the USB is expected to be small enough not to be an issue. USB isochronous data streams are allocated a dedicated portion of USB bandwidth to ensure that data can be delivered at the desired rate. The USB is also designed for minimal delay of isochronous data transfers. (USB-IF Implementers Forum, www.usb.org)*

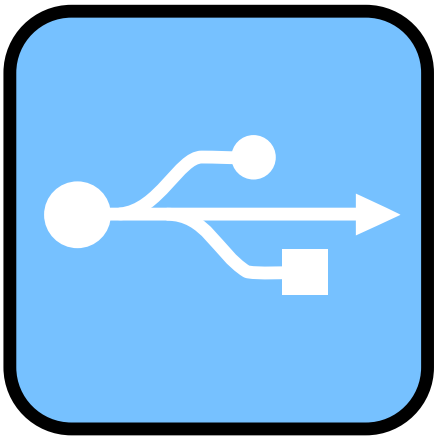
- **Interrupt-Transfer:**
  von lateinisch: inter≡zwischen, rumpere≡brechen/reissen/zerteilen
  auf Bedarf des Device (Abfrage trotzdem mittels Polling) *A limited-latency transfer to or from a device is referred to as interrupt data. Such data may be presented for transfer by a device at any time and is delivered by the USB at a rate no slower than is specified by the device. Interrupt data typically consists of event notification, characters, or coordinates that are organized as one or more bytes. An example of interrupt data is the coordinates from a pointing device. Although an explicit timing rate is not required, interactive data may have response time bounds that the USB must support. (USB-IF Implementers Forum, www.usb.org)*
- **Bulk-Transfer:**
  große Datenmengen, Datenträger, low priority, bei freier Kapazität *Bulk data typically consists of larger amounts of data, such as that used for printers or scanners. Bulk data is sequential. Reliable exchange of data is ensured at the hardware level by using error detection in hardware and invoking a limited number of retries in hardware. Also, the bandwidth taken up by bulk data can vary, depending on other bus activities. (USB-IF Implementers Forum, www.usb.org)*

'Device Class' definiert ua.:

HID - Human Interface Device, KBD, Maus, Joystick . . .
modem
security devices
Kartenleser/schreiber
audio + video device
imaging device
printer
compliance test device
USB hub
. . .
vendor defined

Universal Serial Bus

Der **Universal Serial Bus** (**USB**) [ˌjuːnɪˈvɜːsl ˈsɪɹɪəl bʌs] ist ein serielles Bussystem zur Verbindung eines Computers mit externen Geräten. Mit USB ausgestattete Geräte oder Speichermedien (USB-Speichersticks) können im laufenden Betrieb miteinander verbunden (*Hot Swapping*) und angeschlossene Geräte sowie deren Eigenschaften automatisch erkannt werden. Vor der Einführung von USB gab es eine Vielzahl verschiedener Schnittstellentypen mit unterschiedlichsten Steckern zum Anschluss von Zubehör und Peripheriegeräten an Heim- und Personal Computer. Fast alle diese Schnittstellenvarianten wurden durch USB-Stecker ersetzt, was für die Anwender Vereinfachungen mit sich brachte, die jedoch durch die Vielzahl an USB-Steckern und -Buchsen relativiert wurde. Ursprünglich 1996 eingeführt, wurde im Jahr 2000 die heute noch meistverbreitete Version USB 2.0 spezifiziert, mit der deutlich höhere Datenübertragungsraten möglich wurden. Die aktuelle, 2008 definierte Version 3.0 bietet einen neuen „SuperSpeed"-Übertragungsmodus mit einer Datentransferrate von 5 Gbit/s.



USB-Symbol

1 Überblick

USB ist ein serieller Bus, d. h. die einzelnen Bits eines Datenpaketes werden nacheinander übertragen. Die Datenübertragung erfolgt symmetrisch über zwei verdrillte Leitungen, wobei durch die eine Leitung das Datensignal und durch die andere das dazu jeweils invertierte Signal übertragen wird. Der Signalempfänger bildet die Differenzspannung beider Signale; der Spannungsunterschied zwischen 1- und 0-Pegeln ist dadurch doppelt so groß, eingestrahlte Störungen werden weitgehend eliminiert. Das erhöht die Übertragungssicherheit und unterdrückt Gleichtaktstörungen. Zwei weitere Leitungen dienen zur Stromversorgung der angeschlossenen Geräte. Durch die Verwendung von nur vier Adern in einem Kabel kann dieses dünner und billiger ausgeführt werden als bei parallelen Schnittstellen. Eine hohe Datenübertragungsrate ist mit relativ geringem Aufwand zu erreichen, da nicht mehrere Signale mit identischem elektrischen und zeitlichen Verhalten übertragen werden müssen.

Altes USB-Logo, das nicht mehr verwendet werden soll

Die Bus-Spezifikation sieht einen zentralen Host-Controller *(Master)* vor, der die Koordination der angeschlossenen Peripherie-Geräte (den sog. Slave-Clients) übernimmt. Daran können theoretisch bis zu 127 verschiedene Geräte angeschlossen werden. An einem USB-Port kann immer nur ein USB-Gerät angeschlossen werden. Sollen an einem Host mehrere Geräte angeschlossen werden, muss deshalb ein Verteiler (Hub) für deren Kopplung sorgen. Durch den Einsatz von Hubs entstehen Baumstrukturen, die alle im Host-Controller enden.

2 Einsatzgebiete von USB

USB eignet sich für viele Geräte wie Massenspeicher (etwa Festplatte, Diskette, DVD-Laufwerk), Drucker, Scanner, Webcams, Maus, Tastatur, aktive Lautsprecher, aber auch Dongles und sogar Grafikkarten und Monitore.[1] Einige Geräte, zum Beispiel USB-Speichersticks, sind überhaupt erst mit USB entstanden. USB kann für Geräte mit geringem Stromverbrauch wie Mäuse, Telefone, Tastaturen, aber auch einige

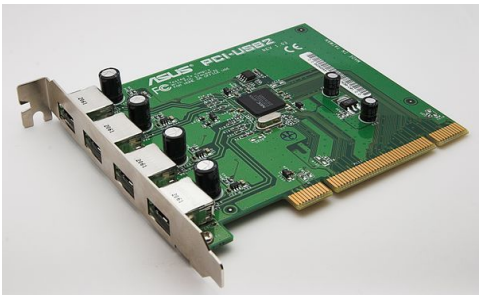
CIS-Scanner oder manche 2,5-Zoll-Festplatten und externe Soundkarten die Stromversorgung übernehmen.

Über USB können heute zahlreiche Gerätetypen angeschlossen werden, die vor der USB-Einführung über eine größere Zahl verschiedener Schnittstellentypen angebunden wurden. Zu den abgelösten älteren Typen gehören sowohl serielle (RS-232, PS/2-Schnittstelle für Tastatur und Maus, Apple Desktop Bus), parallele (Centronics-Schnittstelle) als auch analoge (Gameport) Schnittstellen. Die alten Schnittstellen werden auf manchen Rechner-Hauptplatinen und Notebooks teils noch immer angeboten, auch wenn entsprechende Geräte nicht mehr im Handel erhältlich sind. Alte Geräte mit solchen Anschlüssen, wie serielle 56k-Modems oder parallele Drucker sind aber vielerorts noch vorhanden. Im industriellen Bereich wird noch oft RS-232 über ältere PCs oder Adapterkarten eingesetzt, da entsprechende USB-Adapter nicht echtzeitfähig sind und Peripheriegeräte in diesem Umfeld wesentlich langlebiger sind. Mittlerweile hat USB auch externe SCSI-Schnittstellen weitgehend verdrängt.

Im Vergleich zu den früheren Lösungen bietet USB deutlich höhere Datenübertragungsraten. Die Daten werden jedoch in Paketen übertragen, für manche zeitkritische Anwendungen ist es deshalb weniger geeignet – etwa bei mit nur wenigen Bytes belegten Paketen, die die Übertragungsrate senken, oder wenn das Sammeln von Bytes zum Füllen eines Pakets die Übertragung verzögern würde.

Bereits seit der Einführung der USB-2.0-Spezifikation sind relativ hohe Datenübertragungsraten möglich, dadurch wurde USB für den Anschluss weiterer Gerätearten wie Festplatten, TV-Schnittstellen und Fotokameras geeignet. Bei externen Massenspeicherlösungen steht USB heute in Konkurrenz zu FireWire und eSATA und hat diese zumindest im Heimbereich fast vollständig verdrängt.

3 Geschichte und Entwicklung



USB-2.0-PCI-Erweiterungskarte

Der universelle serielle Bus *(USB 1.0)* wurde vom Hersteller Intel entwickelt und 1996 im Markt eingeführt. Einen wichtigen Beitrag hierzu leistete das Entwicklungsteam um Ajay Bhatt bei Intel. Er war zum Anschluss von Peripheriegeräten an PCs konzipiert und sollte die Nachfolge einer ganzen Reihe damals verwendeter PC-Schnittstellen antreten und diese vereinheitlichen. Deshalb war die USB-Spezifikation nicht auf Tastatur und Maus begrenzt, sondern schloss auch andere Peripheriegeräte wie Drucker und Scanner mit ein. Massenspeicher – wie etwa Festplatten – wurden zwar von USB 1.0 unterstützt, wegen der maximalen Datenrate von 12 Mbit/s waren sie dafür aber nur sehr eingeschränkt zu gebrauchen.

Als einer der ersten Chipsätze unterstützte 1996 der ursprünglich für den Pentium Pro entwickelte und später für den Pentium II verwendete *440FX* das USB-Protokoll, was vor Einführung der ATX-Mainboards jedoch kaum bis gar nicht beworben wurde. Die Hauptursache dafür dürfte zum einen in der mangelhaften beziehungsweise fehlenden Unterstützung von USB durch die damals verbreiteten Betriebssysteme Windows 95 und Windows NT 4.0 gelegen haben, zum anderen waren in der Anfangszeit auch kaum USB-Geräte verfügbar.

Ende 1998 folgte die überarbeitete Spezifikation *USB 1.1*, die in erster Linie Fehler und Unklarheiten in der 1.0-Spezifikation behob und den *Interrupt Out Transfer* hinzufügte. Die Geschwindigkeit erhöhte sich nicht. USB 1.x war deshalb keine Konkurrenz zu Apples FireWire-Standard *(IEEE 1394)*, der von Anfang an (1995) eine Datenrate von bis zu 400 Mbit/s hatte und im April 2003 auf bis zu 800 Mbit/s beschleunigt wurde. Dennoch setzte Apple die Schnittstelle in der Revision USB 1.1 mit der Entwicklung des iMac ein. Mit diesem beginnend ersetzte Apple damit den hauseigenen ADB.

Im Jahr 2000 wurde *USB 2.0* spezifiziert, was vor allem eine weitere Datenrate von 480 Mbit/s hinzufügte und so den Anschluss von Festplatten oder Videogeräten ermöglichte. Produkte dafür erschienen jedoch erst ab 2002 am Markt.

2008 wurden die neuen Spezifikationen für *USB 3.0 SuperSpeed* vorgestellt, die mit einer Datenrate von 5 Gbit/s beworben wird, allerdings nur eine Brutto-Datentransferrate von 4 Gbit/s erlaubt. Die theoretisch maximal mögliche Netto-Datenrate liegt noch einmal etwas unter der Brutto-Datenrate. Mit dieser Spezifikation werden auch neue Stecker, Kabel und Buchsen eingeführt, die größtenteils mit den alten kompatibel sind.[2]

Die ersten Mainboards und Geräte mit USB 3.0 zogen 2011 in den Massenmarkt ein. Im selben Jahr veröffentlichte Intel zusammen mit Apple die Thunderbolt-Schnittstelle in direkter Konkurrenz. Thunderbolt ist dabei doppelt so schnell wie USB 3.0, Thunderbolt 2 sogar viermal so schnell (2× 10 Gbit/s) und bietet darüber hinaus ein mitgeschleiftes DisplayPort-Signal.

4 Spannungsversorgung

Neben dem Datenprotokoll spezifiziert der USB-Standard die bereitgestellte Spannung. Sie ist stabilisiert, liegt bei 5 V ±5 % und liefert eine Stromstärke von mindestens 100 mA. Auf diesem Standard basieren USB-Netzteile.

Erst nach Freigabe durch den Host-Controller darf ein Gerät mehr als die obigen 100 mA, aber nicht mehr als 500 mA (bis USB 2.0) bzw. 900 mA (USB 3.0[3]) Strom beziehen. Am Ausgang des USB-Host muss die Spannung zwischen 4,65 V und 5,25 V liegen, allerdings ist ein Spannungsabfall bis auf 4,40 V am Ende eines USB-Kabels zulässig, hinter einem passiven USB-Hub sind sogar 4,00 V erlaubt.[4]

Externe 2,5″-Festplatten haben Anlaufströme von 600 mA bis 1100 mA, im Betrieb begnügen sie sich mit 250 mA bis 400 mA (Stand: 2010). Die kurzzeitige Überlastung des USB-Ports wird von fast allen Geräten geduldet, nur wenige Geräte haben mit besonders stromhungrigen Festplatten Probleme. Die früher häufig zu findenden Doppel-USB-Anschlüsse (die laut USB-Spezifikation nicht zulässig sind) oder zusätzliche Betriebsspannungseingänge an 2,5"-Festplatten sind selten geworden (Stand: 2011). Externe 1,8″-Festplatten liegen mit Anlaufströmen um die 400 mA und Betriebsströmen um die 150 mA innerhalb der USB-Spezifikation und bereiten somit keine Probleme. Mit USB 3.0 wurde auch der maximale Strom auf 900 mA erhöht.[3] Damit ist die Stromversorgung vieler, aber nicht aller, im Handel erhältlicher externer 2,5″-Festplatten unter Einhaltung der USB-Spezifikationen gesichert. Im Gegensatz zu kleineren Festplattenformaten lassen sich externe 3,5″-Festplatten grundsätzlich nicht ohne separate Spannungsversorgung an einem USB-Anschluss betreiben. Zum einen, weil sie neben den 5 V auch 12 V als Betriebsspannung benötigen, und zum anderen, weil ihr Strombedarf über die spezifizierten 500 mA hinausgeht. Typisch liegt dieser bei 800 bis über 1000 mA.

In der EU-Initiative für einheitliche Mobiltelefon-Lade/Netzgeräte,[5][6] welche sich im Wesentlichen an die 2009 in Version 1.1 erschienene USB „Battery Charging Specification“[7] anlehnt, ist auch ein USB-Lademodus mit einem Ladestrom zwischen 500 und 1500 mA spezifiziert; dieser Lademodus wird mittels Kennung in der Datenleitung aktiviert.

5 Übertragungstechnik/Spezifikation

5.1 Host-Controller

Die Kommunikation bei USB wird von einem Host-Controller gesteuert, der heute in der Regel auf dem Motherboard eines Computers verbaut ist. Nur dieser kann Daten von einem Gerät lesen oder zu einem Gerät senden (Ausnahme: siehe USB On-the-Go). Ein Gerät darf nur dann Daten zum Host-Controller senden, wenn es von diesem abgefragt wird. Bei zeitkritischen Datenströmen, wie etwa bei Mausbewegungen, muss der Hostcontroller von sich aus häufig genug beim Gerät anfragen (Polling), ob es Daten senden will, um ein Ruckeln zu verhindern.

Die USB-Controller-Chips in den PCs halten sich an einen von vier etablierten Standards. Diese unterscheiden sich in ihrer Leistungsfähigkeit und der Implementierung von bestimmten Funktionen. Für ein USB-Gerät sind die verwendeten Controller (fast) vollständig transparent, allerdings ist es für den Benutzer des PC mitunter wichtig, feststellen zu können, welche Art Chip der Rechner verwendet, um den korrekten Treiber auswählen zu können.

Universal Host Controller Interface UHCI wurde im November 1995 von Intel spezifiziert. Die aktuelle Version des Dokuments trägt die Revisionsnummer 1.1. UHCI-Chips bieten Unterstützung für USB-Geräte mit 1,5 oder 12 Mbit/s Datenrate im *Low-* oder *Full-Speed*-Modus. Sie werden ausschließlich von den Herstellern Intel und VIA Technologies gebaut.

Open Host Controller Interface OHCI ist eine Spezifikation, die gemeinsam von Compaq, Microsoft und National Semiconductor entwickelt wurde. Version 1.0 des Standards wurde im Dezember 1995 veröffentlicht, die aktuelle Fassung trägt die Versionsnummer 1.0a und stammt von September 1999. Ein OHCI-Controller hat prinzipiell die gleichen Fähigkeiten wie seine UHCI-Pendants, erledigt aber mehr Aufgaben in Hardware und ist dadurch marginal schneller als ein UHCI-Controller. Dieser Unterschied bewegt sich meist in Bereichen, die gerade noch messbar sind, daher kann man ihn in der Praxis vernachlässigen; Geräteentwickler müssen es jedoch berücksichtigen. Bei USB-Controllern auf Hauptplatinen mit Chipsätzen, die nicht von Intel oder VIA stammen, und auf USB-PCI-Steckkarten mit Nicht-VIA-Chipsätzen handelt es sich mit hoher Wahrscheinlichkeit um OHCI-Controller.

Enhanced Host Controller Interface EHCI stellt USB-2.0-Funktionen bereit. Es wickelt dabei nur die Übertragungen im *Hi-Speed*-Modus (480 Mbit/s) ab. Wenn man USB-1.1-Geräte an einen Port mit EHCI-Chip steckt, reicht der EHCI-Controller den Datenverkehr an einen hinter ihm liegenden UHCI- oder OHCI-Controller weiter (alle Controller sind typischerweise auf demselben Chip). Wenn kein EHCI-Treiber verfügbar ist, werden Hi-Speed-Geräte ebenfalls an den USB-1.1-Controller durchgereicht und arbeiten dann soweit möglich mit langsamerer Geschwindigkeit.

Extensible Host Controller Interface Die xHCI-Spezifikation 1.0 wurde im Mai 2010 von Intel veröffentlicht[8] und stellt zusätzlich zu den mit USB

2.0 verfügbaren Übertragungsgeschwindigkeiten den *SuperSpeed*-Modus mit 5 Gb/s bereit.

5.2 Einstellungen und Schnittstellen

Intern adressiert der USB-Controller die angeschlossenen Geräte mit einer sieben Bit langen Kennung, wodurch sich die 127 maximal anschließbaren Geräte ergeben. Wenn an einem Port neue Geräte detektiert werden, schaltet der Host-Controller diesen ein und sendet dem angeschlossenen Gerät einen Reset, indem er beide Datenleitungen für mindestens 10 ms auf Massepotential legt.[9] Dadurch belegt das Gerät zunächst die *Adresse 0* und bekommt dann vom Host eine eindeutige Adresse zugeteilt. Da immer nur ein Port mit noch nicht konfiguriertem Gerät aktiviert wird, kommt es zu keinen Adresskollisionen.

Der Host-Controller fragt meist zuerst nach einem *Device-Deskriptor*, der unter anderem die Hersteller- und Produkt-ID enthält. Mit weiteren Deskriptoren teilt das Gerät mit, welche alternativen Konfigurationen es besitzt, in die es von seinem Gerätetreiber geschaltet werden kann. Bei einer Webcam könnten diese Alternativen etwa darin bestehen, ob die Kamera eingeschaltet ist oder ob nur das Mikrofon läuft. Für den Controller ist dabei relevant, dass die unterschiedlichen Konfigurationen auch einen unterschiedlichen Strombedarf mit sich bringen.

Innerhalb einer Konfiguration kann das Gerät verschiedene Schnittstellen definieren, die jeweils über einen oder mehrere Endpunkte verfügen. Unterschiedlicher Bedarf an reservierter Datenrate wird über sogenannte *Alternate Settings* signalisiert. Ein Beispiel dafür ist eine Kamera (etwa eine Webcam), die Bilder in zwei verschiedenen Auflösungen senden kann. Das Alternate Setting 0 wird aktiviert, wenn ein Gerät keine Daten übertragen möchte und somit pausiert.

5.3 Geräteklassen

Damit nicht für jedes Gerät ein eigener Treiber nötig ist, definiert der USB-Standard verschiedene Geräteklassen, die sich durch generische Treiber steuern lassen. Auf diese Weise sind USB-Tastaturen, -Mäuse, USB-Massenspeicher, Kommunikations- („Communications Device Class“, kurz: CDC) und andere Geräte mit ihren grundlegenden Funktionen sofort verwendbar, ohne dass zuvor die Installation eines spezifischen Treibers notwendig ist. Herstellerspezifische Erweiterungen (die dann einen eigenen Treiber erfordern) sind möglich. Die Information, zu welchen Geräteklassen sich ein Gerät zählt, kann im *Device-Deskriptor* (wenn das Gerät nur einer Klasse angehört) oder in einem *Interface-Deskriptor* (bei Geräten, die zu mehreren Klassen gehören) untergebracht werden.

5.4 Übertragungsmodi

Der USB bietet den angeschlossenen Geräten verschiedene Übertragungsmodi an, die diese für jeden einzelnen Endpunkt festlegen können.

5.4.1 Endpunkte / Endpoint

USB-Geräte verfügen über eine Anzahl von durchnummerierten „Endpunkten“, gewissermaßen Unteradressen des Gerätes. Die Endpunkte sind in den Geräten hardwareseitig vorhanden und werden von der USB SIE (Serial Interface Engine) bedient. Über diese Endpunkte können voneinander unabhängige Datenströme laufen. Geräte mit mehreren getrennten Funktionen (Webcams, die Video und Audio übertragen) haben mehrere Endpunkte. Die Übertragungen von und zu den Endpunkten erfolgen meist unidirektional, für bidirektionale Übertragungen ist deshalb ein *IN*- und ein *OUT*-Endpunkt erforderlich (IN und OUT beziehen sich jeweils auf die Sicht des Hostcontrollers). Eine Ausnahme davon sind Endpunkte, die den sogenannten *Control Transfer Mode* verwenden.

In jedem USB-Gerät muss ein Endpunkt mit Adresse 0 vorhanden sein, über den die Erkennung und Konfiguration des Gerätes läuft, darüber hinaus kann er auch noch weitere Funktionen übernehmen. Endpunkt 0 verwendet immer den Control Transfer Mode. Ein USB-Gerät darf maximal 31 Endpunkte haben: Den Control-Endpunkt (der eigentlich zwei Endpunkte zusammenfasst) und je 15 In- und 15 Out-Endpunkte. Low Speed-Geräte sind auf Endpunkt 0 plus maximal zwei weitere Endpunkte im Interrupt Transfer Mode mit maximal 8 Bytes pro Transfer beschränkt.

5.4.2 Isochroner Transfer

Der isochrone Transfer ist für Daten geeignet, die eine garantierte Datenrate benötigen. Diese Transferart steht für Full-Speed- und Hi-Speed-Geräte zur Verfügung. Definiert das sogenannte *Alternate Setting* einen Endpunkt mit isochronem Transfer, so reserviert der Host-Controller-Treiber die erforderliche Datenrate. Steht diese Datenrate nicht zur Verfügung, so schlägt die Aktivierung des genannten *Alternate Settings* fehl, und es kann mit diesem Gerät keine isochrone Kommunikation aufgebaut werden.

Die erforderliche Datenrate ergibt sich aus dem Produkt des Abfrageintervalls und der Größe des Datenpuffers. Full-Speed-Geräte können jede Millisekunde bis zu 1023 Bytes je isochronem Endpunkt übertragen (1023 kbyte/s), Hi-Speed-Geräte können bis zu drei Übertragungen je Micro-Frame (125 µs) mit bis zu 1024 kbytes ausführen (24 Mbyte/s). Stehen in einem Gerät mehrere isochrone Endpunkte zur Verfügung, erhöht sich die Datenrate entsprechend. Die Übertragung ist mit einer Prüf-

summe (CRC16) gesichert, wird aber bei einem Übertragungsfehler durch die Hardware nicht wiederholt. Der Empfänger kann erkennen, ob die Daten korrekt übertragen wurden. Isochrone Übertragungen werden zum Beispiel von der USB-Audio-Class benutzt, die bei externen USB-Soundkarten Verwendung findet.

5.4.3 Interrupt-Transfer

USB-Maus für Notebooks

Interrupt-Transfers dienen zur Übertragung von kleinen Datenmengen, die zu nicht genau bestimmbaren Zeitpunkten verfügbar sind. Im Endpoint Descriptor teilt das Gerät mit, in welchen maximalen Zeitabständen es nach neuen Daten gefragt werden möchte. Das kleinstmögliche Abfrageintervall beträgt bei Low Speed 10 ms, bei Full Speed 1 ms und bei Hi-Speed bis zu drei Abfragen in 125 µs. Bei Low Speed können pro Abfrage bis zu 8 Byte, bei Full Speed bis zu 64 Byte und bei Hi-Speed bis zu 1024 Byte übertragen werden. Daraus ergeben sich maximale Datenraten von 800 byte/s bei Low Speed, 64 kbyte/s bei Full Speed und bis zu 24 Mbyte/s bei Hi-Speed. Die Daten sind mit einer Prüfsumme (CRC16) gesichert und werden bei Übertragungsfehlern bis zu dreimal durch die Hardware wiederholt. Geräte der HID-Klasse (Human Interface Device), zum Beispiel Tastaturen, Mäuse und Joysticks, übertragen die Daten über den Interrupt-Transfer.

5.4.4 Bulk-Transfer

Bulk-Transfers sind für große und nicht zeitkritisch Datenmengen gedacht, wie beispielsweise das Lesen oder Schreiben von Dateien auf einer USB-Festplatte. Diese Transfers sind niedrig priorisiert und werden vom Controller durchgeführt, wenn alle isochronen und Interrupt-Transfers abgeschlossen sind und noch Datenrate übrig ist. Bulk-Transfers sind durch eine Prüfsumme (CRC16) gesichert und werden durch die Hardware bis zu dreimal wiederholt. Low Speed-Geräte können diese Transferart nicht benutzen. Full-Speed-Geräte benutzen Puffer-Größen von 8, 16, 32 oder 64 Bytes. Hi-Speed-Geräte verwenden immer einen 512 Byte großen Puffer.

5.4.5 Control-Transfer

Control-Transfers sind eine besondere Art von Datentransfers, die einen Endpunkt erfordern, der sowohl In- als auch Out-Operationen durchführen kann. Control-Transfers werden generell in beide Richtungen bestätigt, so dass Sender und Empfänger immer sicher sein können, dass die Daten auch angekommen sind. Daher wird der Endpunkt 0 im Control-Transfer-Modus verwendet. Control-Transfers sind zum Beispiel nach dem Detektieren des USB-Geräts und zum Austausch der ersten Kommunikation elementar wichtig.

5.5 Datenraten

Logo für USB-Low-Speed- oder -Full-Speed-zertifizierte Geräte

Logo für USB-Hi-Speed-zertifizierte Geräte

USB erlaubt es einem Gerät, Daten mit 1,5 Mbit/s (Low Speed), 12 Mbit/s (Full Speed), 480 Mbit/s (Hi-Speed), 5 Gbit/s (SuperSpeed) oder 10 Gbit/s (Superspeed+) zu übertragen.

Diese Raten basieren auf dem Systemtakt der jeweiligen USB-Geschwindigkeit und stellen die physikalische Datenübertragungsrate dar. Die Toleranzen werden für USB-2.0-Geräte und für die älteren USB-1.0-/1.1-Geräte getrennt behandelt. Der tatsächliche Datendurchsatz liegt – durch Protokoll-Overhead – darunter. Im

Logo für USB-SuperSpeed-zertifizierte Geräte

USB-Standard ist für USB 2.0 eine maximale *theoretische* Datenlast bei Hi-Speed unter idealen Bedingungen von 49.152.000 Byte/s (Isochronous Mode)[11] beziehungsweise 53.248.000 Byte/s (Bulk-Mode)[12] angegeben. Dazu kommt die Verwaltung der Geräte, so dass bei aktuellen Systemen für USB 2.0 eine nutzbare Datenrate in der Größenordnung von 320 Mbit/s (40 MB/s) und für USB 3.0 2400 Mbit/s (300 MB/s)[13] bleibt. Bei älteren Systemen wurde diese durch eine unzureichende Anbindung des USB-Chips an den Systembus zusätzlich reduziert.

Bemerkungen

- Die Schreibweise variiert: Low und Full Speed werden mit Leerzeichen getrennt, Hi-Speed mit Bindestrich (und High wird verkürzt zu Hi), SuperSpeed wird zusammengeschrieben.
- SI-Präfixe sind dezimale Präfixe: 1 kBit = 10^3 Bit, 1 MBit = 10^6 Bit, 1 GBit = 10^9 Bit, gleiches für Byte und Hz.
- USB 1.x und USB 2.0 wird mit der Bruttodatenrate beworben (1,5 MBit/s, 12 MBit/s, 480 MBit/s).
- USB 3.0 überträgt mit der Bitrate 5 GBit/s, die effektive Datenrate nach 8b10b-Kodierung beträgt hier 4 Gbit/s (500 MByte/s). Die Bitrate hat allerdings keinerlei Auswirkungen außerhalb des physischen Übertragungslayers (OSI Layer 1). Bei vielen Kodierungen ist sie wesentlich größer als die effektive Datenrate und wird aus verkaufsfördernden Gründen an deren Stelle angegeben.
- Die theoretisch erzielbare Nettodatenrate liegt um 11,3 Prozent (Bulk-Mode) und 18,1 Prozent (Isochron-Modus) unter der Bruttodatenrate.
- Real erzielbare Nettodatenraten liegen um mindestens 30 Prozent, meist aber um die 45 Prozent unter der Bruttodatenrate (reale Messungen an USB-2.0-Systemen).

Wird die Schnittstelle eines Geräts mit „USB 2.0“ angegeben, heißt das nicht unbedingt, dass dieses Gerät auch die hohe Datenrate von 480 Mbit/s anbietet. Standpunkt der Anbieter ist dabei, dass ein USB-2.0-kompatibles Gerät grundsätzlich jede der drei Geschwindigkeiten benutzen kann und die 2.0-Kompatibilität in erster Linie bedeutet, dass die neueste Fassung der Spezifikation eingehalten wird. 480 Mbit/s dürfen also nur erwartet werden, wenn ein Gerät mit dem Logo „Certified USB Hi-Speed“ ausgezeichnet ist.

5.6 USB On-the-go

Logo für USB-OTG-Geräte

Logo für USB-Hi-Speed-OTG-Geräte

Eine direkte Kommunikation zwischen USB-Geräten, also ohne Beteiligung des zentralen Host-Controllers, war im USB-Standard ursprünglich nicht vorgesehen; diese wurde erst durch die Erweiterung *USB On-the-go* eingeschränkt ermöglicht.

Durch USB On-the-go (OTG) können entsprechend ausgerüstete Geräte kommunizieren, indem eines der beiden eine eingeschränkte Host-Funktionalität übernimmt. Typische Einsatzgebiete von USB OTG sind die Verbindung von Digitalkamera und Drucker oder der Austausch von Musikdateien zwischen zwei MP3-Spielern.

Auch bei USB OTG ist die Kommunikation zentral von einem Host gesteuert. Im Gegensatz dazu bieten andere Kommunikationsmechanismen, etwa der FireWire-Standard, der für ähnliche Einsatzzwecke wie USB geschaffen wurde und mit diesem in Konkurrenz steht, die Möglichkeit einer Peer-to-Peer-Kommunikation zwischen Geräten ohne Beteiligung eines zentralen Hosts. Dies bietet etwa die Möglichkeit, ein Netzwerk aufzubauen.

Gekennzeichnet werden USB-OTG-Produkte durch das USB-Logo mit zusätzlichem grünem Pfeil auf der Unterseite und weißem „On-The-Go“-Schriftzug. Die USB-OTG-Spezifikation wurde am 18. Dezember 2001 verabschiedet. OTG-Geräte sind zum Beispiel die seit November 2007 erhältlichen Nokia-Telefone 6500c, N8, C7, N810, 808 PureView, das Samsung Galaxy S II[14] und andere Android-Smartphones, sowie einige externe Festplatten zum direkten Anschluss an Digitalkameras.

5.7 Wireless USB

→ *Hauptartikel: Wireless USB*

Momentan besetzen zwei Initiativen den Begriff

Logo für die zertifizierten Geräte aus dem Intel-Wireless-USB-Projekt

„Wireless USB“. Die ältere der beiden wurde von dem Unternehmen Cypress initiiert, mittlerweile ist Atmel als zweiter Chiphersteller auf den Zug aufgesprungen. Das „Cypress-WirelessUSB“-System ist eigentlich kein drahtloses USB, sondern eine Technik, um drahtlose Endgeräte zu bauen, die dann über einen am USB angeschlossenen Empfänger/Sender (Transceiver) mit dem Computer verbunden sind. Dazu wird eine Übertragungstechnik im lizenzfreien 2,4-GHz-Band benutzt, die Datenrate beträgt bis zu 62,5 kbit/s (neuere Chips von Cypress erreichen 1 Mbit/s) und ist damit für Eingabegeräte völlig ausreichend, für andere Anwendungen aber oft zu knapp bemessen.

Das zweite Wireless-USB-Projekt wird von der USB-IF vorangetrieben und ist wesentlich anspruchsvoller, neben Intel ist auch NEC dabei, entsprechende Chips zu entwickeln. Ziel ist es, eine Technik zu schaffen, mit der die vollen 480 Mbit/s des Hi-Speed-Übertragungsmodus drahtlos übertragen werden können. Dabei ist eine kurze Reichweite unter 10 m vorgesehen; die Übertragung soll auf einer Ultrabreitband-Technik basieren.

Am 16. Januar 2008 gab in Deutschland die Bundesnetzagentur für die Ultrabreitband-Technik Frequenzbereiche frei.[15] Der dabei für USB vorgesehene Bereich von 6 bis 8,5 GHz ist jedoch nicht so breit wie von USB-IF spezifiziert, so dass Geräte aus anderen Ländern eventuell in Deutschland nicht verwendet werden dürfen.[16]

5.8 USB 3.0

Im November 2008 stellte das *USB Implementers Forum*, dem unter anderem die Unternehmen Hewlett-Packard, Microsoft und Intel angehören, die Spezifikation für *USB 3.0* vor. Im *SuperSpeed*-Modus wird eine Bitrate von exakt 5 Gbit/s verwendet, was aufgrund der verwendeten ANSI-8b10b-Kodierung eine Nettodatenrate von 500 MByte/s ergibt.[17][18] Die höheren Datenraten werden im Wesentlichen durch höhere Frequenzen (ca. achtfach) auf den Datenleitungen möglich. Die Bruttodatenrate steigt dadurch von 60 MByte/s auf 625 MByte/s. Das stellt allerdings erheblich höhere Anforderungen an die Kabel. Weitere Gewinne sind durch das verbesserte USB-Protokoll sowie durch die verwendete Vollduplex-Übertragung möglich.

Die verwendeten Kabel enthalten neben den bisherigen Signal-Adernpaar (D+ und D−) und der Stromversorgung (GND, VBUS) zwei weitere Signal-Adernpaare (SSTX+ und SSTX−, SSRX+ und SSRX−) sowie eine zusätzliche Masseverbindung (GND). Deshalb sind für USB 3.0 sowohl neue Stecker am Host und an den angeschlossenen Geräten als auch neue Kabel notwendig. Die Kabel sind auf Grund der gestiegenen Aderanzahl und der notwendigen besseren HF-Übertragungseigenschaften (ähnlich wie eSATA- oder CAT-5e-/6-Kabel) dicker und weniger flexibel.

Die Kompatibilität besteht in folgendem Sinne:

- USB-3.0-Kabel können auf Grund der Anbauten nicht mit USB-2.0-Endgeräten benutzt werden – USB 3.0-Typ-B-Stecker sind nicht abwärtskompatibel.
- USB-3.0-Kabel können an USB-2.0-Hosts benutzt werden, erfordern dann aber USB-3.0-Endgeräte.
- USB-2.0-Kabel können an USB-3.0-Hosts benutzt werden.
- USB-3.0-Endgeräte können an USB-2.0-Hosts angeschlossen werden. Ggf. gibt es Probleme, wenn diese mehr als 500 mA Strom aufnehmen (USB 3.0 erlaubt bis zu 900 mA, USB 2.0 nur bis 500 mA).
- USB-2.0-Endgeräte können an USB-3.0-Hosts angeschlossen werden.

USB-3.0-Übertragungen finden aber nur statt, wenn *alle* drei Komponenten (Host, Kabel, Endgerät) USB-3.0-tauglich sind. Ansonsten wird auf USB 2.0 heruntergeschaltet.

Mit dem Linux-Kernel ab Version 2.6.31 waren Linux-Distributionen die ersten Betriebssysteme, die USB 3.0 unterstützten.[19]

Weitere Besonderheiten:

- Die bei den bisherigen USB-Standards übliche Reihumabfrage der Geräte (Polling) ist nicht mehr notwendig. Durch das (mögliche) Entfallen dieses dauernden Pollings und durch neue Befehle können Geräte in die Energiesparmodi U0 bis U3 geschaltet werden.
- Am USB-3.0-Port stehen mindestens 150 mA Strom (statt 100 mA wie bei USB 2.0) pro Gerät zur Verfügung. Auf Anforderung können bis 900 mA bereitgestellt werden (USB 2.0 Low Power: 100 mA, USB 2.0 High Power: 500 mA).
- Ältere Treiber sollen weiterverwendbar bleiben. Neuere Versionen sind aber unter Umständen vorteilhaft, etwa um die neuen Stromsparmodi zu nutzen.
- USB-3.0-Hubs nutzen keinen Transaction Translators wie USB-2.0-Hubs (Hi-Speed). Daher hat man keinen Gewinn, wenn man mehrere USB-2.0-Geräte über einen USB-3.0-Hub an einen PC anschließt.
- Zu Hubs siehe USB 3.0 und Hubs

Für einen späteren Zeitpunkt ist auch eine Erweiterung des Standards mit Lichtwellenleitern geplant. Anders als noch bei USB 2.0 dürfen sich Geräte nur dann „USB-3.0-kompatibel“ nennen, wenn sie tatsächlich die schnellstmögliche Geschwindigkeit (hier SuperSpeed-Modus) anbieten.[13]

5.9 USB 3.1

Im Januar 2013 kündigte die USB Promoters Group eine Geschwindigkeitsverdopplung gegenüber USB 3.0 an. Damit sollen 10 Gbit/s brutto erreicht werden, wobei vorhandene USB 3.0-Kabel weiterverwendet werden können.[20] Ebenfalls wurde bekanntgegeben, dass sich zum ersten mal seit 17 Jahren ein neuer Anschluss namens Typ C in der Entwicklung befindet. Dieser wird nicht mehr abwärtskompatibel sein, aber erstmals das Einstecken in jeder Richtung und jeder Orientierung ermöglichen. Der USB-Stecker vom Typ C soll Teil des bisherigen Standards für USB 3.1 werden und auch viele bisher darin vorgesehene, aber kaum genutzte Funktionen, wie das Aufladen mit hohen Stromstärken, unterstützen. Dieses *Power Delivery* genannte Feature wird in sechs Profilen (0–5) jeweils 10, 18, 36, 60 und 100 Watt leisten können.[21] Seine Größe wird laut einer Mitteilung des USB-Implementers Forum in etwa beim bisherigen Micro-USB 2.0 liegen, die Stecker sollen besonders flach sein. Damit passen die neuen Ports besser an portable Geräte wie Smartphones und Tablets. Auch durch die nicht ideale Mechanik sind die Micro-USB-Buchsen in Typ 3.0 dort bisher nur selten zu finden.[22][23] Mittlerweile wurde die Spezifikation 1.0 zum neuen Typ, welche die nötigen Stecker, Kabel und Buchsen definiert, vom Konsortium veröffentlicht. Diese sollen langfristig alle anderen USB-Verbindungen ersetzen und sind für eine Haltbarkeit von 10.000 Ansteckvorgängen konzipiert. Der Standard sieht ausdrücklich passive Kabel ohne integrierte Elektronik vor. Außerdem wird die Abwärtskompatibilität zu Typ A mit der von USB 3.0 maximalen Übertragungsgeschwindigkeit von 5 GBit/s garantiert. Anfangs sind Adapter für die älteren Buchsen vorgesehen.[24] Laut Video Electronics Standards Association (VESA) soll der neue USB 3.1-Anschluss auch die Displayport-Technologie unterstützen. Damit könnten Displays mit 4K / Ultra HD-Auflösung (3.840 x 2.160 Pixel) mit einer Bildwiederholungsrate von 60 Hz angeschlossenen werden. Verzichtet man auf die USB 3.1-Funktionalitäten und nutzt alle Daten-Lanes für die Übertragung des Videosignals, ist sogar 5K-Auflösung (5.120 x 2.880 Pixel) möglich. [25]

6 Hardware

6.1 USB-Stecker und -Buchsen

- Verschiedene USB-Stecker; von links nach rechts: Typ A, Typ B, Typ Mini-B 5-polig (Standard), Typ Miniatur-B 4-polig (Mitsumi), Typ Miniatur-B 4-polig (Aiptek)
- USB Typ A Stecker; deutlich zu erkennen sind die voreilenden äußeren Pins für die Versorgungsspannung
- USB 3.0 Typ A Stecker
- USB Typ A Buchse (1.0 und 2.0)
- 12-V- und 24-V-USB Typ-A-Buchsen samt hochstromfähigem vierpoligen Anschlüssen (poweredUSB)
- USB Typ-B-Stecker
- USB 3.0 Typ-B-Stecker
- USB 3.0 Typ-B-Buchse, Typ-A-Stecker, Typ-A-Buchse und Stack mit Typ-A-Buchsen
- USB 2.0 Mini-B-Stecker
- USB 3.0 Mini-B Buchse
- USB Micro-B Stecker (bei Steckernetzteilen für Mobiltelefone verbreitet)
- USB 3.0 Micro-B Stecker
- USB 3.0 Micro-B Buchse

Die Stecker eines USB-Kabels sind verpolungs- und vertauschungssicher gestaltet.

In Richtung des Hostcontrollers *(Upstream)* werden flache Stecker *(Typ A „DIN IEC 61076-3-107“)* verwendet. Zum angeschlossenen Gerät hin *(Downstream)* werden die Kabel entweder fix montiert oder über annähernd quadratische Steckverbinder *(Typ B „DIN IEC 61076-3-108“)* angeschlossen (vereinzelt und nicht standardkonform auch mit Typ-A-Steckverbindern). Entsprechend den USB-1.0–2.0-Standards besitzen USB-Typ-A- und Typ-B-Verbinder vier Leitungen plus Schirm. Beide Steckverbinder sollen in einer der drei Farben grau, „natur“ (elfenbeinfarben/weiß) oder schwarz ausgeführt werden. Mit USB 3.0 kommen neue Varianten der Typ-A- und Typ-B-Verbinder auf den Markt (siehe unten).

Seit einiger Zeit sind auch Stecker und Buchsen vom Typ A und B mit Rändelschrauben erhältlich, die ein Herausrutschen verhindern. Allerdings muss das empfangende Gerät das auch unterstützen. Verschiedene Hersteller brachten mechanisch inkompatible Ausführungen von USB-Verbindern heraus, die sich jedoch elektrisch nicht von USB 1.x oder 2.0 unterscheiden. So waren einige IBM Thinkpads mit einem sogenannten „UltraPort“ ausgestattet, APC führt USB an ihren USVs über 10-polige Modular-Buchsen (10P10C/RJ50), die Microsoft Xbox benutzt ebenfalls proprietäre USB-Verbinder, und Apple führt USB beim iPod Shuffle über einen Klinkenstecker, der gleichzeitig als Audioverbinder dient. Diese nicht standardisierten Varianten sind jedoch nicht sehr verbreitet.

Für den industriellen Einsatz gibt es mehrere nicht vom USB-Konsortium standardisierte USB-5-V(olt)-, USB-12-V-, USB-19-V- und USB-24-V-Varianten mit deutlich höheren Strombelastbarkeiten von bis zu 6 A (3 A pro Kontakt) über insgesamt vier zusätzliche Leitungen, die um 1999 im Rahmen der PoweredUSB- und PlusPower-Spezifikationen von Firmen wie IBM, Microsoft, NCR und Berg/FCI definiert wurden und zum Teil lizenzpflichtig sind. Diese Varianten werden insbesondere bei POS-Anwendungen, etwa in Kassensystemen, von verschiedenen Herstellern eingesetzt. Die Steckverbinder führen dabei neben dem USB-Typ-A-Stecker eine unabhängige hochstromfähige vierpolige Spannungsversorgung. Diese Stecker sind nicht rechteckig, sondern mehr quadratisch (wie zwei Stecker in einem gemeinsamen Gehäuse, der USB-Teil selbst entspricht mechanisch und elektrisch unverändert USB Typ A). Mittels einer mechanischen Kodierung wird verhindert, dass zum Beispiel USB-12-V-Stecker versehentlich in USB-24-V-Buchsen gesteckt werden können. Eine mechanische Arretierung der Stecker in den Buchsen ist ebenfalls vorgesehen. Zusätzlich wird für diese Stecker eine Farbkodierung empfohlen, naturfarben (teilweise auch gelb) für 5 V (30 W), blaugrün (Pantone Teal 3262C) für 12 V (72 W), rot (Pantone Red 032C) für 24/25 V (144 W) und seltener violett für 19 V. Kommt keine Farbkodierung zum Einsatz, sollen die Stecker für alle Spannungen größer 5 V schwarz ausgeführt werden, wohingegen grau als alternative Farbe für 5 V in Frage kommt. Für die B-Seite ist kein spezieller Stecker definiert, es gibt jedoch verschiedene Empfehlungen, teilweise mit unterschiedlichen HotPlug-Fähigkeiten. Die Bezeichnung für diese industriellen USB-Varianten lautet *Retail USB*, *PoweredUSB*, *USB PlusPower* oder *USB +Power*.[26]

6.1.1 Micro- und Mini-USB

Insbesondere für Geräte mit geringerem Platzangebot (digitale Kameras, Mobiltelefone, MP3-Player und andere mobile Geräte) existieren auch verschiedene kompaktere USB-Steckverbinder. Im USB-2.0-Standard verankert sind dabei lediglich fünfpolige Mini- und Micro-Varianten (plus Schirm), die gegenüber den normalen USB-Steckverbindern über einen zusätzlichen ID-Pin verfügen.

Zunächst wurde im Jahr 2000 ein trapezförmiger Mini-B-Steckverbinder für die Downstream-Seite definiert, der in der Farbe Schwarz ausgeführt werden sollte. Bei zukünftigen Geräten sollen Geräteherstellerjedoch auf die Micro-USB-Verbinder (siehe unten) ausweichen.[27] Auch Mini-A- (in weißer Farbe) und Mini-AB-Steckverbinder (in Grau) waren für eine gewisse Zeit Teil des Standards und sollten insbesondere in Verbindung mit USB On-the-Go (OTG) eine Rolle spielen, wurden jedoch im Mai 2007 offiziell zurückgezogen.[28]

Im Januar 2007 wurden mit der Standarderweiterung Micro-USB für USB 2.0 noch kleinere Steckverbinder vorgestellt, die eine besonders kompakte Bauform der Geräte ermöglichen. Die Micro-USB-Spezifikation kann USB On-the-Go (OTG) unterstützen, was Verkabelung und Kommunikation auch ohne PC als Host ermöglicht.[29] Micro-USB-Steckverbinder sollen bei neueren Geräten in naher Zukunft den Mini-Verbinder komplett ersetzen, lediglich der relativ weitverbreitete Mini-B-Verbinder wird derzeit noch geduldet. Die Micro-USB-Verbinder sind elektrisch gleichwertig, mechanisch allerdings nicht steckkompatibel, dafür jedoch dank der im Standard geforderten Edelstahlkrampe deutlich stabiler ausgeführt. Gemäß USB-2.0-Standard gibt es drei Varianten, die genau wie bei Mini-USB allesamt fünfpolig ausgeführt sind: Micro-A (rechteckige Bauform, für die Host-Seite, Farbe Weiß), Micro-AB (rechteckige Bauform, für USB-On-the-Go-Geräte, Farbe Grau) und Micro-B (Trapez-Bauform, für die Geräteseite, Farbe Schwarz). Die Open Mobile Terminal Platform OMTP hat Micro-USB 2007 als Standardverbinder für den Datentransfer und die Energieversorgung von Mobilfunkgeräten übernommen, in China müssen Mobiltelefone seitdem mit dieser Schnittstelle ausgestattet werden, um eine Zulassung zu bekommen.[30] Mit USB 3.0 kommen neue Varianten der Micro-A-, AB- und -B-Steckverbinder auf den Markt (siehe unten).

Daneben gibt es noch eine ganze Reihe proprietärer, das heißt geräteherstellerspezifische Miniaturbauformen der Steckverbinder (siehe auch Bild), die zwar in der Regel elektrisch mit USB 2.0 kompatibel sind, jedoch nur über teilweise schwer erhältliche Adapterkabel mit USB-Komponenten gemäß dem USB-Standard verbunden werden können. Fälschlicherweise werden jedoch auch diese Steckverbinder häufig als „Mini"-USB bezeichnet, was immer wieder zu Missverständnissen führt und vermieden werden sollte. Verbreitet sind unterschiedlichste Ausführungen mit vier Pins (insbesondere Varianten von Mitsumi, Aiptek, Hirose), eine große Zahl von Varianten mit acht Pins (darunter mehrere inkompatible Varianten, die sich bei Digitalkameras in begrenztem Rahmen auch über Herstellergrenzen hinweg verbreitet haben), elf Pins (*ExtUSB* für HTC-Mobiltelefone; kompatibel zu Mini-USB), zwölf Pins (für verschiedene Olympus-Digitalkameras) und 14 Pins (zwei Varianten für verschiedene Fuji-Finepix-Digitalkameras und als Nokias Pop-Port für manche Mobiltelefone), die auch noch andere, nicht-USB-spezifische Signale (bei Digitalkameras z.B. Analog-Video und -Audio) im gleichen Konnektor vereinen.[31] Zur Eindämmung dieser Vielfalt ist seit 2011 bei Netzteilen für die Geräteklasse der Smartphones der Micro-USB-Stecker europaweit genormt (EN 62684:2010, „Micro-USB-Standard").

Im Rahmen des im Jahr 2008 verabschiedeten USB-3.0-Standards wurden weitere sechs Steckverbindertypen mit zusätzlichen Kontakten definiert:

Diese unterteilen sich in je drei Steckverbinder, die als weitestgehend rückwärtskompatible Erweiterungen der bisherigen Typ A- und Typ B-Steckverbinder angesehen werden können (genannt: USB 3.0 Standard-A, USB 3.0 Standard-B und USB 3.0 Powered-B) sowie drei kleinere Verbinder, die sich an die bisherigen Micro-USB-Verbinder anlehnen (genannt: USB 3.0 Micro-A, USB 3.0 Micro-AB und USB 3.0 Micro-B). Zur eindeutigen Kennzeichnung werden die bisherigen Steckverbinder nun als USB 2.0 Standard-A, USB 2.0 Standard-B, USB 2.0 Micro-A, USB 2.0 Micro-AB und USB 2.0 Micro-B bezeichnet. Zur besseren Unterscheidung sollen die USB-3.0-Standard-A-Verbinder in der Farbe Blau (Pantone 300C) ausgeführt und gegebenenfalls mit einem doppelten S-Symbol gekennzeichnet werden.

6.1.2 USB Typ C

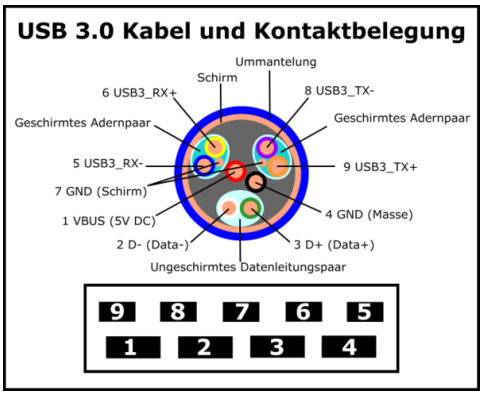
Im August 2014 wurde die Spezifikation für eine neue, zur bisherigen Hardware inkompatible Typ C-Steckverbindung verabschiedet.[32] Die neue Steckverbindung hat keine definierte Ober- und Unterseite mehr und kann somit in beliebiger Richtung eingesteckt werden. Zudem vereint sie alle bisherigen Übertragungsspezifikationen inklusive USB 3.1. und USB Power Delivery.

6.1.3 Spezifikationen

Verbreitet haben sich weiterhin 1×4-, 1×5- und 2×2-polige Varianten von Stiftleisten im Rastermaß 2,54 Millimeter auf PC-Mainboards, ebenso wie Doppel-USB-Verbinder mit 2×4 oder 2×5 Polen im Rastermaß 2,54 mm. Gab es zunächst mehrere zueinander inkompatible Belegungsvarianten, hat sich im Zuge neuerer Mainboard-Spezifikationen von Intel inzwischen eine bestimmte 2×5-polige Belegung etabliert, die auch mit uDOC-Flashmodulen kompatibel ist.

6.2 USB-Kabel

USB-Verlängerungskabel (nicht in der USB-Spezifikation)



USB-3.0-Kabel und Kontaktbelegung

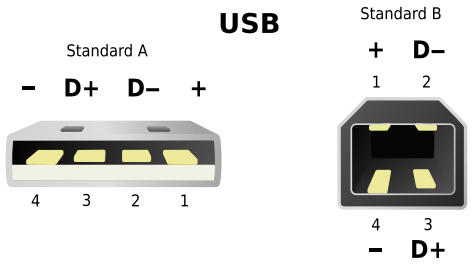
In einem USB-Kabel werden vier Adern benötigt. Zwei Adern übertragen dabei die Daten, die anderen beiden versorgen das angeschlossene Gerät mit einer Spannung von 5 V. Der USB-Spezifikation entsprechende Geräte dürfen bis zu 100 mA oder 500 mA aus dem Bus beziehen, abhängig davon, wie viel der Port liefern kann, an den sie angeschlossen werden. Geräte mit einer Leistung bis zu 2,5 W können also über den Bus versorgt werden. Je nach Kabellänge muss der Querschnitt der beiden Stromversorgungsadern angepasst sein, um den zulässigen Spannungsabfall einzuhalten; auch daher sind Verlängerungsleitungen nicht standardgemäß.

Die Kabel müssen je nach Geschwindigkeit unterschiedlich abgeschirmt werden. Kabel, die lediglich der Spezifikation *Low Speed* entsprechen, dürfen über keinen B-Stecker verfügen, sondern müssen fix am Gerät montiert sein oder einen herstellerspezifischen Stecker verwenden. Sie sind weniger stark abgeschirmt, kommen ohne verdrillte Adern aus und sind dadurch flexibler als Full-/Hi-Speed-Kabel. Sie sind daher gut für zum Beispiel Mäuse und Tastaturen geeignet. Die geringe Abschirmung des Kabels würde zu Problemen bei Geräten mit höheren Geschwindigkeiten führen.

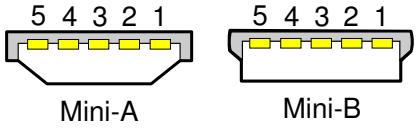
Die Längen von Full-/Hi-Speed- und Low Speed-Kabeln vom Hub zum Gerät sind auf fünf beziehungsweise drei Meter begrenzt. Längere Strecken kann man überwinden, indem USB-Hubs zwischengeschaltet werden. Sogenannte *USB-Repeaterkabel* entsprechen in ihren Funktionen einem *Bus-Powered Hub* (s. u.) mit einem einzigen Downstream-Port und einem fest angeschlossenen Kabel am Upstream-Port. Da die elektrischen Auswirkungen dieser Kabel im USB-Bus denen eines Bus-Powered-USB-Hubs mit fünf Meter Kabel entsprechen, sollten bei ihrer Verwendung zusätzlich die Beschränkungen beim Kaskadieren von USB-Hubs beachtet werden.

USB arbeitet mit einem Wellenwiderstand von 90 Ω, direkte Verbindungskabel sollten daher auch in diesem Wellenwiderstandswert ausgeführt sein. Für die Überbrückung von Längen über 30 Metern werden USB-Line-Extender angeboten. Diese bestehen aus zwei Komponenten: Einem Base-Modul, das an den Computer angeschlossen wird, und einem Remote-Modul für den Anschluss des USB-Gerätes. Zur Distanzüberbrückung zwischen diesen beiden Komponenten werden meist Ethernetkabel oder Lichtleiter eingesetzt. Da sich diese Line-Extender jedoch immer auf bestimmte, nicht vom Standard vorgeschriebene Verhaltensdetails der angeschlossenen Geräte verlassen und zudem bei langen Kabelstrecken die Signallaufzeit zu Protokollverletzungen führt, ist der Einsatz dieser Geräte oft mit Problemen verbunden.

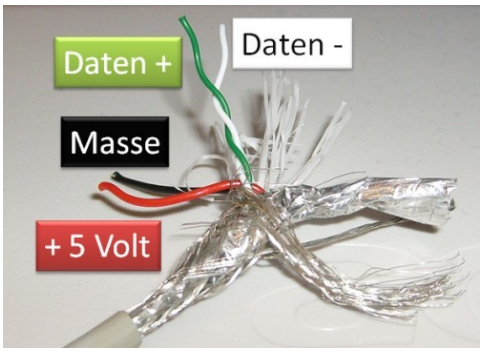
Eine andere Möglichkeit, USB-Geräte weiter entfernt vom Rechner anzuschließen, sind Lösungen, die einen „remote host“ verwenden, also einen USB-Hostcontroller, der außerhalb des PCs liegt. Dabei geschieht die Kommunikation zwischen PC und Hostcontroller zum Beispiel über Ethernet. Das Ethernet ersetzt dabei den lokalen Bus, an dem sonst der Hostcontroller angeschlossen wäre. Auf dem PC muss also nur ein entsprechender Treiber installiert werden, der die Kommunikation mit dem Hostcontroller übernimmt. Treiber für die USB-Geräte erkennen dann keinen Unterschied zu einem lokal angeschlossenen Gerät. Beispiele für ein solches Gerät sind der USB-Server von Keyspan und die USB-Fernanschlussfunktion einer FRITZ!Box.



USB-Standardstecker Typ A und B. Nicht maßstabsgetreu, mit Pinnummern



USB-Ministecker Typ A und B. Nicht maßstabsgetreu, mit Pinnummern, Draufsicht. Es gibt noch Mini-AB-Buchsen, die sich automatisch umschalten. Mini-AB-Buchsen und Mini-A-Stecker sind aus der Spezifikation entfernt worden.



Kabelbelegung eines normalen USB-Kabels

6.2.1 Farbkodierung und Pinouts

Der USB-Standard legt neben der Belegung der Schnittstelle auch die Namen der einzelnen Stecker-Pins fest, für die Aderfarbe werden nur typische Angaben gemacht. Tatsächlich variieren die verwendeten Aderfarben von Hersteller zu Hersteller. Die Nummer eines Stecker-Pins kann in den oben angeführten Schemazeichnungen abgelesen werden.

6.3 USB-Hubs

6.3.1 Allgemeines

Ein USB-Hub ist ein USB-Gerät, welches das USB-Signal an mehrere Ports verteilt. Handelsüblich sind USB-Hubs

*4-Port-USB-Hub mit eigener Stromversorgung (*self powered, *Netzteilbuchse rechts)*

4-Port-USB-Hub mit Stromversorgung aus dem Bus (bus powered)

mit bis zu sieben Downstream-Ports, vereinzelt sind aber inzwischen auch Hubs mit bis zu 28 Ports zu finden.[34]

Hubs können ihren Strom aus dem Bus selbst beziehen *(als Bus-Powered oder passiver Hub bezeichnet)* oder über eine eigene Stromversorgung verfügen *(als Self-Powered oder aktiver Hub bezeichnet)*. Die meisten Self-Powered-Hubs werden über ein Steckernetzteil mit Strom versorgt. Manche Monitore haben auch einen USB-Hub eingebaut, der über die Stromversorgung des Monitors mitgespeist wird. Self-Powered-Hubs haben den Vorteil, dass jedes an sie angeschlossene Gerät bis zu 500 mA Strom beziehen kann. Bei Bus-Powered-Hubs dürfen der Hub und alle an ihn angeschlossenen Geräte gemeinsam maximal 500 mA beziehen. Hybride Self- und Bus-Powered-Hubs sind möglich – der Hub ist dann Self-Powered, wenn ein Netzteil an ihn angeschlossen ist, ansonsten Bus-Powered.

Bei der Verschachtelung von Hubs werden die Grenzen durch die maximal 127 möglichen USB-Geräte pro root-hub und durch die Signallaufzeit festgelegt – jeder Hub erhöht die Laufzeit, die Verschachtelungstiefe ist auf maximal fünf (Hub-)Ebenen unterhalb des Hostcontrollers beziehungsweise des Root-Hub begrenzt. Die maximale Distanz zwischen zwei mit USB verbundenen Geräten liegt wegen der Beschränkung von 5 m pro USB-Kabel bei 30 m – sechs Kabel mit je fünf Metern Länge und dazwischen fünf Hubs.

Bei integrierten Bausteinen für USB-Hubs dominiert die Ausstattung mit vier Ports (das gilt insbesondere für Bausteine in eher preisgünstigen Hubs). Wird eine größere Anzahl von USB-Anschlüssen benötigt, können mehrere Bausteine kaskadiert werden. Infolgedessen belegt ein solcher Hub mehrere USB-Adressen und ist gleichbedeutend mit mindestens zwei hintereinander geschalteten Hubs.

6.3.2 USB 2.0 und Hubs

Low-, Full- und Hi-Speed-Geräte lassen sich an einem USB-2.0-Host fast beliebig mischen, ohne dass Geschwindigkeitsnachteile entstehen. Hubs nach dem USB-1.x-Standard können an USB-2.0-Hosts verwendet werden. Geräte, die direkt oder indirekt an einen solchen Hub angeschlossen werden, können allerdings lediglich die Geschwindigkeit *Full Speed* erreichen, also 12 Mbit/s. Ein USB-2.0-Host und ein USB-2.0-Hub kommunizieren immer mit Hi-Speed, selbst wenn an dem Hub Low- oder Full-Speed-Geräte angeschlossen sind. Es ist Aufgabe des Hubs, die Daten dieser Geräte in das Hi-Speed-Protokoll zu verpacken, dazu hat er einen oder mehrere sogenannte „Transaction Translators“ eingebaut. Die Anzahl der Transaction Translators bestimmt, wie viele langsame Geräte an einen USB-2.0-Hub angeschlossen werden können, ohne sich gegenseitig auszubremsen. Wird diese Zahl überschritten, so bricht die Datenrate aller an diesen Host angeschlossenen Low Speed- und Full-Speed-Geräte auf Geschwindigkeiten deutlich unter denen eines USB-1.1-Hosts ein; der Durchsatz von Hi-Speed-Geräten am selben Hub bleibt jedoch unbeeinflusst. An der Spezifikation des Stromverbrauchs hat sich bei USB 2.0 im Vergleich zu USB 1.1 nichts geändert.

6.3.3 USB 3.0 und Hubs

Unter USB 3.0 gibt es ein neues Hub-Konzept. Hubs bestehen aus zwei Unter-Hubs. Der eine ist speziell für den neuen SuperSpeed-Modus zuständig, der andere für die bisherigen Geschwindigkeitsmodi (Low Speed, Full Speed, Hi-Speed). Erst an den Ports werden beide Teile zusammengeführt. Im Dezember 2009 hat das Unternehmen VIA die ersten Chips für USB-3.0-Hubs vorgestellt. Der VL810 genannte Chip ist mit allen Geschwindigkeitsmodi kompatibel.[35]

USB-2.0-CardBus-Controller

6.4 USB-Card-Bus

Der *Cardbus*-Standard[36] (PC Card Standard 5.0) wurde ursprünglich für PCMCIA-Karten als Datenspeichermedium entwickelt, unterscheidet sich aber vom eigentlichen PCMCIA-Standard durch eine völlig andere Architektur. Es sind auch Steckkarten mit CardBus-Controller am Markt erhältlich, die USB in CardBus umsetzen, so dass USB-Stecker beispielsweise auch an Mobilgeräten ohne integrierte USB-Schnittstelle verwendet werden können – sie sind aber auf den 32 Bit breiten CardBus beschränkt. Ein Nachrüsten bei Computern mit 16-Bit-Bus ist daher nicht möglich.[37] CardBus wurde vom neueren und leistungsfähigeren ExpressCard-Standard abgelöst.

7 Software-Architektur

Alle USB-Transaktionen werden durch die USB-Software auf dem Host-Computer realisiert. Das geschieht durch den jeweiligen USB-Gerätetreiber, der mit seinem Gerät kommunizieren will. Der USB-Bustreiber ist die Schnittstelle zwischen dem USB-Gerätetreiber und dem USB-Host-Controller.

7.1 USB-Bustreiber

Der USB-Bustreiber (USB-Driver) kennt die spezifischen Kommunikationseigenschaften der einzelnen USB-Geräte, zum Beispiel die Datenmenge pro Frame oder Abstände zwischen den periodischen Zugriffen. Er erkennt diese Eigenschaften beim Analysieren der Geräte-Deskriptoren während der Konfigurationsphase. Wenn der USB-Bustreiber ein IRP von einem USB-Gerätetreiber erhält, erzeugt er entsprechend diesem Request einzelne Transaktionen, die innerhalb des Übertragungsrahmens (Frame) von einer Millisekunde ausführbar sind.

7.2 USB-Host-Controller-Treiber

Der Universal-Serial-Bus-Host-Controller-Treiber (*host controller driver*) organisiert die zeitliche Abfolge der einzelnen Transaktionen (Scheduling). Dazu baut er eine Folge von Transaktionslisten auf. Jede dieser Listen besteht aus den noch nicht abgearbeiteten Transaktionen in Richtung eines Gerätes, das am Bus angeschlossen ist. Sie definiert die Reihenfolge der Transaktionen innerhalb des 1-ms-Zeitrahmens. Der USB-Bustreiber kann eine einzelne Anfrage für einen Datentransfer in mehrere Transaktionen zerlegen. Das Scheduling hängt von einer Reihe von Einflussfaktoren wie Transferart, Geräteeigenschaften und Busbelastung ab. Der USB-Host-Controller-Treiber löst die Transaktionen dann über den Root-Hub aus. Dieser setzt der Reihe nach alle Transaktionen um, die in der aktuellen Liste enthalten sind.

7.3 Unterstützung in Betriebssystemen

- Amiga OS4.x unterstützt seit Version 4.0 USB1.1. Seit dem AmigaOS4.1 Update3 unterstützt es auch USB2.0.
- Amiga OS3.x unterstützt standardmäßig kein USB. Lediglich mit Hard- und Software anderer Anbieter (Poseidon, Sirion, Anaiis) ist eine Anbindung von USB-1.1- und USB-2.0-Geräten möglich (mit breiter Unterstützung verschiedenster Geräteklassen bei Poseidon). Bei Poseidon kann in Zusammenarbeit mit einer Flash-Rom-Karte sogar von USB-Massenspeichern gebootet werden. Ab Amiga OS4 wird – je nach Hardware – USB 1.1 und 2.0 unterstützt (kein USB 2.0 Hi-Speed, da der EHCI-Treiber noch fehlt). Unter AmigaOS 4 Classic kann jedoch alternativ auch Poseidon eingesetzt werden.
- AROS enthält seit August 2009 eine quelloffene Portierung von Poseidon, der die alte Implementierung ersetzt. Es unterstützt OHCI/UHCI (USB 1.1) und EHCI (USB 2.0 Hi-Speed) sowie die meisten der in Poseidon für AmigaOS vorhandenen Gerätetreiber. Der Stack liegt (teilweise) im Kernel und es kann damit von USB-Massenspeichern gebootet werden.
- Atari MiNT unterstützt standardmäßig kein USB, es sind jedoch für MiNT verschiedene Treiber in Entwicklung, die Add-on-Karten (wie EtherNAT, eine Kombination aus USB- und Ethernet Erweiterung für den Atari Falcon) unterstützen.
- eComStation als Nachfolger von OS/2 bringt ebenfalls Unterstützung für USB 2.0 mit.
- Der Linux-Kernel unterstützt seit Version 2.2 USB-Controller. Seit der Kernelversion 2.4 sind Treiber für UHCI-, OHCI- und EHCI-Controller sowie Unterstützung für gängige USB-Endgeräte integriert. Die Unterstützung für EHCI-Controller in

der Kernelversion 2.4 gilt jedoch als fehleranfällig und läuft erst seit Version 2.6 stabil. Weiterhin existieren sogenannte Gadget-Treiber, damit kann ein Linux-basiertes System, das an einem USB-Host angeschlossen wird, selbst als USB-Gerät erscheinen, zum Beispiel als Massenspeicher, Netzwerkkarte oder serielle Schnittstelle. Seit der Version 2.6.31 wird auch USB 3.0 vom Linux-Kernel unterstützt.[38]

- Mac OS unterstützt USB 1.1 ab Mac OS 8.1. Mit der Zeit wurde der Umfang an Geräten, die mit Klassentreibern unterstützt werden, deutlich erweitert; seit Mac OS 8.5 werden die meisten üblichen Geräteklassen unterstützt.
- Mac OS X unterstützt in allen Versionen USB 1.1 und ab Version 10.2.8 auch USB 2.0. Die aktuelle Version unterstützt auch USB 3.0.
- Microsoft DOS und kompatible unterstützen USB standardmäßig nicht. USB-Tastaturen und USB-Massenspeicher sind über die Legacy-Emulation vieler moderner PC-BIOSe dennoch verwendbar, aber meist nicht Hotplug-fähig. Auch USB-Mäuse funktionieren meist mit für PS/2-Mäuse vorgesehenen Treibern, wenn der Legacy-Mode aktiviert ist. Für Free-DOS gibt es den „motto hairu“-Treiber, der USB 2.0 zur Verfügung stellt. Andere Hersteller bieten Spezialtreiber an, die aber viel konventionellen Speicher belegen und deshalb mit vielen DOS-Programmen nicht kompatibel sind.
- Microsoft Windows 95 hat ab OEM Service Release 2.1 eine rudimentäre Unterstützung von USB 1.0, die jedoch als so fehlerhaft gilt, dass eine Verwendung meist nicht möglich ist.
- Microsoft Windows 98 unterstützt USB 1.0, ab Windows 98 SE auch USB 1.1. USB 2.0 ist nur mit Treibern von Chipsatzherstellern möglich.
- Microsoft Windows Me unterstützt USB 1.1. und verfügt als einziges System der 9x-Serie über einen generischen Gerätetreiber für Massenspeicher. USB 2.0 ist nur mit Treibern von Chipsatzherstellern möglich. Im Gegensatz zu Windows 98 und 95 ist nach der Installation gerätespezifischer USB-Treiber kein Neustart erforderlich.
- Microsoft Windows NT 4.0 hat keinerlei USB-Unterstützung, von anderen Herstellern sind jedoch Systemerweiterungen dafür erhältlich. Gerätehersteller testen ihre Produkte selten mit derartigen Erweiterungen, deshalb gelten diese Systemerweiterungen nur für Spezialfälle als tauglich.
- Microsoft Windows 2000 (SP4), Microsoft Windows XP (ab SP1), Microsoft Windows Server 2003, Microsoft Windows Vista, Microsoft Windows Server 2008, Microsoft Windows 7 und Microsoft Windows Server 2008 R2 unterstützen USB 1.1 und USB 2.0. und unterstützen generisch von Anfang an Massenspeicher. Weil der USB-Hostcontroller allerdings manchmal fehlerhaft erkannt wird, raten die meisten Hersteller dazu, die Treiber des Chipsatzherstellers zu installieren.
- Microsoft Windows 8 unterstützt USB 1.0, 1.1, 2.0, 3.0.
- MorphOS wird mit dem *Poseidon*-USB-Stack ausgeliefert mit voller Unterstützung von UHCI, OHCI und EHCI (nicht alle Treiber unterstützen isochronen Transfer).
- NetBSD, FreeBSD und OpenBSD unterstützen UHCI, OHCI und EHCI sowie gängige Endgeräte. NetBSD war 1998 das erste freie Betriebssystem mit USB-Unterstützung.
- OS/2 Warp4 unterstützt erst über den Aufrüstpack Warp 4.51 Convenience Pack 1 (vom Dezember 2000) USB 1.1. Dieser ist kostenpflichtig. Treiber-Aktualisierungen auf USB 2.0 sind ebenfalls verfügbar.
- Palm OS unterstützt ab Version 3.2 USB als Kommunikationsplattform für HotSync, ab Palm OS 5 können (teilweise mit Zusatzprogrammen) auch Modemfunktionen über USB genutzt werden. Bestimmte PDAs (so bei Sony Clié) können mit der USB-Schnittstelle einen Massenspeicher emulieren.
- QNX unterstützt ab der Version 6 UHCI, OHCI und EHCI, mit separat erhältlichen Treibern ist USB-Support auch in QNX4 nachrüstbar. Die mitgelieferten Treiber beschränken sich auf den HID-Bootmode, einige RS232- und Ethernet-Adapter sowie Massenspeicher.

Bei Betriebssystemen ohne USB-Unterstützung kann das BIOS nach Aktivieren von „USB Legacy Support“ (engl. etwa „USB-Unterstützung für Altlasten“) in seinen Einstellungen Abhilfe schaffen, dadurch erscheinen USB-Eingabegeräte wie Mäuse und Tastaturen dem Betriebssystem gegenüber als PS/2-Geräte. Je nach BIOS wird meist genau ein USB-Laufwerk (wie USB-Stick, USB-Kartenleser, USB-Festplatte, USB-Floppy) eingebunden. USB-CD/DVD-Laufwerke werden nur dann eingebunden, wenn von ihnen gebootet wird.

8 Sicherheitsproblematik

Auf der Black Hat 2014 wies Karsten Nohl zusammen mit Jakob Lell auf die Sicherheitsrisiken von USB-Geräten hin.[39][40][41][42] Das Verfahren basiert auf der Umprogrammierung von USB-Controller-Chips, welche eine weite Verbreitung haben und zum Beispiel in USB-Sticks zum Einsatz kommen.[41] Ein wirksamer Schutz

vor einer Neubeschreibung besteht nicht, so dass sich ein harmloses USB-Gerät in vieler Hinsicht als schädliches Gerät missbrauchen lässt.[41]Mögliche Szenarien sind:

- Ein Gerät kann eine Tastatur und Befehle im Namen des angemeldeten Benutzers emulieren und Malware installieren, welche angeschlossene USB-Geräte ebenfalls infiziert.[41]
- Ein Gerät kann sich als Netzwerkkarte ausgeben, die DNS-Einstellung des Computers ändern und Datenverkehr umleiten.[41]
- Ein modifizierter USB-Stick oder eine USB-Festplatte können beim Bootvorgang einen kleinen Virus laden, welcher das Betriebssystem vor dem Booten infiziert.[41]

Eine Abwehr solcher Angriffe ist bisher nicht möglich, da Malware-Scanner keinen Zugriff auf die Firmware Version von USB-Geräten haben und eine Verhaltenserkennung schwierig ist.[41] USB-Firewalls, welche nur bestimmte Geräteklassen blocken können, existieren (noch) nicht.[41] Die sonst übliche Routine zur Beseitigung von Malware — eine Neuinstallation des Betriebssystem — schlägt hier fehl, da der USB-Stick, von dem installiert wird, bereits infiziert sein kann, ebenso wie beispielsweise eine eingebaute Webcam oder andere USB-Geräte.[41]

Im Oktober 2014 rekonstruierten die Sicherheitsforscher Adam Caudill und Brandon Wilson auf der Konferenz DerbyCon die Sicherheitslücke und stellten eine modifizierte Firmware, sowie Werkzeuge zur Verfügung.[43]

9 Kurioses

USB-Spielzeug-Raketenwerfer, der auf Befehl kleine Schaumstoffraketen abfeuert

Inzwischen sind auch ausgefallene Geräte auf den Markt gekommen, wie beispielsweise USB-Heizplatten, mit denen etwa eine Kaffeetasse über die USB-Schnittstelle warmgehalten werden kann. Daneben gibt es auch Hardware, wie USB-Lampen für Notebooks, um die Tastatur zu beleuchten, USB-Tastatur-Staubsauger, USB-Ventilatoren, Rotoren mit LED-Lichteffekten, USB-Weihnachtsbäume oder beheizbare USB-Handschuhe und USB-Pantoffeln.

Des Weiteren wird USB teilweise als standardisierte Spannungsquelle eingesetzt. So haben sich im Jahr 2009 namhafte Mobiltelefonhersteller auf Druck der EU-Kommission darauf geeinigt, Micro-USB als Standard-Gerätebuchse für den Ladekontakt einzusetzen.[44] Vereinzelt sind Hersteller anderer elektronischer Kleingeräte wie kompakten Digitalkameras nachgezogen – im Bereich der portablen Media Player (insbesondere bei den MP3-Playern) war ein Aufladen via USB-Schnittstelle schon zuvor verbreitet.

Der USB-Standard sieht vor, dass Geräte zunächst im Low Power-Mode (100 mA oder 150 mA) starten und bei höherem Strombedarf diesen erst vom Host anfordern, bevor sie den normalen Modus schalten. Das können bei USB 2.0 bis zu weiteren 4x 100 mA, bei USB 3.0 bis zu weiteren 5 × 150 mA sein. Schlägt diese Anforderung fehl, muss sich das Gerät abschalten. Die meisten der vorgenannten Spielzeuge verwenden den USB-Anschluss jedoch ungefragt nur als Spannungsquelle und verstoßen gegen den USB-Standard, indem sie ohne Erlaubnis des Hosts mehr als 100 mA Strom beziehen. Das könnte im Extremfall den USB-Anschluss des Host beschädigen oder das Energiemanagement des Rechners durcheinanderbringen, was zu instabilem Verhalten führen kann. Sparsame 2,5"-Notebook-Festplatten können meist mit 2,5 W (500 mA) an einem 2.0-USB-Port mit Adapter betrieben werden, größere 3,5"-Festplatten jedoch nicht. Weiterhin gibt es sparsame Notebook-CD/DVD/Bluray-Brenner, die am USB-Port betrieben werden können. Deren Stromaufnahme liegt jedoch insbesondere beim Brennen mit höheren Geschwindigkeiten mit teilweise permanent über 1000 mA weit außerhalb der USB-Spezifikation.

Mittlerweile gibt es Netzteile, die an einer USB-A-Buchse oder einem Kabel mit Micro-USB-B-Stecker 5 V zur Verfügung stellen. Der verfügbare Strom liegt meist bei 1000 mA (allgemein zwischen 500 und 2000 mA). Einfache Geräte stellen einfach eine Spannungsquelle dar, bessere Geräte sind so gebaut, dass sie vom zu ladenden Gerät als solch ein Netzteil erkannt werden. Dabei gilt die USB-Battery Charging Specification als Referenz (dies ist nicht mit dem Powermanagement zu verwechseln, das beim Enumeration-Prozess, beim Anschließen an einen USB-Host, stattfindet). Wenn ein Gerät nicht an einem dafür gebauten Netzteil, sondern an einem USB-Host (z. B. PC/Notebook) geladen wird, werden Befehle bzgl. des Powermanagements während der Enumeration ausgetauscht. Letzteres ist notwendig, wenn das zu ladende Gerät sich exakt an den USB-Standard hält und nur den Strom entnimmt, der ihm genehmigt wurde. Ein bekannter Vertreter ist das iPhone: Es erkennt, dass es an einem

Netzteil geladen wird, wenn bestimmte Spannungspegel an den Datenleitungen anliegen. An einem Rechner wird stattdessen über das Powermanagement verhandelt, wie viel Strom das Gerät entnehmen darf.

Ajay Bhatt wurde aus der Gruppe der Entwickler des USB-Standards besonders hervorgehoben, als er in dem Werbespot *Ajay Bhatt – The Real USB Rock Star!* des Unternehmens Intel als Rockstar porträtiert wurde.[45][46]

10 Literatur

- Hans-Joachim Kelm: USB 2.0. Franzis, Poing 2006, ISBN 3-7723-7965-6.
- Jan Axelson: USB Complete. Everything You Need to Develop Custom USB Peripherals. 4. Auflage. Lakeview Research, Madison 2009, ISBN 978-1-931448-08-6. (deutsch: USB 2.0. Handbuch für Entwickler. 3. Auflage. mitp, Heidelberg 2007 (übersetzt von Gerhard Franken), ISBN 978-3-8266-1690-7.)
- Bernhard Redemann: Steuern und Messen mit USB, Hard- und Softwareentwicklung mit dem FT232, 245 und 2232. Eigenverlag, Berlin 2006, ISBN 3-00-017884-8.

11 Weblinks

Wiktionary: USB – Bedeutungserklärungen, Wortherkunft, Synonyme, Übersetzungen

Commons: USB – Sammlung von Bildern

12 Einzelnachweise

[1] *DVI-Ausgang per USB nachrüsten.* Im: *heise online.* 5. Juni 2007.

[2] Sven Hesse: *USB 3.0 kommt 2009 mit 4,8 Gbit/s angerast.* In: *Allround-PC.com.* 21. November 2008.

[3] Oliver Ehm: *Stromversorgung für den USB-Anschluss.* Com-Magazin.de, 22. Februar 2012.

[4] *USB Voltage Drop and Droop Measurement (PDF; 184kb).* Intel Corporation, 18. November 1996.

[5] Industrie stellt einheitliches Handyladegerät vor http://derstandard.at/1296696631540/Universell-Industrie-stellt-einheitliches-Handyladegeraet-vor derstandard.at Abgerufen am 8. Februar 2011

[6] *Stecker rein! – Ein Ladegerät für alle.*

[7] Approved Class Specification Documents Battery Charging http://www.usb.org/developers/devclass_docs Engl.

[8] *Extensible Host Controller Interface (xHCI) Specification for USB 3.0.* Intel Corporation, abgerufen am 26. April 2012 (englisch).
eXtensible Host Controller Interface for Universal Serial Bus (xHCI). Revision 1.0. Intel Corporation, 21. Mai 2010, S. 468, abgerufen am 26. April 2012 (pdf, englisch).

[9] USB-Spezifikation 2.0 Seite 153

[10] *USB Class Codes. usb.org.* 17. November 2009 (englisch).

[11] USB-Spezifikation 2.0 Seite 46

[12] USB Spezifikation 2.0 Seite 55

[13] Benjamin Benz: *Pfeilschnell – Die dritte USB-Generation liefert Transferraten von 300 MByte/s.* In: *c't.* Nr. 22, 2008, S. 212.

[14] *Samsung Galaxy S II bei samsungmobile.com*

[15] Bundesnetzagentur (Hrsg.): *Allgemeinzuteilung von Frequenzen für die Nutzung durch Anwendungen geringer Leistung der Ultra-Wideband-(UWB)-Technologie.* (PDF; 94 kB) 2010.

[16] Matthias Kremp: *Kappt die Kabel – lieber noch nicht.* In: *Spiegel Online.* 18. Januar 2008.

[17] *Von USB 1.0 bis USB 3.0: Der Siegeszug des Universal Serial Bus.* In: *player.de.* 25. September 2009.

[18] *USB 3.0 ist fertig.* In: *PC Professionell.* 18. November 2008.

[19] Anika Kehrer: *Linux unterstützt USB 3.0.* In: *Linux Community.* 8. Juni 2009.

[20] *USB 3.1: Startschuss für Entwickler* auf: *heise.de.* 1. August 2013

[21] Nico Ernst: Stromzufuhr per Superspeed. Golem.de. 4. Dezember 2013. Abgerufen am 6. Januar 2014.

[22] Typ C - USB bekommt neuen verdrehsicheren Stecker (english) usb.org. 20. Februar 2014. Abgerufen am 15. März 2014.

[23] Nico Ernst: Typ C - USB bekommt neuen verdrehsicheren Stecker. Golem.de. 4. Dezember 2013. Abgerufen am 6. Dezember 2013.

[24] Nico Ernst: Der neue USB-Stecker ist fertig und geht in Produktion. Golem.de. S. 2. 12. August 2014. Abgerufen am 12. August 2014.

[25] Dominic Jahn: Neuer USB Type-C Anschluss versorgt euren 4K Monitor mit Videosignalen (german) 4kfilme.de. S. 1. 23. September 2014. Abgerufen am 11. Oktober 2014.

[26] PowerdUSB.org (englisch)

[27] *Mobile phones to adopt new, smaller USB connector.* In: *USB Implementers Forum.* 4. Januar 2007 (englisch, PDF; 128 kB).

[28] *Mini-A und Mini-AB-Verbinder zurückgezogen.* Mai 2007 (Presseerklärung des USB-IF), PDF.

[29] *Kleinere USB-Steckverbindung für Mobilgeräte.* In: *heise online.* 5. Januar 2007.

[30] OMTP Local Connectivity Recommendations. Common Charging and Local Data Connectivity. In: *omtp.org.* 8. Juni 2010.

[31] Übersicht über verschiedene Nicht-Standard-Miniatur-USB-Steckverbinder (englisch, Informationsseite eines Adapterkabelherstellers)

[32] Der neue USB-Stecker ist fertig und geht in Produktion auf Golem.de

[33] *USB 3.0 Specification.* 5.2.2 Compliant Cable Assemblies. USB Implementers Forum, Inc., 1. Mai 2011, S. 79, abgerufen am 8. November 2013 (zip/pdf (32,4 MB), englisch).

[34] http://www.thinkgeek.com/product/eecf/?srp=3 abgerufen am 1. Juli 2013(englisch)

[35] Andreas Link: *USB 3.0: Alle Infos zum neuen Technologie-Standard.* In: *PC Games Hardware.* 25. Januar 2010.

[36] Christoph Windeck: PC-Card und USB (Praxis/Hotline). In: c't. Nr. 14, 2003, S. 204.

[37] Alexander von Obert: *Kann ich eine USB-Schnittstelle nachrüsten?* In: *USB-Memory-Stick (USB-Speichermodul) FAQ – Teil 1: Hardware.* 21. Dezember 2007, abgerufen am 27. März 2008.

[38] *Linux-Kernel 2.6.31 unterstützt USB 3.0.* In: *golem.de.* 10. September 2009.

[39] *BadUSB – On Accessories that Turn Evil by Karsten Nohl + Jakob Lell.* Black Hat, 11. August 2014, abgerufen am 15. September 2014.

[40] *Black Hat USA 2014.* Abgerufen am 15. September 2014.

[41] *Turning USB peripherals into BadUSB.* Security Research Labs, abgerufen am 15. Juli 2014.

[42] Patrick Beuth: *Jedes USB-Gerät kann zur Waffe werden.* Die Zeit, 31. Juli 2014, abgerufen am 15. September 2014.

[43] Ronald Eikenberg: *BadUSB-Tools kursieren im Netz, Angriffs-Stick im Eigenbau.* 3. Oktober 2014, abgerufen am 16. Oktober 2014.

[44] *Kommission begrüßt Einigung der Industrie auf ein universelles Ladegerät für Mobiltelefone.* Presseveröffentlichung der Europäischen Union, 29. Juni 2009.

[45] *Ajay Bhatt – The Real USB Rock Star!*

[46] Mike Magee: *Intel turns USB man into rock star.* In: *TG Daily.* 7. Mai 2009.

13 Text- und Bildquellen, Autoren und Lizenzen

13.1 Text

- **Universal Serial Bus** *Quelle:* http://de.wikipedia.org/wiki/Universal%20Serial%20Bus?oldid=136226722 *Autoren:* Wst, Kurt Jansson, Ben-Zin, Fgb, Fristu, Kku, Jed, Asb, DaB., Aka, Stefan Kühn, Ulrich.fuchs, Mikue, Echoray, Gurt, Pkn, Dishayloo, GNosis, Reinhard Kraasch, Katharina, Nd, Filzstift, Matt1971, Joscha, Kaktus, Hubi, HenrikHolke, Ralf Roletschek, Hthole, Karl Gruber, Andim, Singu, Mdo, Migra, Wolfgang glock, Guillermo, TheBug, Kubieziel, GDK, Zwobot, Kai11, Ocrho, Faxel, D, Necrophorus, HaeB, Alex42, Stern, Southpark, Wurzeldrei, Karl-Henner, Berthold Werner, Jpp, HaSee, Cat, Wiegels, Stefan64, Rdb, Naddy, Alexander.stohr, G, Nocturne, Pietz, RokerHRO, Hutschi, Perrak, Niemand, Mijobe, Neitram, Feliz, Sinn, Peter200, Masr, StYxXx, Elborn, Arzach, Okatjerute, Steschke, Plenz, Martin-vogel, Mnh, Theclaw, Ot, Ishidro, 4Frankie, Asgar, Ahellwig, Solid State, Supaari, ErhardRainer, TheK, Benji, Trainspotter, Anneke Wolf, Much89, Mikenolte, Tocotronaut, Olaf2, Torsch, Unscheinbar, Cljk, PeeCee, Grille Chompa, Pascal Giessler, Srittau, Fabian Bieker, Pierre gronau, NiTenIchiRyu, Cepheiden, WiESi, Afrank99, Dickbauch, Thomas Willerich, Pcgod, Fubar, Corsair bs, Ilion, Traut, Stefan h, Uwe Gille, DasBee, Kam Solusar, LivingShadow, Adornix, 1-1111, Marsupilami, Carbenium, Frank Klemm, Cyper, Igge, MarkusHagenlocher, Nightwish79, Captaingrog, Konrad Stein, BWBot, Rhododendronbusch, Jonathan Groß, Leipnizkeks, Nightwish62, Sabata, Botteler, D235, Nicor, Polluks, Sunnyman, Michaelsy, BLueFiSH.as, BK, Martin Bahmann, ChristophK, Appaloosa, Udo T., Mr. Anderson, Ormek, Tullius, Verwüstung, Thorbjoern, A.Heidemann, Rosenzweig, Denniss, Diba, Wachhund, Kopoltra, Jergen, Herr-Vorragend, AQ, Frog23, Richarddd, Laza, Tauwasser, Jailbird, FlaBot, Baumanns, Ralf5000, Musik-chris, Red alert, RoDo, Binter, Hofres, Achim Raschka, Fxp, Burns, AndreasPrang, -jha-, Jackson, Lurchi5, Hunding, Kniekel, Blaubahn, E3c2d6ec0ca59f4588b8bb5cb621cfa6, Flominator, AchimP, Ataub2qf, Mrsurrender, RedBot, Dschen, Jodoform, Tuxo, Clemensfranz, Dapeda, O.Koslowski, Brazzy, Scooter, Zuffi, DiplomBastler, Itti, Thomas05, Gunther, Krje, Rieschl, Ath, WikiNick, Elderas, NilsSchneider, W-sky, Heurik, Bholliger, Pemu, FritzG, Marcus Cyron, JuTa, Claaser, MovGP0, Froggl, Kapege.de, Hans B., Florian Adler, Mist, Tomakos, MrBurns, LiQuidator, Dantor, Arma, WikiPimpi, Krischan111, Wilfried Elmenreich, Uwe W., Georg Slickers, Dr.G, Fkoch, W!B:, Roterraecher, Winfried Gänßler, Chobot, EinKonstanzer, STBR, Grapelli, Jackalope, Ephraim33, Gronau, Wassily, JFKCom, Suirenn, Ulm, Exxu, RobotQuistnix, M-sch, Mistmano, Smial, Bota47, Martin schulte, Tsca.bot, Andreasklug, ProloSozz, Lars Trebing, Kirschblut, YurikBot, Schaufi, GeroZ, Stefan506, Edvjacob, Savin 2005, Smurftrooper, WikiMax, Marc-André Aßbrock, Löschfix, Gromobir, MIL, Chaddy, Misery, Schmitty, Braunbaer87, Sallynase, Zac67, Video2005, DerHexer, WAH, Keymaster, Keykiller, Pizzero, McB, Micwil, Shmia, Wikipit, HBR, TRS, Timecube, Nightflyer, Quensen, Secretgardener, P-chan, PortalBot, LKD, Werner Koller, Vehrter, ThePeritus, Cointel, Fomafix, Sotel, Thogo, Chlewbot, HolgM, Dannys9, Manecke, Shadak, Pool, JoBa2282, Backsideficker, DerHerrMigo, NSX-Racer, Saemon, Hundehalter, Korinth, DHN-bot, Cottbus, Dansei, BambooBeast, Mediocrity, Stefan Knauf, Invisigoth67, Thgoiter, CedricBLN, DuMonde, Oefe, ALpHatheONE, Ernst Schwartz, Locusta, Falk2, Joschi71, Wikinger77, Wdwd, Pendulin, -Benni-, Arnulf zu Linden, Redlukas, Jasper Perky, Sixot, Church of emacs, Tönjes, Zooloo, D-Kuru, MichaelFrey, Armin P., Doctor B, WikiMM, Lofote, Roo1812, Jom, Y2kbug, TobiToaster, Till.niermann, Cancun, Spuk968, Alvanx, Thijs!bot, XenonX3, PrimeEvil, YMS, FBE2005, PsY.cHo, Sloesch, Pyxlyst, Djangothebest, Leider, Markus Bärlocher, Schwijker, SymTec ltd., Stefan Salewski, Horst Gräbner, Bernard Ladenthin, Smurf511, Tobi B., Chillvie, Scooty, D0ktorz, Berliner76, Schnahacken, GT1976, Stummi, Source gmbh, JAnDbot, Regani, Matgoth, Redhound, NobbiP, Dagobert666, YourEyesOnly, ComillaBot, Goldenbird, Sebbot, K0nze, Marcus Schätzle, J-g-s, EcceNux, WinfriedSchneider, Dem Zwickelbert sei Frau, PeterPaan, CommonsDelinker, Tobias "ToMar" Maier, Tsfla, Primus von Quack, Giftmischer, Blaufisch, Don Magnifico, Bernhard Wallisch, Chicken1303, Zollernalb, Cvf-ps, Diwas, BK-Master, Nobody perfect, Euphoriceyes, Fraidenker, Stoffel0976, SashatoBot, Complex, Ebcdic, Gunnar.offel, Reaper35, VolkovBot, Reissdorf, Waldi66, AlnoktaBOT, Sveniboy89, TXiKiBoT, AndreasFahrrad, Hagis3, Regi51, Erdbeerquetscher, Boonekamp, Ole62, Duschgeldrache2, Idioma-bot, KAI42, Landydoc, 2sd, Synthebot, Rhino2, Noogle, Mc-404, Ennimate, AlleborgoBot, OecherAlemanne, Phmovie, Carminox, Krawi, Bugert, LabFox, SieBot, Echtner, VVVBot, Toa7d6, Hypersim, DaBot, Tunc, Der kleine grüne Schornstein, Der.Traeumer, Kibert, Perfektionist, Kleinigkeiter, Engie, Qhx, Funkruf, Wsfm, Trustable, INM, Kreuzschnabel, Rotkaeppchen68, Hxhbot, Avoided, KnopfBot, Dachbewohner, Treaki, Torsten88, Alnilam, Kh555, CenterMan, Pittimann, Patchy, Björn Bornhöft, Horst2000, Se4598, DragonBot, ManfredEP, Delta82, Uvw, Sebssebi, Vanger, Alexbot, Inkowik, Bibermaus, Nirazul, DumZiBoT, Abutoum, Agash C, BodhisattvaBot, Rowland, Feierfrosch, VanRaz, Cutnyakdhien, Cm256, Martin Homuth-Rosemann, Mbutscher, Sprachpfleger, Cookiemonster, DerSchnüffler, J. H. Traven, Schotterebene, Johnny Controletti, LaaknorBot, Grip99, Marco74, Lars Beck, Unterstrichmoepunterstrich, Waterstrider, Graf Matzerath, Amirobot, MystBot, Luckas-bot, Hukukçu, Gruß Tom, Reinraum, Ptbotgourou, Null Drei Null, Nallimbot, Blootwoosch, Jotterbot, TheMightyPirate, Small Axe, Dahlmann, Garnichtsoeinfach, Benjamin Rommel, Bauruine, Scavanger667, Ersthelfer, Oollii, Xqbot, ArthurBot, GiftBot, Kixotea, Howwi, Enomil, Der Messer, Itu, Keysinger, N-regen, Grindinger, WissensDürster, Philleb, Umweltschützen, Almabot, Linear77, Popelmaus, Ketterer, RibotBOT, Nameless23, PS2801, Seelentau, Patagonier, LucienBOT, Informatica, Conan174, Farin12, BenzolBot, Jivee Blau, HRoestBot, Hilarmont, MorbZ-Bot, MatzUp, Zauguin, Timk70, Carolyn67, Corrigo, Martinlo64, Vhancer, Daniel5Ko, Alraunenstern, Helium4, Patrick Gajic, DerGraueWolf, Ripchip Bot, Jjeka, Wassertraeger, Mr. Froude, Uwe Dedering, Akkakk, EmausBot, Easy Israel, Powenz, Unsterblicher, Vorrauslöscher, Haendy-freak, Phiarc, Nierskiesel, Otterinfo, Improve, George Animal, Henri97, Hajue59, Xayax, 0815mondo, WikitanvirBot, Randolph33, ChuispastonBot, Teleutomyrmex, 9of17, In dubio pro dubio, Toru10, Trigonomie, Phry, Lucia Clemens, Iste Praetor, CherryX, Mha1993, Giebenrath, Hephaion, Paseo Maritimo, Nin-TD, Chjb, MerlIwBot, KLBot2, Master Sheep, Robby1978, Van'Dhunter, Lómelinde, Hans Haase, Deadlyhappen, Teninger, It-bill, Schecke, Henrik Haftmann, Mauerquadrant, LordOider, Dateientlinkerbot, Flo998, Ibis12, Exoport, Rmcharb, Buchexperte, Philipp97714, Roughztah, Anıl Öztaş, -revi, JackInTheBox82, Solaris3, Chayze und Anonyme: 879

13.2 Bilder

- **Datei:4port-usb-hub-bus-powered.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/b/b2/4port-usb-hub-bus-powered.jpg *Lizenz:* CC-BY-SA-3.0 *Autoren:* Transfered from de.wikipedia
  Originalkünstler: PrimeEvil at de.wikipedia
- **Datei:4port-usb-hub.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/1/1e/4port-usb-hub.jpg *Lizenz:* CC-BY-SA-3.0 *Autoren:* Transferred from de.wikipedia; transferred to Commons using CommonsHelper.
  Originalkünstler: Oder Zeichner: Pascal Giessler

  Original uploader was Pascal Giessler at de.wikipedia.

- **Datei:Commons-logo.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/4/4a/Commons-logo.svg *Lizenz:* Public domain *Autoren:* This version created by Pumbaa, using a proper partial circle and SVG geometry features. (Former versions used to be slightly warped.) *Originalkünstler:* SVG version was created by User:Grunt and cleaned up by 3247, based on the earlier PNG version, created by Reidab.
- **Datei:Disambig-dark.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/e/ea/Disambig-dark.svg *Lizenz:* CC-BY-SA-3.0 *Autoren:* Original Commons upload as Logo Begriffsklärung.png by Baumst on 2005-02-15 *Originalkünstler:* Stephan Baum
- **Datei:Missilelauncher.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/6/6d/Missilelauncher.jpg *Lizenz:* CC-BY-SA-3.0 *Autoren:* http://www.stuffgeekswant.com/2009/01/dream-cheeky-782-usb-missile-launcher-3-foam-missiles/ *Originalkünstler:* http://www.dreamcheeky.com/
- **Datei:Qsicon_Exzellent.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/4/41/Qsicon_Exzellent.svg *Lizenz:* CC-BY-SA-3.0-2.5-2.0-1.0 *Autoren:* File:Qsicon exzellent.png *Originalkünstler:* User:Niabot
- **Datei:SuperSpeed_USB.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/0/0c/SuperSpeed_USB.svg *Lizenz:* Public domain *Autoren:* http://www.usb.org *Originalkünstler:* Originally uploaded by Caspertheghost at en.wikipedia
- **Datei:USB-HighSpeed-certified-Logo.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/de/0/0b/USB-HighSpeed-certified-Logo.svg *Lizenz:* ? *Autoren:*

  nicht angegeben

  Originalkünstler:

  unbekannt
- **Datei:USB-Logo_generic.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/de/6/62/USB-Logo_generic.svg *Lizenz:* ? *Autoren:*

  nicht angegeben

  Originalkünstler:

  unbekannt
- **Datei:USB-certified-Logo.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/de/e/ef/USB-certified-Logo.svg *Lizenz:* ? *Autoren:*

  nicht angegeben

  Originalkünstler:

  unbekannt
- **Datei:USB.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/6/67/USB.svg *Lizenz:* CC-BY-SA-3.0 *Autoren:* Own painting/graphic *Originalkünstler:* Simon Eugster – Simon / ?! 19:02, 7 January 2008 (UTC)
- **Datei:USB2-PCI_Card.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/6/6b/USB2-PCI_Card.jpg *Lizenz:* CC-BY-SA-2.0 *Autoren:* Eigenes Werk *Originalkünstler:* Andreas Frank, Germany. This photograph is my work and has NOT been taken by Ricardo Moctezuma López.
- **Datei:USB2.0_CardBus_Controller.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/4/49/USB2.0_CardBus_Controller.jpg *Lizenz:* CC-BY-SA-2.0 *Autoren:* Eigenes Werk *Originalkünstler:* Afrank99 (talk)
- **Datei:USB_3.0_B_plug.png** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/d/dc/USB_3.0_B_plug.png *Lizenz:* CC-BY-SA-3.0 *Autoren:* Eigenes Werk *Originalkünstler:* Hundehalter
- **Datei:USB_3.0_Kabel_und_Stecker.png** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/e/ea/USB_3.0_Kabel_und_Stecker.png *Lizenz:* CC-BY-SA-3.0 *Autoren:* self made *Originalkünstler:* Winfried Gaenssler
- **Datei:USB_3.0_Micro_B_plug.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/7/78/USB_3.0_Micro_B_plug.svg *Lizenz:* CC-BY-3.0 *Autoren:* Own work by uploader (Ref: Nikkei Electronics 2008.10.06 Page 91) *Originalkünstler:* Tosaka
- **Datei:USB_AB_mini.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/5/56/USB_AB_mini.svg *Lizenz:* CC-BY-SA-3.0 *Autoren:* Eigenes Werk *Originalkünstler:* Appaloosa
- **Datei:USB_A_plug_size.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/c/c0/USB_A_plug_size.svg *Lizenz:* GFDL *Autoren:* Transferred from de.wikipedia; transferred to Commons by User:Wdwd using CommonsHelper.
  Originalkünstler: Martin Trautmann + PS2801 (Vektorisierung). Original uploader was PS2801 at de.wikipedia
- **Datei:USB_B_plug_size.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/9/97/USB_B_plug_size.svg *Lizenz:* GFDL *Autoren:* Transferred from de.wikipedia; transferred to Commons by User:Wdwd using CommonsHelper.
  Originalkünstler: Martin Trautmann + PS2801 (Vektorisierung). Original uploader was PS2801 at de.wikipedia
- **Datei:USB_High_Speed_on-the-go_Logo.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/de/1/18/USB_High_Speed_on-the-go_Logo.svg *Lizenz:* ? *Autoren:*

  nicht angegeben

  Originalkünstler:

  unbekannt
- **Datei:USB_Micro-A_plug_size.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/3/36/USB_Micro-A_plug_size.svg *Lizenz:* GFDL *Autoren:* Transferred from de.wikipedia; transferred to Commons by User:Wdwd using CommonsHelper.
  Originalkünstler: Martin Trautmann + PS2801 (Vektorisierung). Original uploader was PS2801 at de.wikipedia
- **Datei:USB_Micro-B_plug_size.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/3/3c/USB_Micro-B_plug_size.svg *Lizenz:* GFDL *Autoren:* Transferred from de.wikipedia; transferred to Commons by User:Wdwd using CommonsHelper.
  Originalkünstler: Martin Trautmann + PS2801 (Vektorisierung). Original uploader was PS2801 at de.wikipedia
- **Datei:USB_Mini-B_plug_size.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/8/8f/USB_Mini-B_plug_size.svg *Lizenz:* GFDL *Autoren:* Transferred from de.wikipedia; transferred to Commons by User:Wdwd using CommonsHelper.
  Originalkünstler: Martin Trautmann + PS2801 (Vektorisierung). Original uploader was PS2801 at de.wikipedia

- **Datei:USB_Notebook_Maus.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/9/9f/USB_Notebook_Maus.jpg *Lizenz:* GPL *Autoren:* Transferred from de.wikipedia
  Originalkünstler: Original uploader was PrimeEvil at de.wikipedia
- **Datei:USB_Wireless_certified_Logo.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/de/a/a1/USB_Wireless_certified_Logo.svg *Lizenz:* ? *Autoren:*
  nicht angegeben
  Originalkünstler:
  unbekannt
- **Datei:USB_on-the-go_Logo.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/de/e/ef/USB_on-the-go_Logo.svg *Lizenz:* ? *Autoren:*
  nicht angegeben
  Originalkünstler:
  unbekannt
- **Datei:Usb-Verlängerungskabel.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/b/bd/Usb-Verl%C3%A4ngerungskabel.jpg *Lizenz:* CC-BY-SA-3.0 *Autoren:* Transferred from de.wikipedia
  Originalkünstler: PrimeEvil at de.wikipedia
- **Datei:Usb-svg.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/2/22/Usb-svg.svg *Lizenz:* CC-BY-SA-2.5 *Autoren:* Eigenes Werk *Originalkünstler:* DocMiller
- **Datei:Usb_kabel_beschriftet.jpg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/a/a7/Usb_kabel_beschriftet.jpg *Lizenz:* Public domain *Autoren:* Transferred from de.wikipedia; transferred to Commons by User:Wdwd using CommonsHelper.
  Originalkünstler: Denise Nepraunig. Original uploader was Cm256 at de.wikipedia
- **Datei:Wiktfavicon_en.svg** *Quelle:* http://upload.wikimedia.org/wikipedia/commons/c/c3/Wiktfavicon_en.svg *Lizenz:* CC-BY-SA-3.0 *Autoren:* ? *Originalkünstler:* ?

13.3 Inhaltslizenz

- Creative Commons Attribution-Share Alike 3.0

Teil VI
3-PLD

15 PLD - Programmable Logic Device

Eine programmierbare logische Schaltung, häufig auch in deutschsprachiger Fachliteratur als Programmable Logic Device oder kurz PLD bezeichnet, ist ein elektronisches Bauelement für integrierte Schaltkreise. Anders als logische Gatter, die eine feste Funktion vorgegeben haben, erhalten PLDs erst nach der Herstellung ihre Funktion durch die entsprechende Programmierung (Konfiguration). PLD ist der Oberbegriff für Logik-IC-Technologien wie PAL, PLA, GAL, CPLD und für neuere Technologien wie FPGAs. Einfache programmierbare logische Schaltungen bestehen meist aus einem Array aus UND-Verknüpfungen gefolgt von einem Array aus ODER-Verknüpfungen.

Programmable Read-Only Memory (PROM)
Ein PROM stellt ein festes UND-Array mit einem programmierbaren ODER-Array bzw. eine Lookup-Tabelle (LUT) dar.

Programmable Array Logic (PAL) bzw. Generic Array Logic (GAL)
Ein PAL stellt ein programmierbares UND-Array mit einem festen ODER-Array dar. Ein GAL ist im Gegensatz zu einem PAL wiederbeschreibbar.

PLA (programmable logic array)
Bei einem PLA sind sowohl das UND-Array als auch das ODER-Array programmierbar. PLAs wurden meistens eingesetzt, um sogenannte Glue Logic zu ersetzen. Inzwischen werden sie kaum noch eingesetzt und sind durch folgende PLDs ersetzt worden.

Complex Programmable Logic Device (CPLD)
Ein CPLD besteht aus Blöcken, die ein PLA, Ein- und Ausgangsblöcke sowie eine programmierbare Rückkopplung enthalten. Diese Blöcke können untereinander verbunden werden. In der Regel ist für jeden I/O-Pin auch ein Flipflop enthalten.

Field Programmable Gate Array (FPGA)
Ein FPGA besteht ähnlich wie ein CPLD aus untereinander vernetzten Blöcken, jedoch sind diese komplexer. Ein Block besteht hier aus Flip-Flops und LUTs. Auch die Möglichkeiten, diese Blöcke untereinander zu verbinden, sind gegenüber dem CPLD stark erweitert. Ein FPGA enthält oft auch fertige Funktionsblöcke wie RAM, PLLs oder ganze CPU-Kerne.

CPLDs
wie FPGAs bieten außerdem oft programmierbare I/O-Zellen, die es erlauben, verschiedene Signalschnittstellen (z. B. TTL, PCI oder LVDS) an den Baustein anzuschließen.

Unterscheidung nach Programmierbarkeit

15.0.1 Maskenprogrammiert

Hier wird die Konfiguration schon bei der Produktion des Bauteils festgelegt (siehe Gate-Array). Sollen FPGAs in großen Stückzahlen eingesetzt werden, können diese bei einigen Herstellern maskenprogrammiert geordert werden. Dies spart zusätzliche Produktionsschritte und die zur Konfiguration notwendigen externen Bauteile.

One Time Programmable (OTP)
Hier gibt es die Programmierung durch Durchbrennen von Verbindungen (Fusible-link) oder das Schaffen von Verbindungen bei der Antifuse-Technologie.

Erasable Programmable Read Only Memory (EPROM)
Die EPROM-Programmierung wurde meist nur bei PLAs eingesetzt.

Electrically Erasable Programmable Read-Only Memory (EEPROM) oder Flash
GALs sind EEPROM programmiert und können deshalb im Gegensatz zu PALs wiederverwendet werden. Die Konfiguration von CPLDs findet in der Regel über Flash-Speicher statt. Die Programmierung über EEPROM bzw. Flash hat den Vorteil, dass das Bauteil sofort nach dem Einschalten fertig konfiguriert zu Verfügung steht.

SRAM basiert
Die Konfiguration von FPGAs ist in der Regel SRAM-basiert. Diese muss nach dem Einschalten des FPGAs erst in den Baustein geladen werden, entweder durch ein Konfigurations-PROM oder einen angeschlossenen Mikroprozessor. Ein FPGA kann sogar während des Betriebes ganz oder teilweise neu programmiert werden, z. B. um einen laufenden Verarbeitungsalgorithmus zu ändern. Ein Anwendungsgebiet dafür ist das **Reconfigurable Computing.**

(aus https://de.wikipedia.org/wiki/Programmierbare_logische_Schaltung 31Aug20)

s. Böhmer 'Elemente der angewandten Elektronik', 15.Aufl., Kapitel ROMs, PROMs und PLDs. (ROM, PLD und PAL S. 268–269, Aufbau und Programmierung von GALs im Anhang S. 418–419,

Early programmable logic

In 1969, Motorola offered the XC157, a mask-programmed gate array with 12 gates and 30 uncommitted input/output pins.

In 1970, Texas Instruments developed a mask-programmable IC based on the IBM read-only associative memory or ROAM. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. The TMS2000 had up to 17 inputs and 18 outputs with 8 JK flip flop for memory. TI coined the term Programmable Logic Array (PLA) for this device.

In 1971, General Electric Company (GE) was developing a programmable logic device based on the new Programmable Read-Only Memory (PROM) technology. This experimental device improved on IBM's ROAM

by allowing multilevel logic. Intel had just introduced the floating-gate UV erasable PROM so the researcher at GE incorporated that technology. The GE device was the first erasable PLD ever developed, predating the Altera EPLD by over a decade. GE obtained several early patents on programmable logic devices.

In 1973 National Semiconductor introduced a mask-programmable PLA device (DM7575) with 14 inputs and 8 outputs with no memory registers. This was more popular than the TI part but cost of making the metal mask limited its use. The device is significant because it was the basis for the field programmable logic array produced by Signetics in 1975, the 82S100. (Intersil actually beat Signetics to market but poor yield doomed their part.)

In 1974 GE entered into an agreement with Monolithic Memories (MMI) to develop a mask–programmable logic device incorporating the GE innovations. The device was named the 'Programmable Associative Logic Array' or PALA. The MMI 5760 was completed in 1976 and could implement multilevel or sequential circuits of over 100 gates. The device was supported by a GE design environment where Boolean equations would be converted to mask patterns for configuring the device. The part was never brought to market.

In 1970, Texas Instruments developed a mask-programmable IC based on the IBM read-only associative memory or ROAM. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. The TMS2000 had up to 17 inputs and 18 outputs with 8 JK flip-flops for memory. TI coined the term programmable logic array for this device.

A programmable logic array (PLA)
has a programmable AND gate array, which links to a programmable OR gate array, which can then be conditionally complemented to produce an output. A PLA is similar to a ROM concept, however a PLA does not provide full decoding of a variable and does not generate all the minterms as in a ROM.

15.0.2 PAL

devices have arrays of transistor cells arranged in a "fixed-OR, programmable-AND"plane used to implement ßum-of-products"binary logic equations for each of the outputs in terms of the inputs and either synchronous or asynchronous feedback from the outputs. MMI introduced a breakthrough device in 1978, the programmable array logic or PAL. The architecture was simpler than that of Signetics FPLA because it omitted the programmable OR array. This made the parts faster, smaller and cheaper. They were available in 20 pin 300 mil DIP packages while the FPLAs came in 28 pin 600 mil packages. The PAL Handbook demystified the design process. The PALASM design software (PAL assembler) converted the engineers' Boolean equations into the fuse pattern required to program the part. The PAL devices were soon second-sourced by National Semiconductor, Texas Instruments and AMD. After MMI succeeded with the 20-pin PAL parts, AMD introduced the 24-pin 22V10 PAL with additional features. After buying out MMI (1987), AMD spun off a consolidated operation as Vantis, and that business was acquired by Lattice Semiconductor in 1999.

15.0.3 GALs

Lattice GAL 16V8 and 20V8
An improvement on the PAL was the generic array logic device, or GAL, invented by Lattice Semiconductor in 1985. This device has the same logical properties as the PAL but can be erased and reprogrammed. The GAL is very useful in the prototyping stage of a design, when any bugs in the logic can be corrected by reprogramming. GALs are programmed and reprogrammed using a PAL programmer, or by using the in-circuit programming technique on supporting chips. Lattice GALs combine CMOS and electrically erasable (E2) floating gate technology for a high-speed, low-power logic device. A similar device called a PEEL (programmable electrically erasable logic) was introduced by the International CMOS Technology (ICT) corporation.

15.0.4 CPLDs

PALs and GALs are available only in small sizes, equivalent to a few hundred logic gates. For bigger logic circuits, complex PLDs or CPLDs can be used. These contain the equivalent of several PALs linked by programmable interconnections, all in one integrated circuit. CPLDs can replace thousands, or even hundreds of thousands, of logic gates. Some CPLDs are programmed using a PAL programmer, but this method becomes inconvenient for devices with hundreds of pins. A second method of programming is to solder the device to its printed circuit board, then feed it with a serial data stream from a personal computer. The CPLD contains a circuit that decodes the data stream and configures the CPLD to perform its specified logic function. Some manufacturers (including Altera and Microsemi) use JTAG to program CPLDs in-circuit from .JAM files.

FPGAs
While PALs were being developed into GALs and CPLDs, a separate stream of development was happening. This type of device is based on gate array technology and is called the field-programmable gate array (FPGA). Early examples of FPGAs are the 82s100 array, and 82S105 sequencer, by Signetics, introduced in the late 1970s. The 82S100 was an array of AND terms. The 82S105 also had flip flop functions. (Remark: 82S100 and similar ICs from Signetics have PLA-Structure, AND-Plane + OR-Plane) FPGAs use a grid of logic gates, and once stored, the data doesn't change, similar to that of an ordinary gate array. The term "field-programmable"means the device is programmed by the customer, not the manufacturer. FPGAs are usually programmed after being soldered down to the circuit board, in a manner similar to that of larger CPLDs. In most larger FPGAs, the configuration is volatile and must be re-loaded into the device whenever power is applied or different functionality is required. Configuration is typically stored in a configuration PROM or EEPROM. EEPROM versions may be in-system programmable (typically via JTAG).

The difference between FPGAs and CPLDs is that FPGAs are internally based on look-up tables (LUTs)

whereas CPLDs form the logic functions with sea-of-gates (e.g. sum of products). CPLDs are meant for simpler designs while FPGAs are meant for more complex designs. In general, CPLDs are a good choice for wide combinational logic applications, whereas FPGAs are more suitable for large state machines such as microprocessors.

Other variants

These are microprocessor circuits that contain some fixed functions and other functions that can be altered by code running on the processor. Designing self-altering systems requires engineers to learn new methods, and that new software tools be developed. PLDs are being sold now that contain a microprocessor with a fixed function (the so-called core) surrounded by programmable logic. These devices let designers concentrate on adding new features to designs without having to worry about making the microprocessor work. Also, the fixed-function microprocessor takes less space on the chip than a part of the programmable gate array implementing the same processor, leaving more space for the programmable gate array to contain the designer's specialized circuits.

How PLDs retain their configuration

A PLD is a combination of a logic device and a memory device. The memory is used to store the pattern that was given to the chip during programming. Most of the methods for storing data in an integrated circuit have been adapted for use in PLDs. These include:

-Silicon antifuses
-SRAM
-EPROM or EEPROM memory cells
-Flash memory

Silicon antifuses are connections that are made by applying a voltage across a modified area of silicon inside the chip. They are called antifuses because they work in the opposite way to normal fuses, which begin life as connections until they are broken by an electric current.

SRAM, or static RAM, is a volatile type of memory, meaning that its contents are lost each time the power is switched off. SRAM-based PLDs therefore have to be programmed every time the circuit is switched on. This is usually done automatically by another part of the circuit.

An EPROM memory cell is a MOSFET (metal-oxide semiconductor field-effect transistor, or MOS transistor) that can be switched on by trapping an electric charge permanently on its gate electrode. This is done by a PAL programmer. The charge remains for many years and can only be removed by exposing the chip to strong ultraviolet light in a device called an EPROM eraser.

Flash memory is non-volatile, retaining its contents even when the power is switched off. It is stored on floating-gate MOSFET memory cells, and can be erased and reprogrammed as required. This makes it useful in PLDs that may be reprogrammed frequently, such as PLDs used in prototypes. Flash memory is a kind of EEPROM that holds information using trapped electric charges similar to EPROM. Consequently, Flash memory can hold information for years, but possibly not as many years as EPROM.

As of 2005, most CPLDs are electrically programmable and erasable, and non-volatile. This is because they are too small to justify the inconvenience of programming internal SRAM cells every time they start up, and EPROM cells are more expensive due to their ceramic package with a quartz window.

PLD programming languages

Many PAL programming devices accept input in a standard file format, commonly referred to as 'JEDEC files'. They are analogous to software compilers. The languages used as source code for logic compilers are called hardware description languages, or HDLs.

PALASM, ABEL and CUPL are frequently used for low-complexity devices, while Verilog and VHDL are popular higher-level description languages for more complex devices. The more limited ABEL is often used for historical reasons, but for new designs VHDL is more popular, even for low-complexity designs.

PLD programming devices

A device programmer is used to transfer the boolean logic pattern into the programmable device. In the early days of programmable logic, every PLD manufacturer also produced a specialized device programmer for its family of logic devices. Later, universal device programmers came onto the market that supported several logic device families from different manufacturers. Today's device programmers usually can program common PLDs (mostly PAL/GAL equivalents) from all existing manufacturers. Common file formats used to store the boolean logic pattern (fuses) are JEDEC, Altera POF (programmable object file), or Xilinx BITstream.

(aus https://en.wikipedia.org/wiki/Programmable_logic_device 31Aug20)

16 FPGA - Field Programmable Gate Array

Field Programmable Gate Array

Ein Field Programmable Gate Array (englisch, FPGA) oder programmierbares Logikgatter ist ein integrierter Schaltkreis (IC) der Digitaltechnik, in welchen eine logische Schaltung geladen werden kann. Die englische Bezeichnung kann übersetzt werden als im Feld (also vor Ort, beim Kunden) programmierbare (Logik-)Gatter-Anordnung. Anders als bei der Programmierung von Computern, Microcontrollern oder Steuerungen bezieht sich hier der Begriff Programmierung nicht nur auf die Vorgabe zeitlicher Abläufe, sondern vor allem auch auf die Definition der gewünschten Schaltungsstruktur. Diese wird mittels einer Hardwarebeschreibungssprache formuliert und von einer Erzeugersoftware in eine Konfigurationsdatei übersetzt, welche vorgibt, wie die physischen Elemente im FPGA ver-

schaltet werden sollen. Man spricht daher auch von der Konfiguration des FPGA. Ohne diese hat der Baustein keine Funktion. FPGAs stellen eine Weiterentwicklung der PLDs dar und kommen in vielen Gebieten der digitalen Elektronik zum Einsatz.

Anwendung

Durch Konfiguration der intern vorhandenen Elemente können in einem FPGA verschiedene Schaltungen und Funktionen realisiert werden. Diese reichen von Schaltungen geringer Komplexität, wie z. B. ein einfacher Synchronzähler oder Interfaces für Digitalbausteine, bis hin zu hochkomplexen Schaltungen wie Speicher-Controller und vollständige Mikroprozessoren. FPGAs werden in allen Bereichen der Digitaltechnik eingesetzt, vor allem aber dort, wo es auf schnelle Signalverarbeitung und flexible Änderung der Schaltung ankommt, um beispielsweise nachträgliche Verbesserungen an den implementierten Funktionen vornehmen zu können, ohne dabei direkt die Hardware ändern zu müssen. Ein großes Anwendungsgebiet ist die Erstellung von Prototypen in der ASIC-Entwicklung zum vorherigen Test sowie auch der Bereich Maintenance, in dem es darum geht, Elektroniken für alte, nicht mehr lieferbare digitale Bausteine oder Microcontroller vorzuhalten. Mit der Einführung der FPGAs wurden kompakte, anwenderspezifische Schaltungen in geringen Stückzahlen ermöglicht. Heute gestatten sie die preiswerte und flexible Fertigung komplexer Systeme wie Mobilfunk-Basisstationen als Alternative zur teureren Auftragsfertigung durch Halbleiterhersteller. Neben den FPGAs existieren auch FPAAs (Field Programmable Analog Array), die nicht nur digitale, sondern vor allem analoge Funktionsblöcke enthalten, die vom Anwender programmiert und verschaltet werden können. Es handelt sich dabei in erster Linie um Filter und HF-Bauelemente.

Aufbau und Struktur

- Logikblock eines FPGA mit 4-bit-LUT und optionalem Ausgangs-Flipflop
- Schaltmatrix als Verbindungsstruktur zwischen Logikblöcken

Die wesensbestimmende Grundstruktur eines FPGA ist ein Feld (engl. Array) aus Basisblöcken mit jeweils einer einfachen programmierbaren Lookup-Tabelle (LUT) und einem 1-Bit-Register (Flipflop). Die LUTs können, je nach Anzahl der verfügbaren Eingänge, jede beliebige n-stellige Binärfunktion realisieren. Die Programmierung der gewünschten Funktion erfolgt durch die Hinterlegung der definierenden Wahrheitstabelle in den SRAM-Zellen der LUT, die Funktionsberechnung durch das Auslesen der durch die Eingänge bestimmten Speicheradresse. Lange Zeit waren LUT-Strukturen mit 4 binären Eingängen üblich. Neuere FPGAs gehen zur Verringerung des Aufwandes an LUT-zu-LUT-Verbindungen zur Realisierung von Funktionen mit mehr Eingängen auf LUTs mit bis zu 6 Eingängen über. Neben den LUTs ist auf dem FPGA auch die Verschaltung der Komponenten in großen Freiheitsgraden konfigurierbar. Multiplexer-Strukturen in den Basisblöcken ermöglichen häufig sehr schnelle lokale Signalpfade, zur Einbindung oder Umgehung des Flipflops, zur Rückkopplung von dessen Ausgang, zur Verbindung von Nachbarblöcken und ähnlichem. Für die ferneren Verbindungen liegt zwischen den Basisblöcken ein Gitter aus immensen Busstrukturen, an das Ein- und Ausgänge angeschlossen werden können. Weitere programmierbare Schaltkomponenten in den Kreuzungspunkten des Gitters erlauben die Signalverteilung über den gesamten Chip.

Weitere, oft vorzufindende Elemente von FPGA sind:
Eingangs-/Ausgangs-Blöcke (engl. IO-Blocks oder IOB) dienen der Kommunikation mit der Außenwelt. Über sie werden die Anschlüsse des FPGA mit der Schaltmatrix verbunden. Auch diese Blöcke können an die jeweilige Anwendung angepasst werden, z. B. kann die Ausgangsspannung an den jeweiligen Standard angepasst werden (TTL/CMOS usw.). Ein oder mehrere Taktgeneratoren erzeugen ausgehend von an den Eingängen zur Verfügung gestellten Takten alle für die Anwendung benötigten internen Takte. Diese können gegenüber den Eingangstakten in der Phase verschoben sein und besitzen eine von dem jeweiligen Eingangstakt abgeleitete Frequenz. Taktverstärker sowie eine entsprechende Verschaltung sorgen dafür, dass jeder verwendete Takt überall im Chip synchron zur Verfügung steht. Moderne FPGAs besitzen typischerweise eine oder mehrere Phase-locked loops (PLL), mit denen sich rational gebrochene Taktunterteilungen/-vervielfachungen erzeugen lassen. Den gleichen Zweck erfüllen Delay-locked loops (DLL) und digital frequency synthesizer (DFS), die manche FPGA-Hersteller gegenüber der PLL bevorzugen. In komplexen FPGAs sind zusätzliche fest verdrahtete Funktionen enthalten, wie z. B. Speicherblöcke (sog. Block RAM), die sich in vielfältiger Weise nutzen lassen. Für Aufgaben der digitalen Signalverarbeitung wie bei digitalen Filtern sind in manchen FPGAs auch Multiplizierereinheiten enthalten. Diese erlauben es, besonders schnell zwei Binärzahlen zu multiplizieren, ohne dafür Logikzellen zu benötigen. FPGAs, die im Bereich von System-on-a-Chip Anwendung (SoC) finden, besitzen eine Reihe meist komplexer Hard-Cores, um ein komplettes System aufzunehmen. Hard-Cores sind feste und unveränderliche Schaltungen von meist komplexen Funktionsblöcken, wie beispielsweise Mikrocontroller oder Ethernetschnittstellen. Komplexe Hard-Cores belegen sehr viel weniger Chipfläche als es die gleiche Funktion, realisiert mit Logikblöcken, benötigen würde und sind typischerweise drei- bis viermal schneller als die frei konfigurierbare Logik der FPGA. Dafür sind diese Schaltungsteile nicht so flexibel und können in ihrer Funktion auch nicht mehr verändert werden. Hersteller wie Xilinx bieten mittlerweile auch auf SRAM basierende FPGAs an, die bereits über im Chipgehäuse untergebrachten Flash-Speicher zur Konfiguration verfügen und keinen externen Flash-Speicher mehr benötigen. Man nennt solche integrierten Schaltungen mit mehreren Chips in einem Gehäuse auch Multi-Die. Die Lade- bzw. Startzeiten des FPGA bleiben im Vergleich zu externem Speicher in etwa gleich, sind allerdings speziell vom Hersteller optimiert. Ein weiterer Vorteil ist der Schutz vor unrechtmäßigen Kopien durch das Auslesen des extern befindlichen Speichers. Derzeit und auf absehbare Zeit ist es technologisch nicht möglich, die SRAM-basierten Schalterzellen in einem FPGA direkt

wie bei den viel einfacheren CPLDs durch Flash- bzw. EEPROM-Zellen zu ersetzen.

Unterschiede zu CPLDs

Oft werden FPGAs mit den digitalen und ebenfalls rekonfigurierbaren CPLD-Bausteinen (Complex Programmable Logic Devices) gleichgesetzt bzw. verglichen. Die wesentlichen Unterschiede zwischen FPGAs und CPLDs sind:

- CPLD gehört noch zur Gruppe der PAL(Programmable Array Logic)/PLA(Programmable Logic Array)/PLD(Programmable Logic Device)/GAL(Generic Array Logic), hat aber einen komplizierteren Innenaufbau im IC als PAL und GAL. Die noch neueren FPGAs gehören hingegen zu einer anderen Technologie[3]. CPLDs weisen im Vergleich zu FPGAs eine wesentlich einfachere Struktur auf. Sie besitzen kein feinmaschiges Array (Feld) von Logikblöcken und Flipflops, sondern nur eine konfigurierbare Schaltmatrix, die verschiedene Eingangssignale zu verschiedenen Ausgangssignalen verbinden kann. Die Signale können dabei durch logische Operationen wie AND/OR verknüpft werden. Dadurch haben CPLDs eine vom jeweiligen Design unabhängige, konstante Signaldurchlaufzeit. Durch entsprechende Designmethoden kann auch bei FPGAs eine definierte maximale Durchlaufzeit erreicht werden (engl. timing constraints).
- CPLDs weisen vergleichsweise wenige Flipflops auf. Vor allem längere Schieberegister, Zähler, Zustandsspeicher und ähnliche Schaltungen, die viele Flipflops benötigen, sind in CPLDs nur ineffizient zu realisieren.
- CPLDs besitzen, da jeder IO-Pin ein Flipflop besitzt, meist sehr viele IO-Pins, die in vielen Anwendungen nur teilweise verwendet werden. In Anwendungen, in denen nur vergleichsweise einfache digitale Schaltungen, sogenannte glue logic, mit einem hohen Bedarf an IO-Pins Verwendung findet, sind CPLDs meist die bessere Wahl.
- CPLDs können durch ihre einfache Struktur technologisch mit Flash-Zellen in den Schaltmatrizen arbeiten. Damit ist ein CPLD sofort nach dem Einschalten (engl. Power-Up) betriebsbereit, während rekonfigurierbare FPGAs mit SRAM-basierenden Zellen erst einen Ladezyklus für die Konfiguration durchlaufen müssen. Von einigen Herstellern gibt es aber seit längerem auch FPGAs in Flash-Technik. Seit Ende der 1990er Jahre ist bei den CPLDs eine Annäherung an die FPGA-SRAM-Technologie zu beobachten. Heute arbeiten die meisten CPLDs intern SRAM-basiert. Bei einigen CPLD-Familien besteht inzwischen das logische Grundelement aus dem FPGA-typischen LUT4-Flipflop-Verbund (Altera MAX II-Serie).
- CPLDs weisen durch den einfacheren Aufbau und die geringere Größe auch einen wesentlich geringeren Stromverbrauch auf.

Konfiguration

Die Programmierung der Logikelemente kann je nach FPGA unterschiedlich gelöst werden. Man kann zwischen Methoden unterscheiden, die es ermöglichen, den FPGA mehrmals zu programmieren, und Methoden, die nur eine einmalige Programmierung zulassen. Bei den mehrmals programmierbaren FPGAs wird die Konfiguration in Speicherzellen (z. B. SRAM, EPROM, EEPROM, Flash) gespeichert. Bei den einmalig programmierbaren FPGAs werden die physikalischen Eigenschaften der Verbindungswege permanent geändert (Antifuse-Technologie). Diese Technologie bietet im Feld eine größere Sicherheit gegen äußere Störungen. Die Konfiguration der LUTs und der Verbindungsstrukturen bei SRAM-basierten FPGAs erfolgt typischerweise einmal vor jedem Einsatz, wodurch der FPGA auf eine konkrete Funktion festgelegt wird. Das ist notwendig, da der FPGA durch Abschalten der Betriebsspannung seine Konfiguration wieder verliert. Im Einsatz steht dem FPGA deshalb meist eine Art EPROM zur Seite, das die Konfiguration vorhält und dessen Inhalt selbst auch aktualisierbar ist. Der FPGA kann sich selbst aus diesem nichtflüchtigen Speicher beladen. Alternativ kann die Konfiguration auch aktiv durch einen Microcontroller oder einen anderen FPGA erfolgen, der die Konfiguration in den FPGA schreibt. In der Regel behält der FPGA seine Funktion dann bis zum Abschalten bei. Mehr und mehr wird aber auch eine bereichsweise Umprogrammierung von einzelnen FPGA-Bereichen im laufenden Betrieb unterstützt, um den Platz im FPGA effektiver zu nutzen und eigentlich festverdrahtete Funktionen zu flexibilisieren. Der Vorgang des Konfigurierens des FPGAs wird oft auch als Programmieren bezeichnet, was jedoch eine begriffliche Überschneidung mit dem Entwurf bedingt.

Entwurfsschritte und -werkzeuge

Oftmals wird bei der Entwicklung von FPGA-Schaltungen von Programmierung gesprochen. Der Begriff ist in diesem Kontext aber anders zu verstehen, als es von der Erstellung von Software für Prozessoren her bekannt ist. Im Rahmen des Entwicklungsprozesses erfolgt zunächst ein Schaltungsentwurf, gefolgt von einer Prüfung der entstandenen Hardwarebeschreibung mittels Simulationswerkzeugen und dann eine Implementierung (Place and Route) sowie gfs laufzeitbasierende Simulation. Erst danach kann die implementierte Schaltung am realen FPGA erprobt werden.

Schaltungsentwurf

Erstellt wird die Konfiguration eines FPGA entweder grafisch mittels eines Schaltplans (engl. schematic) oder textuell mit einer Hardwarebeschreibungssprache. Dies sind primär VHDL oder Verilog, welche die gesamte Funktion der Schaltung in Form von Strukturen und Abläufen beschreibt. Ein sogenanntes Synthesewerkzeug führt diese Beschreibung wie ein Programm aus und erstellt dann in mehreren Schritten für einen gewünschten Baustein eine spezifische Netzliste unter Nutzung der in diesem Baustein verfügbaren Ressourcen. Die im FPGA erforderlichen Ablaufsteuerungen wiederum lassen sich durch endliche Automaten darstellen. Der hardwarebeschreibende Code wird dann in einem Zwischenschritt automatisch erzeugt. Darüber hinaus lassen sich mit grafischen Programmiersystemen wie LabVIEW oder Matlab / Simulink oder dem kostenfreien Logiflash ebenfalls Schaltungsmodule für ein FPGA automatisch erstellen. In den letzten Jahren versuchten mehrere Projekte, Hardwareimplementierungen für ASICs, FPGAs und CPLDs mit der Programmiersprache C (HardwareC, HandelC, BachC) zu beschreiben. Aktuelle Ansätze bauen direkt auf den weit verbreiteten Standardsprachen ANSI C bzw. C++ oder Python (mit MyHDL) auf. Für SystemC existieren keine Synthesetools, der praktische Nutzen für konkrete FPGA-Entwicklungen liegt bei

der abstrakten Verhaltensmodellierung und deutlich beschleunigten Systemsimulationen, weshalb es dort zum weitverbreiteten Industriestandard avanciert ist. Es gibt auch High-Level-Synthese-Werkzeuge, um aus Hochsprachen (C/C++, MATLAB, Java, Python, UML) einen Entwurf auf Registertransferebene (VHDL, Verilog) zu erzeugen. Beispiele sind Catapult C Synthesis von Mentor Graphics, CoDeveloper von Impulse Accelerated Technologies, Cynthesizer von Forte Design Systems oder das oben erwähnte freie MyHDL. Herstellerspezifische Sprachen wie Altera-HDL (AHDL) oder auch die kaum noch verwendete Hardwarebeschreibungssprache ABEL wurden ebenso genutzt wie UDL/I (Japan). Zur Integration eingebetteter Systeme in FPGAs gibt es Werkzeuge, die eine Konstruktion auf Funktionsblockebene anbieten, z. B. Xilinx EDK (Embedded Development Kit). Funktionsblöcke wie FIFOs, Prozessoren, serielle Schnittstellen, Ethernet-MAC-Layer, RAM-Controller, Parallel-IO etc. werden vom Hersteller zur Verfügung gestellt. Diese Funktionseinheiten, IP-Core genannt, liegen manchmal als Quellcode, oder meist als verschlüsselte Netzliste vor und sind in der Regel parametrierbar (z. B. Baudrate bei seriellen asynchronen Schnittstellen oder Fifo-Tiefe oder Breite der Parallelschnittstelle). Diese werden über Busse mit anderen Funktionseinheiten verbunden.

Simulation

Nach der Beschreibung während des Entwurfsflusses folgen weitere Schritte wie die funktionale Simulation, Synthese. Ein bekanntes Werkzeug für die Simulation ist ModelSim®. Für FPGAs wird beim Schaltungsentwurf ein synchrones Schaltungsdesign empfohlen, wenngleich nicht zwingend erforderlich. Das bedeutet: An allen Flipflops in einer sogenannten Takt-Domäne (engl. clock domain) liegt der gleiche Takt. Gesteuert wird die Datenübernahme in ein FPGA-Flipflop am besten nur über die zusätzlich vorhandenen Clock Enable-Eingänge und nicht über geteilte Taktsignale (engl. gated clocks). Das vermeidet schwer handhabbare Laufzeiteffekte. Manche FPGAs bieten spezielle Umschalter an, die das garantiert störungsfreie (glitchfreie) Wechseln zwischen verschiedenen Taktquellen im Betrieb erlauben.

Implementierung und Test

Ausgehend von einer verifizierten Logikschaltung erfolgt dann ein Umsetzen der Netzliste für das konkrete FPGA, wobei auch externe Funktionsblöcke, die von Drittanbietern angeboten werden und bisher nur als black box existierten, eingefügt werden. Ebenso können bei diesem Schritt Teilschaltungen, die zum Testen des FPGAs gedacht sind, wie integrierte Logic Analyzer, hinzugefügt werden. Unter Anwendung weiterer Randbedingungen erfolgt dann in letztlich das Umsetzen auf ein Programmierfile zum Beladen des FPGAs, aus dem wiederum ein weiteres Programmierfile erzeugt werden kann, das zum Beladen des Flash-Speichers dient.

Anwendungsbeispiele

FPGAs haben seit ihren Anfängen ihren Anwendungsbereich von der klassischen „Glue-Logic“, also der reinen Verbindungslogik zwischen verschiedenen digitalen Bausteinen, zunehmend erweitert und werden heute auch bei mittleren Stückzahlen für die Realisierung komplexer digitaler Schaltungen bis hin zu kompletten digitalen Systemen eingesetzt. Durch die Rekonfigurierbarkeit von FPGAs direkt beim Endanwender besteht darüber hinaus der wesentliche Vorteil, auf aktuelle technische Entwicklungen reagieren zu können und die digitalen Schaltungen durch Updates anpassen zu können, ohne direkt die zugrundeliegende Hardware der FPGA-Chips verändern zu müssen.

FPGAs werden beispielsweise zur Echtzeit-Verarbeitung von einfachen bis komplexen Algorithmen genutzt, zur digitalen Signalverarbeitung im Rahmen von digitalen Filtern oder zur schnellen Fourier-Transformation. Aber auch Protokoll-Implementierungen wie Teile des Ethernet-MAC-Layers, die Kodierung von digitalen Videosignalen, die Verschlüsselung von Daten und Fehlerkorrekturverfahren sind Anwendungsgebiete. Besonders in Bereichen, in denen Algorithmen bzw. Protokolle einer schnellen Weiterentwicklung unterliegen, ist die Verwendung rekonfigurierbarer FPGAs statt ASICs angebracht. Die Vorteile sind schnelle Marktreife, die Möglichkeit nachfolgender Fehlerbehebungen, Anpassung an neue Entwicklungen. Für einige Klassen von Rechenproblemen sind auch FPGA-basierte Parallelcomputer sehr geeignet. Das wahrscheinlich bekannteste Beispiel sind FPGA-Rechner zum Brechen kryptographischer Verfahren, wie dem Data Encryption Standard (DES). Der aus 120 FPGAs bestehende Parallelrechner COPACOBANA ist ein solcher Parallelcomputer zum Codebrechen. Die inzwischen erreichbare Anzahl von Logikblöcken erlaubt die Integration mehrerer eingebetteter Computersysteme in einen einzigen FPGA-Baustein inklusive CPU(s), Bussystem(en), RAM, ROM, RAM-Controller, Peripherie-Controller etc. Solche kompletten Systeme werden als System-on-a-Chip (SoC) bezeichnet. Auf Grund ihrer Rekonfigurierbarkeit bilden die SRAM- und Flash-basierten FPGAs die Grundlage für rekonfigurierbare Computer.

Digitale Speicheroszilloskope werden oft mit FPGAs realisiert, da deren Stückzahlen meist zu gering sind, um einen ASIC für diesen Anwendungsfall zu entwerfen. Schnelle digitale Speicheroszilloskope verwenden pro Kanal mehrere A/D-Wandler parallel, welche das zu messende Signal phasenverschoben abtasten. Das erfordert ein sehr hohes Maß an paralleler Datenverarbeitung und -speicherung, wofür FPGAs gut geeignet sind. Oszilloskope auf FPGA-Basis können beispielsweise auf sehr kurze Impulse unterhalb der Abtastrate der A/D-Wandler triggern oder diese zählen. Ein Vorteil bei der Verwendung von FPGAs liegt generell auch darin, dass anders als bei DSPs verhältnismäßig geringe Entwicklungskosten bei Neuentwürfen entstehen, falls einer der verwendeten ICs nicht mehr erhältlich ist und dessen Funktion in einen existenten Baustein integriert werden kann. FPGAs werden sehr häufig auch als Entwicklungsplattform für den digitalen Teil von ASICs verwendet, um die Funktion zu verifizieren. Das ist nötig, da aufgrund der Komplexität heutiger Schaltungen alleinige Simulationen zu zeitaufwändig wären. Viele Veröffentlichungen aus unterschiedlichsten Anwendungsgebieten berichten

über Migration einer Anwendung von Software nach Configware mit Beschleunigungsfaktoren von einer bis zu vier Größenordnungen. Deshalb finden FPGAs Eingang beim Reconfigurable Computing und beim Ersatz von Microcontrollern. Ein besonders auf FPGAs zugeschnittener Tochterkartenstandard ist die FPGA Mezzanine Card.

Mikroprozessoren

FPGAs bilden je nach Konfiguration beliebige Anordnungen digitaler Schaltungsfunktionen ab und bieten damit grundsätzlich die Möglichkeit, Informationen vollkommen parallel zu verarbeiten. So können die anfallenden Datenflüsse in Bandbreite und Informationstiefe optimal einander angepasst werden. Schnell zu erfassende Signale werden dabei oft voll parallel mit Kopien identischer Schaltungsblöcke, langsamer auftretende Signale vermehrt zyklisch mit einer einzigen Schaltung und damit platzsparend verarbeitet. Externe Prozessoren können dagegen mit wenig Hardware sehr komplexe und verschachtelte Programme sequentiell abarbeiten. Bei einem FPGA müsste für jede Operation ein eigenes Stück Hardware synthetisiert werden, wobei sich die eingeschränkte Anzahl an Logikgattern begrenzend auswirkt, zudem ist der Aufbau einer vergleichbar flexiblen Struktur äußerst schwierig und zeitaufwändig. Daher bedient man sich bei komplexeren Aufgaben einer sogenannten soft core CPU, die in das FPGA-Design eingebunden wird. Diese gleicht den externen CPUs und stellt eine standardisierte Struktur bereit, die in klassischer Weise in C programmiert werden. Heutige FPGAs sind teilweise so leistungsfähig, dass man eine Vielzahl an 8-, 16- oder 32-Bit-CPU-Kernen integrieren kann. Allerdings beanspruchen komplexere CPU-Kerne je nach Konfiguration recht viele Logikressourcen, was sich neben den Kosten in relativ geringer Verarbeitungsleistung verglichen mit Standardprozessoren niederschlägt. Daher gibt es inzwischen FPGAs, die einen oder mehrere hardware-basierte CPU-Kerne enthalten. Nach ersten Anläufen mit ARM-9 (Altera) und PowerPC405 (Xilinx) geht der Trend seit 2010 in Richtung ARM-Cortex-Architektur. Bei Altera und Xilinx haben sich sogenannte SoC-FPGAs mit Dual-Core-Cortex-A9 und fest integrierter Peripherie etabliert. Xilinx bietet in der High-End-Klasse mit der UltraScale-Familie Bausteine an, die zwei CPU-Cluster (QuadCore Cortex-A53 und DualCore Cortex-R5) enthalten. Microsemi hingegen integriert die einfachere Cortex-M3-Architektur in seine SmartFusion-Familie. Im Gegensatz zu FPGAs sind Single-Core-Prozessoren reine endliche Zustandsautomaten, die mit einer festgelegten Hardware auskommen müssen und ihr Programm sequentiell abarbeiten, woraus sich auch wesentliche Unterschiede in der Gestaltung bei der Implementierung von Algorithmen ergeben. Mikroprozessoren besitzen häufig eine fest vorgeschriebene Peripherie. Das ist Vor- und Nachteil zugleich. Einerseits muss die Peripherie nicht noch extra erzeugt werden, jedoch lässt sich diese in einem FPGA problemgerecht anpassen. Moderne Prozessoren mit SIMD-Befehlen wie zum Beispiel der Intel i7-3930K verarbeiten bis zu 96 Gleitkommabefehle parallel (48 Additionen + 48 Multiplikationen), der erreichbare Durchsatz liegt bei 3,2 GHz bei theoretisch etwa 300 GFlops, von denen 250 GFlops bei praxisrelevanten Aufgaben erreichbar sind. Aktuelle FPGAs können zehntausende Festkommaadditionen (bis 48 bit) und tausende Festkommamultiplikationen (bis 18×25 bit) bei Taktfrequenzen von 500 MHz zur gleichen Zeit ausführen. Damit sind für Verarbeitungsaufgaben, die sich mit Festkommaarithmetik bearbeiten lassen, um den Faktor 2 bis 3 höhere Verarbeitungsleistung bei deutlich geringerer Leistungsaufnahme möglich.

Vor- und Nachteile im Vergleich zu ASICs

Vorteile

- Deutlich geringere Entwicklungskosten (im Gegensatz zu ASICs werden keine Masken mit sehr hohen Fixkosten benötigt)
- bei Prototypen und Kleinserien sehr kostengünstig
- Kürzere Implementierungszeiten
- Einfach korrigier- und erweiterbar (rekonfigurierbar)
- Geringeres Designrisiko, da es nicht lange vor der Hardwareauslieferung fertig sein muss
- flexible IO-Ports und Standards, d. h. bei technischen Änderungen der Umgebung anpassbar

Nachteile

- Ab mittleren Stückzahlen höherer Stückpreis (als ASICs)
- Geringere Taktraten (aktuell verfügbar bis 1,5 GHz, typisch werden 20–500 MHz realisiert; digitale ASICs bieten >3 GHz)
- Geringere Logikdichte (ca. 10-facher Flächenbedarf gegenüber ASIC gleicher Technologie)
- Geringere mögliche Komplexität der programmierbaren Logik
- Deutlich höherer Leistungsbedarf für gleiche Menge an Logik bzw. Funktionen
- Höhere Empfindlichkeit gegenüber Teilchenstrahlung und elektromagnetischen Wellen (da über RAM-Zellen und nicht durch Hartverdrahtung programmiert)
- Geringere Flexibilität, was Ausstattung z. B. mit eingebettetem Speicher oder analogen Elementen angeht, aber auch bei IO-Buffern
- Der kürzere Designzyklus und die Möglichkeit, sehr spät noch Fehler korrigieren zu können, verleiten dazu, im Vorfeld weniger funktionale Tests durchzuführen.
- SRAM-basierte FPGAs (das sind z. B. alle von den Marktführern Xilinx und Altera angebotenen) müssen bei jeder Spannungsunterbrechung neu geladen werden. Das bedeutet, dass die Funktionalität nicht direkt nach dem Einschalten zur Verfügung steht. Das Laden kann – je nach eingesetzter Technik – bis zu einigen Sekunden dauern. Handelt es sich nicht um spezielle FPGAs mit integriertem Flashspeicher, sind dazu zusätzliche, externe Komponenten notwendig, z. B. ein herstellerspezifisches EEPROM oder Flash-Speicher, das die Konfiguration enthält oder ein Mikrocontroller mit zusätzlichem Flash-Speicher, der den Ladevorgang durchführt.

Hersteller von FPGAs

- Xilinx -seit 1984 FPGA-Produzent- Marktführer, bietet das am weitesten ausgebaute SOPC-System
- Intel PSG (früher: Altera) – Anbieter eines Migrationspfads vom FPGA zu strukturierten ASICs
- Lattice – Anbieter eines freien 32 Bit-Open-Source-SoftCore-Prozessors sowie von GAL-Technik. Hat SiliconBlue aufgekauft (Stromsparende FPGAs der iCE Familie)

- Microchip (früher: Atmel & Actel bzw. Microsemi) - FPGAs, auch mit zusätzlich integriertem RAM und AVR-Mikrocontroller
- NanoXplore – Strahlungsresistente FPGAs
- QuickLogic -seit 1988– Anbieter von stromsparenden FPGAs
- Aeroflex – Strahlungsresistente FPGAs
- Achronix Semiconductor – Sehr schnelle FPGAs (bis 1,5 GHz) in 22 nm
- Abound Logic – Stromsparende FPGAs mit hoher Logikdichte
- GOWIN Semiconductor – FPGAs mit geringer Zahl an Logikblöcken, nichtflüchtige und SRAM-basierte, SoC mit Bluetooth

(aus https://de.wikipedia.org/wiki/Field_Programmable_Gate_Array#Aufbau_und_Struktur 31Aug20)

17 LUT - Lookup Table

Lookup-Tabellen (LUT) bzw. Umsetzungstabellen werden in der Informatik und in der Digitaltechnik verwendet, um Informationen statisch zu definieren und diese zur Laufzeit des Programms – zur Vermeidung aufwändiger Berechnungen oder hohen Speicherverbrauchs – zu benutzen.[1]

Grundstruktur

In einer Tabelle werden für bestimmte Konstellationen vorberechnete Ergebnisse oder andere Informationen definiert. Die einzelnen Einträge der LUT werden entweder über eine Spalte Kurzcode (der als Suchbegriff verwendet wird) oder über ihre Position (Eintrag 01-nn gilt für Sachverhalt 1-nn) identifiziert. Jeder Eintrag enthält die vordefinierte Information, bei Bedarf auch weitere für den Eintrag geltende Attribute. In der Ausführung von Programmen, d. h. zur Verwendung der LUT-Inhalte, wird auf die einzelnen Einträge der LUT durch Referenzierung (über im Programm gebildete oder verfügbare Schlüssel) zugegriffen.

Nutzen und Zweck

Komplexe Berechnungen zur Programmlaufzeit lassen sich durch eine in der Regel schnellere Wertsuche ersetzen. Speicherplatz lässt sich einsparen, weil in den eigentlichen Datenbeständen (mit einer hohen Anzahl von Einträgen) nur ein Kurzcode geführt und die zugehörige Langbezeichnung aus der Tabelle verwendet wird. Auch lässt sich der Erfassungsaufwand sowie die Fehlerwahrscheinlichkeit durch Eingabe von Kurzcodes (anstatt langer Bezeichnungen) bzw. durch Verwendung von Auswahlboxen (mit Vorbelegung möglicher Eingaben) minimieren. Die 'ausgelagerten' Informationen können bei Bedarf geändert werden (z. B. Langbezeichnungen), ohne die Änderung in den eigentlichen Daten selbst vornehmen zu müssen.

Speicherung, Erzeugung, Verarbeitung

Zur Speicherung und Verarbeitung von Lookup-Tabellen sind folgende Varianten üblich:

Der Tabelleninhalt wird im Programm selbst (in speicherinternen Datenstrukturen) gespeichert und zur Verarbeitung verwendet. Der Tabelleninhalt wird in externen Datenbeständen (z. B. in einer Datenbanktabelle oder einer Datei) gespeichert. Zur Verarbeitung wird auf diese Daten entweder jeweils direkt zugegriffen oder sie werden bei Programmstart in den internen Speicher des Programms geladen und wie unter 1. verarbeitet.

Zur Erzeugung der LUT-Daten kommen mehrere Verfahrensweisen infrage:

Die LUT-Inhalte werden im Programm statisch definiert; nur zu Variante (1) möglich. Änderungen bedingen eine neue Programmversion. Die LUT-Inhalte werden automatisch (z. B. bei Programmanfang oder in einem Vorprogramm) ermittelt und temporär gespeichert. Ist das erzeugende und das verwendende Programm identisch, so kann wie bei (1) intern gespeichert werden. Werden die LUT-Daten von anderen Programmen, evtl. mehreren, benutzt, so wird wie bei (2) extern gespeichert. Die LUT-Inhalte werden mit einer eigenen Anwendung oder mithilfe von Standardwerkzeugen zur Datenpflege erzeugt und geändert. Hierbei ist nur die externe Speicherung gem. (2) üblich.

Externe Lookup-Tabellen definieren sich lediglich über die Art ihrer Verwendung (look-up = "nachschlagen"); bezüglich der Speicherungstechnik unterscheiden sie sich in keiner Weise von anderen Daten.

Anwendung für vorberechnete Ergebnisse Prinzip

Die Werte einer Funktion werden vorab ermittelt und im Speicher als Tabelle abgelegt. Damit gleicht das LUT-Verfahren der Benutzung einer Tabelle aus der Vor-Taschenrechner-Ära, wie bei Zinstabellen, im Tafelwerk und manchen Rechenschiebern.

Die LUT wird als eine Datenstruktur, meist ein (assoziatives) Array, das komplizierte Laufzeitberechnungen durch einen einfachen indizierten Zugriff auf die Datenstruktur ersetzt, realisiert. Dies führt zu einem signifikanten Geschwindigkeitsgewinn, sofern die benötigten Speicherzugriffe schneller sind als die normale Berechnung.

Da die Zugriffe auf den Index der Lookup-Tabelle mit einem geringerwertigeren Datentyp durchgeführt werden können als die in der Tabelle enthaltenen Werte, kann die Methode auch zur Einsparung von Speicherplatz verwendet werden.

Ein klassisches Beispiel ist eine trigonometrische Tabelle. Die Berechnung des Sinus etwa kann sehr lange dauern und ist bei jedem Aufruf der Funktion wiederholt nötig. Um das zu vermeiden, werden zu Beginn einige Werte des Sinus berechnet und in einer Tabelle gespeichert, zum Beispiel für jede ganze Gradzahl. Später, wenn ein konkreter Sinus berechnet werden soll, rundet die Funktion die gewünschte Gradzahl und liest den Sinuswert aus der Tabelle.

Pre-Intervallabbildung und Post-Interpolation

Zu beachten ist weiterhin, dass z. B. ein reelles Argument (bzw. eine Real-Zahl mit einigen Nachkommastellen) erst auf einen natürlichen (Integer-)Index abge-

bildet werden muss, um als Schlüssel für eine Speicherstelle Verwendung zu finden. Dazu muss, wenn beispielsweise für eine periodische Funktion nur Werte aus der ersten Periode um 0 herum in der LUT vorhanden sind, das Argument zunächst in das Periodenintervall abgebildet („reelles Modulo“) und danach gehasht (auf eine Speicherstelle abgebildet) werden.

Um die Genauigkeit der Berechnung zu verbessern, kann auch ein Interpolations-Algorithmus verwendet werden. Dabei wird versucht, durch mehrere benachbarte Einträge aus der Tabelle (im obigen Beispiel die darüber und darunter liegende ganze Gradzahl) und einige weitere Berechnungen den Wert genauer abzuschätzen. Das benötigt zwar etwas mehr Zeit, kann die Genauigkeit aber enorm verbessern. Die Methode kann auch dazu verwendet werden, die Lookup-Tabelle bei gleicher Genauigkeit zu verkleinern. Nachteile Bei der Benutzung von Lookup-Tabellen ist zu beachten, dass sie auch langsamer als die direkte Berechnung sein können, wenn die Berechnung relativ simpel ist. Das liegt nicht nur an langsamen Speicherzugriffen, sondern auch an einem erhöhten Speicherbedarf und einer Beeinträchtigung des Caches. Dies wird immer wichtiger, da Mikroprozessoren zunehmend schneller als Speicherchips werden. Optimierungen wie die beispielhaft angeführten Sinus-Tabellen sind mit modernen Prozessorgenerationen oft unnötig oder sogar kontraproduktiv.

Anwendung in Integrierten Schaltungen → Hauptartikel: „Aufbau und Struktur“ im Artikel Field Programmable Gate Array Beispielhafter Logikblock eines FPGAs, mit LUT und Flipflop

In der digitalen Schaltungstechnik werden im Gegensatz zur Programmierung auch sehr einfache Funktionen wie logische Gleichungen (AND, OR, XOR) durch eine LUT ersetzt. Eine Tabelle ist leichter anzupassen als eine Transistorschaltung, daher wird die LUT in der programmierbaren Logik, insbesondere FPGAs und bei der Herstellung von ICs nach Kundenwunsch (ASIC), implementiert.

Bei FPGAs wird die Tabelle in einem kleinen SRAM-Feld gespeichert. So kann mit einem Speicher von 16×1 Bit jede logische Funktion mit 4 Eingängen realisiert und durch Programmierung geändert werden. Die Anzahl der LUT-Eingänge hängt von der FPGA-Architektur ab. In ASICs wird die LUT u. a. als (Masken-)ROM realisiert. Anstatt einen IC komplett für den Auftraggeber maßzufertigen, werden insbesondere bei Gate-Arrays variable Grundschaltungen als LUTs vorgefertigt; nur wenige Fertigungsschritte (Metallisierung) werden speziell für den Kunden ausgeführt.

Eine weitere LUT-Schaltung basiert auf einem 2n-nach-1-Multiplexer mit n Steuereingängen und 2n Speicherstellen. Auch wurden PROM-Speicher zur Realisierung einer 8-Bit-ALU verwendet.

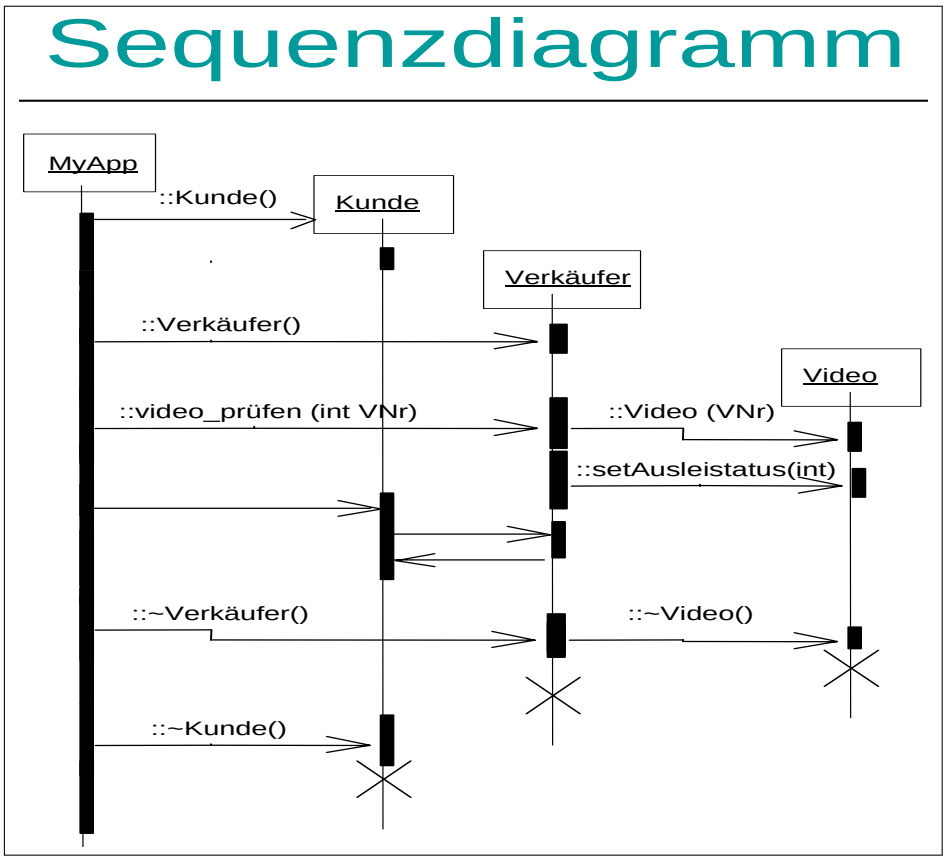
(aus https://de.wikipedia.org/wiki/Lookup-Tabelle 31Aug20)

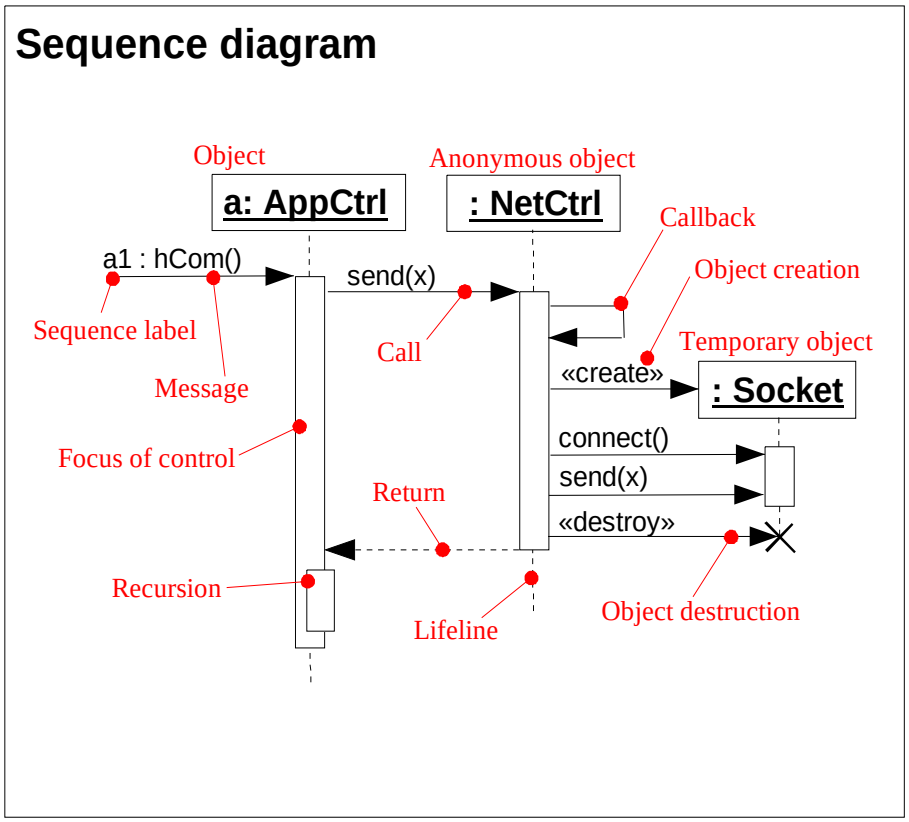
s. ’HDL’ Kap.?? S.??
s. ’VHDL’ Kap.13 S.114

Teil VII

Programmierung

18 UML - sequence DGm





19 UML unified modeling language

Die klassischen Entwurfs- und Darstellungsmethoden wie

Flussdiagramm (flowchart),

Struktogramm,

HIPO Design

usw. bieten kaum bis keine Instrumente für

a) Objektorientierung

b) Nebenläufigkeit (concurrency)

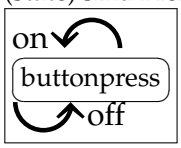
an.

Dies hat man mit dem
Klassendiagramm
(class diagram)
und dem den Petri-Netzen verwandten
Aktivitätsdiagramm
(activity diagram)
gelöst und dabei gleich weitere Aspekte wie Zeitverhalten (sequence diagram), Zustände (state chart), Verteilung (deployment) oder Aufgabenstellung (problem definition) und Grobdesign (draft) (=use cases) integriert.

Die Bezeichnung *'unified'* bezieht sich auf diese Zusammenwürfelung.

[...] UML ist heute die dominierende Sprache für die Softwaresystem-Modellierung. Der erste Kontakt zu UML besteht häufig darin, dass Diagramme in UML im Rahmen von Softwareprojekten zu erstellen, zu verstehen oder zu beurteilen sind [...] (aus `https://de.wikipedia.org/w/index.php?title=Unified_Modeling_Language&oldid=152646977`*, 23Apr16)*

AD activity diagram (UML)
Vorgänge (activities) in Ovalen, Zustände (state) sind Pfeile.



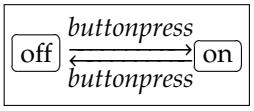
verwandt zum *Flussdiagramm = flow chart = work flow*

CompDG component diagram (UML)

DeplDG deployment diagram (UML)

CD class diagram, Klassendiagramm (UML)
(heisst in UML eigentlich *static structure diagram*

StD statechart diagram, Zustandsdiagramm (UML)
Vorgänge (activities) sind Pfeile, Zustände (state) in Ovalen.



verwandt zum *bubble-diagram*

SD/SqD sequence diagram (UML)
(des mit di Objekt-Lebenslinien)

UCD use-case diagram (UML)
(des mit di Strichmandln)

19.1 Strukturdiagramme

Klassendiagramm
UseCaseDiagramm
ObjektDiagramm
PaketDiagramm
VerteilungsDiagramm
KompositionsstrukturDG
KomponentenDiagramm

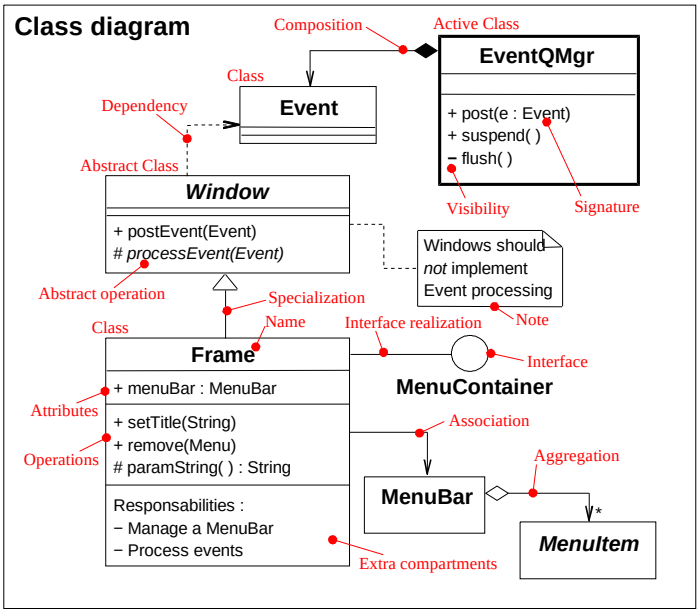
19.2 Verhaltensmodellierung

AktivitätsDiagramm
SequenzDiagramm
Zustandsautomat, StateDiagram
InteraktionsübersichtsDiagramm
Timing-Diagramm
KommunikationsDiagramm

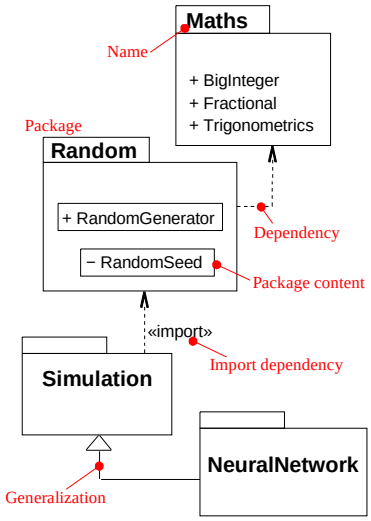
Eine übersichtliche Zusammenstellung in flyerartiger Aufmachung fand sich unter `http://www.oose.de/uml` 'Notationsübersicht'

UML Quick Reference Card

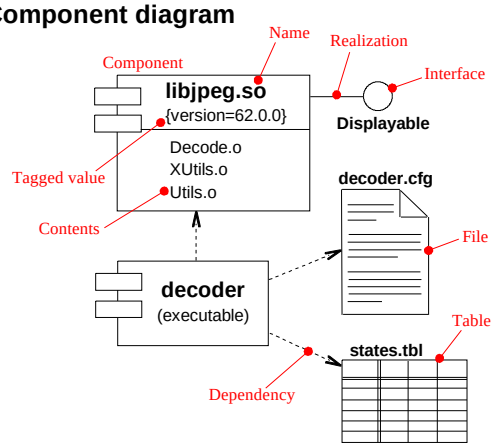
Class diagram



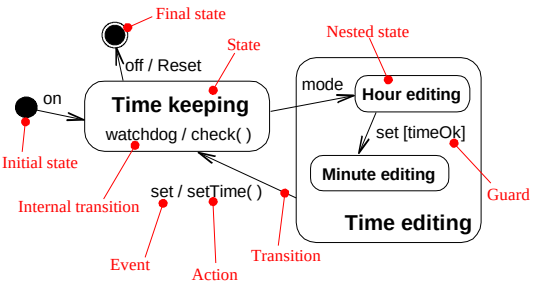
Collaboration diagram



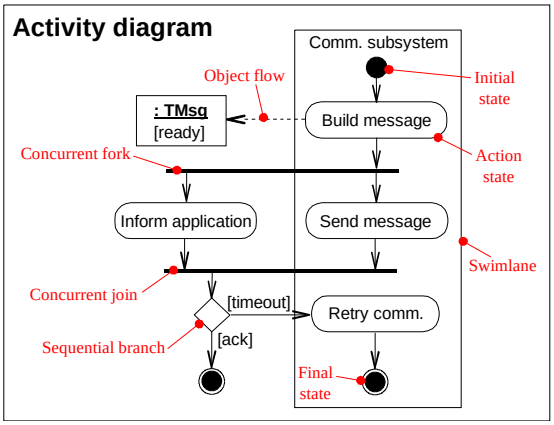
Component diagram



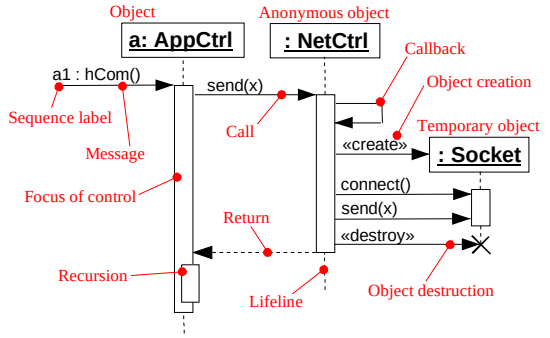
State diagram



Activity diagram



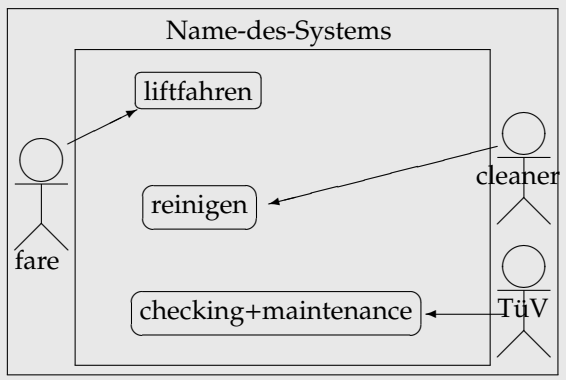
Sequence diagram



19.3 Use Case Diagram

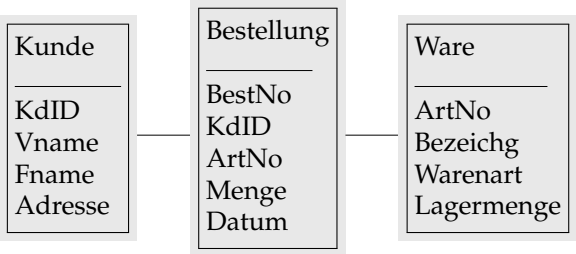
'za wos brauchi des' - Diagramm Zeigt die Anwendungen des zu entwickelnden Systems.

1 wer
verwendet
2 das System
3 wozu

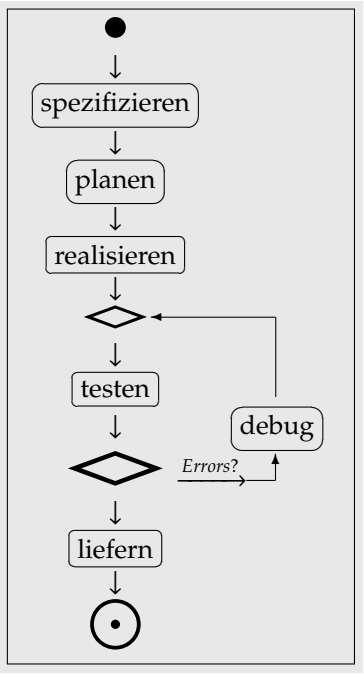


19.4 Class Diagram

ähnelt (logischerweise) sehr den **ER-Diagrammen** aus der Datenbanktechnik



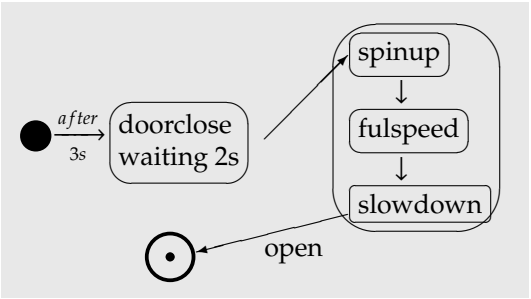
19.5 Activity Diagram



ist dem **Flussdiagramm** nachempfunden.
Es gibt die Raute ◇ als 'if' Symbol und die Kastln sein rund:
Aktion

19.6 State Chart Diagram

Das ist ein dem Bubble-Diagramm aus der Schaltwerkentwurfstechnik entsprechender Zustandsgraph

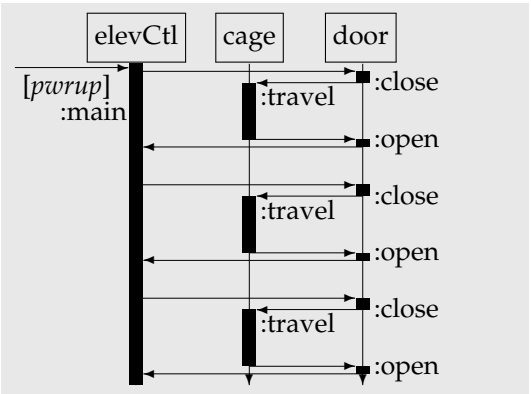


UML statechart diagram

19.7 Sequence Diagram

- welche ':'Methode
- in welchem Objekt
  wird
- unter welcher [Bedingung]
- wann (→)

aktiviert:



UML sequence diagram

20 AVR'ing in Assembler

Linux-Kommandos (rPi):

```
df -h
pwd
cd /home/pi
cd Documents
cd ~/Downloads
ls -tr
ls -1tr
ls -ltr
du -s *
which gcc
ls -l /usr/bin/gcc
gcc  TN13pArm-VeA1.c
ls -l a.out
./a.out
gcc  TN13pArm-VeA1.c  -o TN13p.arm
./TN13p.arm  x  xxx
exit
sudo poweroff
```

20.1 das kleine leere AVR TN13 Programm (ASM)

Listing 4: kleine leere AVR Prog

```
;das (kleine) "LEERE" Atmel AVR AT Tiny13 Programm
;  (fuer Linux-Assembler "gavrasm")
;---------------------------------------------------------------------
.device ATtiny13        ;Gerd's AVR-Assembler taugt fyr viele Controller
.def    Rcnt    =R16    ;Schleifenzaehler Rcnt...alias/"nick"-Name fyr "Counter"
.def    Rtemp   =R17    ;"temporary" fuer kurzes Zwischenspeichern
 ;
        .CSEG           ;CodeSpeicher (ProgrammFlash) Segment
        .ORG    0x0000  ;Speicher-Adress-Festlegung "Origin"
        RJMP    INIT            ;wird beim Start und RESET aktiviert
         RETI   ;EXT_INT0       ;ISR ist "leer" (nur "RETI")
         RETI   ;PINCHG
         RETI   ;T0OVF
         RETI   ;EE_RDY
         RETI   ;ANACOMP
         RETI   ;TMR0_CMPA
         RETI   ;TMR0_CMPB
         RETI   ;WDR
         RETI   ;ADC
INIT:
        CLI                             ;disable all Interrupts
        LDI     Rtemp,Low(RAMEND)       ;"stack" einrichten:
        OUT     SPL,Rtemp
        LDI     Rtemp,0b00000011        ;PB0,PB1 ... output
        OUT     DDRB,Rtemp
        OUT     PORTB,Rtemp             ;Outputs auf "Hi" schalten
        SEI
;
MAIN:           ;das HAUPTPROGRAMM: Nur "Lebenszeichen" ausgeben:
        IN      Rtemp,PORTB     ;PINB...InputRegister von Port-B
        CBI     PORTB,1         ;Bit Nr.1 auf "Low"
        SBRS    Rtemp,1         ;Ueberspringe, wenn das Bit==1 war
        SBI     PORTB,1         ;Bit Nr.1 auf "High"
        RJMP    MAIN            ;RJMP == "Relative JuMP" == "goto"
.EXIT                           ;Ende der Assemblierung
```

20.2 dessen 'listing file' .LST

Listing 5: kleine leere AVR Listing

```
gavrasm Gerd's AVR assembler version 2.2 (C)2008 by DG4FAC
------------------------------------------------------------
Source file: TN13-B06kl.ASM
Hex file:    TN13-B06kl.hex
Eeprom file: TN13-B06kl.eep
Compiled:    09.03.2012, 11:22:56
Pass:        2
     1: ;das (kleine) "LEERE" Atmel AVR AT Tiny13 Programm
     2: ;  (fuer Linux-Assembler "gavrasm")
     3: ;---------------------------------------------------------------------------
     4: .device ATtiny13         ;Gerd's AVR-Assembler taugt fyr viele Controller
     5: .def    Rcnt    =R16     ;Schleifenzaehler Rcnt...alias/"nick"-Name fyr "Counter"
     6: .def    Rtemp   =R17     ;"temporary" fuer kurzes Zwischenspeichern
     7:  ;
     8:         .CSEG            ;CodeSpeicher (ProgrammFlash) Segment
     9:         .ORG    0x0000   ;Speicher-Adress-Festlegung "Origin"
    10: 000000   C009   RJMP     INIT             ;wird beim Start und RESET aktiviert
    11: 000001   9518   RETI     ;EXT_INT0        ;ISR ist "leer" (nur "RETI")
    12: 000002   9518   RETI     ;PINCHG
    13: 000003   9518   RETI     ;T0OVF
    14: 000004   9518   RETI     ;EE_RDY
    15: 000005   9518   RETI     ;ANACOMP
    16: 000006   9518   RETI     ;TMR0_CMPA
    17: 000007   9518   RETI     ;TMR0_CMPB
    18: 000008   9518   RETI     ;WDR
    19: 000009   9518   RETI     ;ADC
    20: INIT:
    21: 00000A   94F8   CLI                                ;disable all Interrupts
    22: 00000B   E91F   LDI      Rtemp,Low(RAMEND)         ;"stack" einrichten:
    23: 00000C   BF1D   OUT      SPL,Rtemp
    24: 00000D   E013   LDI      Rtemp,0b00000011          ;PB0,PB1 ... output
    25: 00000E   BB17   OUT      DDRB,Rtemp
    26: 00000F   BB18   OUT      PORTB,Rtemp               ;Outputs auf "Hi" schalten
    27: 000010   9478   SEI
    28: ;
    29: MAIN:            ;das HAUPTPROGRAMM: Nur "Lebenszeichen" ausgeben:
    30: 000011   B318   IN       Rtemp,PORTB       ;PINB...InputRegister von Port-B
    31: 000012   98C1   CBI      PORTB,1           ;Bit Nr.1 auf "Low"
    32: 000013   FF11   SBRS     Rtemp,1           ;Ueberspringe, wenn das Bit==1 war
    33: 000014   9AC1   SBI      PORTB,1           ;Bit Nr.1 auf "High"
    34: 000015   CFFB   RJMP     MAIN              ;RJMP == "Relative JuMP" == "goto"
    35: .EXIT                              ;Ende der Assemblierung
 -> Warning 001: 1 symbol(s) defined, but not used!

Program              :       22 words.
Constants            :        0 words.
Total program memory:        22 words.
Eeprom space         :        0 bytes.
Data segment         :        0 bytes.
Compilation completed, no errors.
Compilation endet 09.03.2012, 11:22:56
```

20.3 AM16 interrupt vector table

Listing 6: AM16 Interrupt Vectors

```
 0 RESET
 1 INT0          external interrupt
 2 INT1          external interrupt
 3 TIMER2 COMP   counter 2 compare match
 4 TIMER2 OVF    timer 2 overflow
 5 TIMER1 CAPT   counter 1 capture
 6 TIMER1 COMPA counter 1 compare match A
 7 TIMER1 COMPB counter 1 compare match B
 8 TIMER1 OVF    timer 1 overflow
 9 TIMER0 OVF    timer 0 overflow
10 SPI, STC      serial transfer complete
11 USART, RXC    USART RX complete
12 USART, UDRE   USART data reg empty
13 USART, TXC    USART TX complete
14 ADC           AD conversion complete
15 EE_RDY                EEPROM ready (write)
16 ANA_COMP      analog comparator match
17 TWI           TWI/IIC
18 INT2          external interrupt
19 TIMER0 COMP   counter 0 compare atch
20 SPM_RDY               store program memory ready
----------------
=21 vectors
```

20.4 AM32 interrupt vector table (nicht ident!)

Listing 7: AM32 Interrupt Vectors

```
 0 RESET
 1 INT0          external interrupt
 2 INT1          external interrupt
 3 INT2          external interrupt
 4 TIMER2 COMP   counter 2 compare match
 5 TIMER2 OVF    timer 2 overflow
 6 TIMER1 CAPT   counter 1 capture
 7 TIMER1 COMPA counter 1 compare match A
 8 TIMER1 COMPB counter 1 compare match B
 9 TIMER1 OVF    timer 1 overflow
10 TIMER0 OVF    timer 0 overflow
11 SPI, STC      serial transfer complete
12 USART, RXC    USART RX complete
13 USART, UDRE   USART data reg empty
14 USART, TXC    USART TX complete
15 ADC           AD conversion complete
16 EE_RDY               EEPROM ready (write)
17 ANA_COMP      analog comparator match
18 TWI           TWI/IIC
19 TIMER0 COMP   counter 0 compare atch
20 SPM_RDY              store program memory ready
----------------
=21 vectors
```

20.5 AVR 8bit instruction set

Arithmetic and Logic Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
ADD	Rd,Rr	Add without Carry	Rd= Rd+Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	Add with Carry	Rd= Rd+Rr+C	Z,C,N,V,H,S	1
SUB	Rd,Rr	Subtract without Carry	Rd= Rd-Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	Subtract Immediate	Rd= Rd-K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	Subtract with Carry	Rd= Rd-Rr-C	Z,C,N,V,H,S	1
SBCI	Rd,K8	Subtract with Carry Immedtiate	Rd=Rd-K8-C	Z,C,N,V,H,S	1
AND	Rd,Rr	Logical AND	Rd= Rd·Rr	Z,N,V,S	1
ANDI	Rd,K8	Logical AND with Immediate	Rd=Rd·K8	Z,N,V,S	1
OR	Rd,Rr	Logical OR	Rd= Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Logical OR with Immediate	Rd=Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Logical Exclusive OR	Rd=Rd⊕Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd=$FF-Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd=$00-Rd	Z,C,N,V,H,S	1
SBR	Rd,K8	Set Bit(s) in Register	Rd=Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Clear Bit(s) in Register	Rd=Rd·($FF-K8)	Z,C,N,V,S	1
INC	Rd	Increment Register	Rd=Rd+1	Z,N,V,S	1
DEC	Rd	Decrement Register	Rd=Rd-1	Z,N,V,S	1
TST	Rd	Test for Zero or Negative	Rd=Rd·Rd	Z,C,N,V,S	1
CLR	Rd	Clear Register	Rd=0	Z,C,N,V,S	1
SER	Rd	Set Register	Rd=$FF	—	1
ADIW	Rdl,K6	Add Immediate to Word	Rdh:Rdl= Rdh:Rdl+K6	Z,C,N,V,S	2
SBIW	Rdl,K6	Subtract Immediate from Word	Rdh:Rdl= Rdh:Rdl-K6	Z,C,N,V,S	2
MUL	Rd,Rr	Multiply Unsigned	R1:R0=Rd*Rr	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0=Rd*Rr	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0=Rd*Rr	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0=(Rd*Rr)«1	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0=(Rd*Rr)«1	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0=(Rd*Rr)«1	Z,C	2

(aus 'ATMEL AVR instruction set Rex. 0856B-06/99' data sheet, © Atmel Corporation 1999.)

Multiplier nicht auf allen Controllern verfügbar.

Branch Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
RJMP	k	Relative Jump	PC=PC+k+1	—	2
IJMP	—	Indirect Jump to (Z)	PC=Z	—	2
EIJMP	—	Extended Indirect Jump (Z)	STACK=PC+1,PC(15:0)=Z, PC(21:16)=EIND	—	2
JMP	k	Jump	PC = k	—	3
RCALL	k	Relative Call Subroutine	STACK=PC+1,PC=PC+k+1	—	3/4
ICALL	—	Indirect Call to (Z)	STACK=PC+1, PC=Z	—	3/4
EICALL	—	Extended Indirect Call to (Z)	STACK=PC+1,PC(15:0)=Z, PC(21:16)=EIND	—	4
CALL	k	Call Subroutine	STACK=PC+2, PC=k	—	4/5
RET	—	Subroutine Return	PC=STACK	—	4/5
RETI	—	Interrupt Return	PC=STACK	I	4/5
CPSE	Rd,Rr	Compare, Skip if equal	if(Rd ==Rr) PC=PC+2/3	—	1/2/3
CP	Rd,Rr	Compare	Rd-Rr	Z,C,N,V,H,S	1
CPC	Rd,Rr	Compare with Carry	Rd-Rr-C	Z,C,N,V,H,S	1
CPI	Rd,K8	Compare with Immediate	Rd-K	Z,C,N,V,H,S	1
SBRC	Rr,b	Skip if bit in register cleared	if(Rr(b)==0) PC=PC+2/3	—	1/2/3
SBRS	Rr,b	Skip if bit in register set	if(Rr(b)==1) PC=PC+2/3	—	1/2/3
SBIC	P,b	Skip if bit in I/O register cleared	if(I/O(P,b)==0) PC=PC+2/3	—	1/2/3
SBIS	P,b	Skip if bit in I/O register set	if(I/O(P,b)==1) PC=PC+2/3	—	1/2/3
BRBC	s,k	Branch if Status flag cleared	if(SREG(s)==0) PC=PC+k+1	—	1/2
BRBS	s,k	Branch if Status flag set	if(SREG(s)==1) PC=PC+k+1	—	1/2
BREQ	k	Branch if equal	if(Z==1) PC=PC+k+1	—	1/2
BRNE	k	Branch if not equal	if(Z==0) PC=PC+k+1	—	1/2
BRCS	k	Branch if carry set	if(C==1) PC=PC+k+1	—	1/2
BRCC	k	Branch if carry cleared	if(C==0) PC=PC+k+1	—	1/2
BRSH	k	Branch if same or higher	if(C==0) PC=PC+k+1	—	1/2
BRLO	k	Branch if lower	if(C==1) PC=PC+k+1	—	1/2
BRMI	k	Branch if minus	if(N==1) PC=PC+k+1	—	1/2
BRPL	k	Branch if plus	if(N==0) PC=PC+k+1	—	1/2
BRGE	k	Branch if greater than or equal (signed)	if(S==0) PC=PC+k+1	—	1/2
BRLT	k	Branch if less than (signed)	if(S==1) PC=PC+k+1	—	1/2
BRHS	k	Branch if half carry flag set	if(H==1) PC=PC+k+1	—	1/2
BRHC	k	Branch if half carry flag cleared	if(H==0) PC=PC+k+1	—	1/2
BRTS	k	Branch if T flag set	if(T==1) PC=PC+k+1	—	1/2
BRTC	k	Branch if T flag cleared	if(T==0) PC=PC+k+1	—	1/2
BRVS	k	Branch if overflow flag set	if(V==1) PC=PC+k+1	—	1/2
BRVC	k	Branch if overflow flag cleared	if(V==0) PC=PC+k+1	—	1/2
BRIE	k	Branch if interrupt enabled	if(I==1) PC=PC+k+1	—	1/2
BRID	k	Branch if interrupt disabled	if(I==0) PC=PC+k+1	—	1/2

Data Transfer Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
MOV	Rd,Rr	Copy register	Rd=Rr	—	1
MOVW	Rd,Rr	Copy register pair	Rd+1:Rd=Rr+1:Rr	—	1
LDI	Rd,K8	Load Immediate	Rd=K	—	1
LDS	Rd,k	Load Direct	Rd=(k)	—	2
LD	Rd,X	Load Indirect	Rd=(X)	—	2
LD	Rd,X+	Load Indirect and Post-Increment	Rd=(X),X=X+1	—	2
LD	Rd,-X	Load Indirect and Pre-Decrement	X=X-1, Rd=(X)	—	2
LD	Rd,Y	Load Indirect	Rd=(Y)	—	2
LD	Rd,Y+	Load Indirect and Post-Increment	Rd=(Y),Y=Y+1	—	2
LD	Rd,-Y	Load Indirect and Pre-Decrement	Y=Y-1, Rd=(Y)	—	2
LDD	Rd,Y+q	Load Indirect with displacement	Rd=(Y+q)	—	2
LD	Rd,Z	Load Indirect	Rd=(Z)	—	2
LD	Rd,Z+	Load Indirect and Post-Increment	Rd=(Z), Z=Z+1	—	2
LD	Rd,-Z	Load Indirect and Pre-Decrement	Z=Z-1, Rd=(Z)	—	2
LDD	Rd,Z+q	Load Indirect with displacement	Rd=(Z+q)	—	2
STS	k,Rr	Store Direct	(k)=Rr	—	2
ST	X,Rr	Store Indirect	(X)=Rr	—	2
ST	X+,Rr	Store Indirect and Post-Increment	(X)=Rr, X=X+1	—	2
ST	-X,Rr	Store Indirect and Pre-Decrement	X=X-1, (X)=Rr	—	2
ST	Y,Rr	Store Indirect	(Y)=Rr	—	2
ST	Y+,Rr	Store Indirect and Post-Increment	(Y)=Rr, Y=Y+1	—	2
ST	-Y,Rr	Store Indirect and Pre-Decrement	Y=Y-1, (Y)=Rr	—	2
ST	Y+q,Rr	Store Indirect with displacement	(Y+q)=Rr	—	2
ST	Z,Rr	Store Indirect	(Z)=Rr	—	2
ST	Z+,Rr	Store Indirect and Post-Increment	(Z)=Rr, Z=Z+1	—	2
ST	-Z,Rr	Store Indirect and Pre-Decrement	Z=Z-1, (Z)=Rr	—	2
ST	Z+q,Rr	Store Indirect with displacement	(Z+q)=Rr	—	2
LPM	—	Load Program Memory	R0=(Z)	—	3
LPM	Rd,Z	Load Program Memory	Rd=(Z)	—	3
LPM	Rd,Z+	Load Program Memory and Post-Increment	Rd=(Z), Z=Z+1	—	3
ELPM	—	Extended Load Program Memory	R0=(RAMPZ:Z)	—	3
ELPM	Rd,Z	Extended Load Program Memory	Rd=(RAMPZ:Z)	—	3
ELPM	Rd,Z+	Extended Load Program Memory and Post Increment	Rd=(RAMPZ:Z), Z=Z+1	—	3
SPM	—	Store Program Memory	(Z)=R1:R0	—	-
ESPM	—	Extended Store Program Memory	(RAMPZ:Z)=R1:R0	—	-
IN	Rd,P	In Port	Rd=P	—	1
OUT	P,Rr	Out Port	P=Rr	—	1
PUSH	Rr	Push register on Stack	STACK=Rr	—	2
POP	Rd	Pop register from Stack	Rd=STACK	—	2

Bit and Bit-test Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
LSL	Rd	Logical shift left	Rd(n+1)=Rd(n),Rd(0)=0,C=Rd(7)	Z,C,N,V,H,S	1
LSR	Rd	Logical shift right	Rd(n)=Rd(n+1),Rd(7)=0,C=Rd(0)	Z,C,N,V,S	1
ROL	Rd	Rotate left through carry	Rd(0)=C,Rd(n+1)=Rd(n),C=Rd(7)	Z,C,N,V,H,S	1
ROR	Rd	Rotate right through carry	Rd(7)=C,Rd(n)=Rd(n+1),C=Rd(0)	Z,C,N,V,S	1
ASR	Rd	Arithmetic shift right	Rd(n)=Rd(n+1),n=0,...,6	Z,C,N,V,S	1
SWAP	Rd	Swap nibbles	Rd(3..0)=Rd(7..4),Rd(7..4)=Rd(3..0)	—	1
BSET	s	Set flag	SREG(s)=1	SREG(s)	1
BCLR	s	Clear flag	SREG(s)=0	SREG(s)	1
SBI	P,b	Set bit in I/O register	I/O(P,b)=1	—	2
CBI	P,b	Clear bit in I/O register	I/O(P,b)=0	—	2
BST	Rr,b	Bit store from register to T	T=Rr(b)	T	1
BLD	Rd,b	Bit load from register to T	Rd(b)=T	—	1
SEC	—	Set carry flag	C=1	C	1
CLC	—	Clear carry flag	C=0	C	1
SEN	—	Set negative flag	N=1	N	1
CLN	—	Clear negative flag	N=0	N	1
SEZ	—	Set zero flag	Z=1	Z	1
CLZ	—	Clear zero flag	Z=0	Z	1
SEI	—	Set interrupt flag	I=1	I	1
CLI	—	Clear interrupt flag	I=0	I	1
SES	—	Set signed flag	S=1	S	1
CLN	—	Clear signed flag	S=0	S	1
SEV	—	Set overflow flag	V=1	V	1
CLV	—	Clear overflow flag	V=0	V	1
SET	—	Set T-flag	T=1	T	1
CLT	—	Clear T-flag	T=0	T	1
SEH	—	Set half carry flag	H=1	H	1
CLH	—	Clear half carry flag	H=0	H	1
NOP	—	No operation	—	—	1
SLEEP	—	Sleep	See instruction manual	—	1
WDR	—	Watchdog Reset	See instruction manual	—	1

Rd: Destination (and source) register in the register file
Rr: Source register in the register file
b: Constant (0-7), can be a constant expression
s: Constant (0-7), can be a constant expression
P: Constant (0-31/63), can be a constant expression
K6; Constant (0-63), can be a constant expression
K8: Constant (0-255), can be a constant expression
k: Constant, value range depending on instruction. Can be a constant expression
q: Constant (0-63), can be a constant expression
Rdl: R24, R26, R28, R30. For ADIW and SBIW instructions
X,Y,Z: Indirect address registers (X=R27:R26, Y=R29:R28, Z=R31:R30)
'nibble': Halb-Byte (4 Bit), HiNibble: B7..4, LoNibble: B3..0

Assembler directives

Directive	Description
.BYTE	Reserve byte to a variable
.CSEG	Code Segment
.DB	Define constant byte(s)
.DEF	Define a symbolic name on a register
.DEVICE	Define which device to assemble for
.DSEG	Data Segment
.DW	Define Constant word(s)
.ENDM, .ENDMACRO	End macro
.EQU	Set a symbol equal to an expression
.ESEG	EEPROM Segment
.EXIT	Exit from file
.INCLUDE	Read source from another file
.LIST	Turn listfile generation on
.LISTMAC	Turn Macro expansion in list file on
.NOLIST	Turn listfile generation off
.ORG	Set program origin
.SET	Set a symbol to an expression

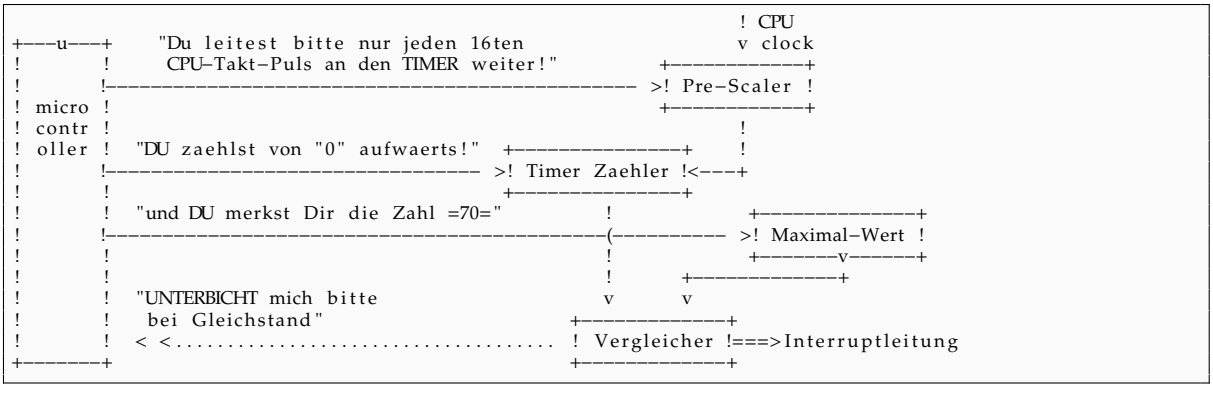
20.6 Microcontroller Basics: Timer

Bitte lern *data sheets*:
- AVR Instruction Set (pdf)
- 8-pol 'TN13' ATtiny13A, 'TN26' ATtiny26
- ATmega8, ATmega16, ATmega32

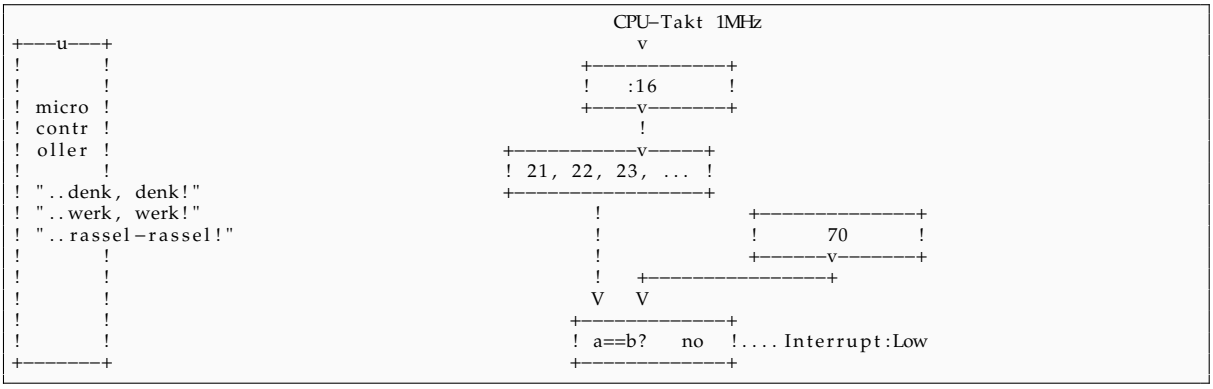
neuerbesser:
- 8-pol AT-Tiny25-45-85(V)
- AT-Tiny261-461-861V
- ATmega88, ATmega168, ATmega328

Timer Module:
- das *busy waiting* (Warteschleifen) hat den **Nachteil**, dass 'derweil' nix anderes erledigt oder reagiert werden kann.
- In der Wartezeit könnte der uC andere Aufgaben bearbeiten, zB.
  - Bedien-Matrix-Keyboard abfragen
  - Sensor-Daten wandeln,
  - rechnen (Skalierung, Linearisierung, binär/-dezimal,...)
  - PC-Kommunikation,
  - Anzeige-Elemente schalten
  - usw.

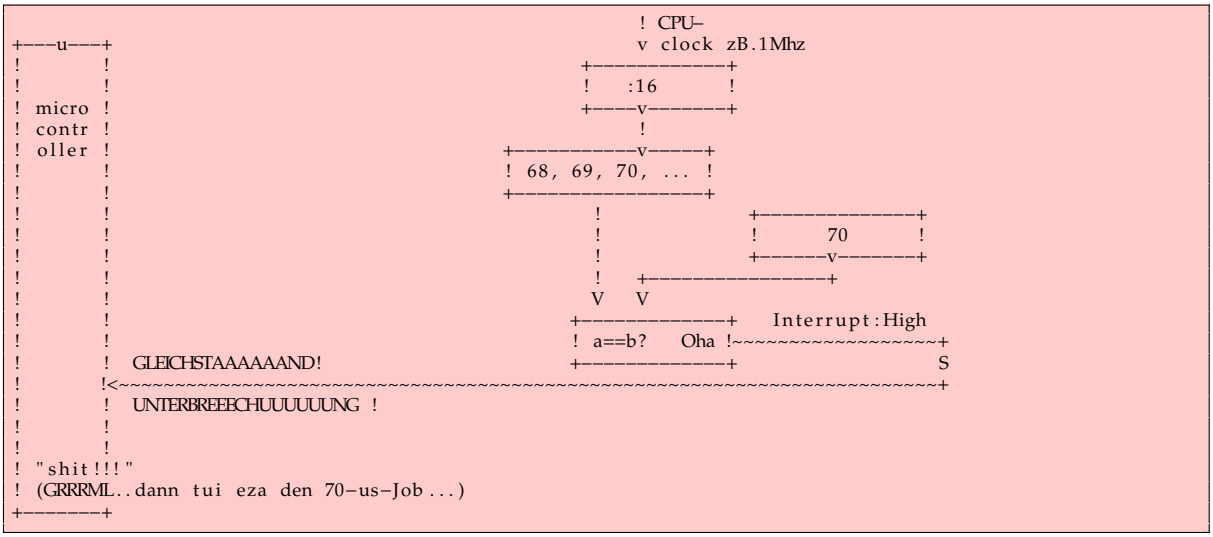
→
deshalb benutzt man **TIMER**. (Zusatz-Hardware im Prozessor-Chip drin):
Einfach ein Binärzähler, der bei jedem CPU-Taktsignal (oder einem Vielfachen davon) weiterzählt und bei einem voreinstellbaren Maximal-Wert einen **'INTERRUPT'** (Unterbrechung der Programmbearbeitung) auslöst.



↓

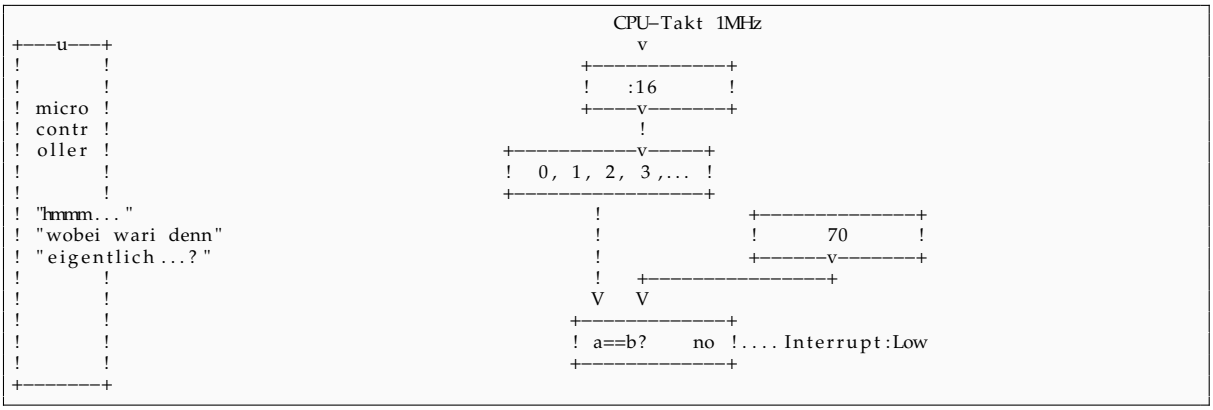


↓



Man nennt das einen ***Timer-Interrupt*** (Voreinstellungen sind weggelassen)

↓



20.7 Tonausgabe mit Timer-Interrupt

```
;-------------------------------------------------------------------------
;Ein-/Aus-schaltbare Tonausgabe mit Timer-Interrupt:
;  fuer ATtiny13
;-------------------------------------------------------------------------
;"temp" ... Temporaer-Speicher-Register
;"ISR" .... Interrupt-Service-Routine
.equ    CpuTaktInHz             =1000000        ;1MHz=Internal.RC.Oszillator
```

```
.equ    TonHertz                =1000
.equ    TakteBisFlanke          =CpuTaktInHz/TonHertz
.equ    TimerPrescaleFactor     =8
.equ    TicksBisFlanke          =TakteBisFlanke/TimerPrescaleFactor
.equ    TimerReloadValue        =256-TicksBisFlanke
;-------------------------------------------------------------------------------
;  PIN ! Name ! Verwendung
;------+------+-----------------
;    01 ! PB5  ! -Reset
;    02 ! PB3  ! -- >LS
;    03 ! PB4  ! -- >liveSQ
;    04 ! Gnd  !
;------+------+-----------------
;    08 ! Vcc  !
;    07 ! PB2  !  &lt;--Taste
;    06 ! PB1  !
;    05 ! PB0  !
;-------------------------------------------------------------------------------
; symbolische Namen fuer Bitnummern und Ports
; ( = aenderungsfreundlicher )
.equ    Pn_Taste        =2      ;PB2
.equ    PT_Taste        =PORTB
.equ    Pi_Taste        =PINB
.equ    Pn_LED          =4      ;PB4
.equ    PT_LED          =PORTB
.equ    Pi_LED          =PINB
.equ    Pn_LS           =3      ;PB3
.equ    PT_LS           =PORTB
.def    temp=R16
.def    ISRtemp=R17
;---------------------------------------
        .CSEG                   ;"ab jetzt ins Code-SEGment (=ProgrammROM) hinein assemblieren"
        .ORG    0x0000          ;ORG = ORiGinate ... Anfangsadressen-Angabe
        ;
        ;Interrupt-(Sprung-)-Vektoren:
        rjmp    MAIN_INIT       ;bei "power-up" wird dieser Befehl ausgefuehrt
        rjmp    EXT_INT0
        rjmp    PINCHG
        rjmp    T0OVF_ISR       ;bei Timer-Unterbrechung wird dieser Befehl ausgefuehrt
        rjmp    EE_RDY
        rjmp    ANACOMP
        rjmp    TMR0_CMPA
        rjmp    TMR0_CMPB
        rjmp    WDR
        rjmp    ADC
;-------------------------
EXT_INT0:
PINCHG:
;T0OVF_ISR:
EE_RDY:
ANACOMP:
TMR0_CMPA:
TMR0_CMPB:
WDR:
ADC:
        RETI
;=====================================================
T0OVF_ISR: ;(=Timer-Interrupt)
        ;diese ISR veraendert -bemerkenswerterweise!- keine MCU-status-flags!
        ; -- > daher muss man keine Register PUSHen/POPen/retten
        LDI     temp,TimerReloadValue
        OUT     TCNT0,temp              ;Zeit-Zaehler Anfangswert neu setzen
        ;
        IN      ISRtemp,PT_LS
        CBI     PT_LS,Pn_LS
        SBIS    Pi_Taste,Pn_Taste
        RJMP    TMR_ENDE
        SBRS    ISRtemp,Pn_LS
        SBI     PT_LS,Pn_LS
TMR_ENDE:
        RETI
;=====================================================
;
MAIN_INIT:
        CLI             ;Interrupt-Sperre (default)
        ;- - - - - - - - - - - - - - - - -
        LDI     temp,0b00011000
        ; /                 !!+--- Taste:In
        ;|                  !+---- LS    :Out
        ; \                 +----- LED   :Out
        OUT     DDRB,temp
        ;- - - - - - - - - - - - - -- - - -

        LDI     temp,0b00010100
        ; /                 !!+--- Taste:Pullup=ON
        ;|                  !+---- LS    :LO=OFF
        ; \                 +----- LED   :HI=OFF
```

```
        OUT     PORTB,temp
        ;- - - - - - - - - - - - - - - - - - - - -

        LDI     temp,   0b00000000      ;2F
        ; /               !!!!  ++---- WGM01+00: 0=normal 2=CTC+OCRA 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
        ;|                !!++-------- Com0B1, Com0B0: 0=off, 3=clear/set, 2=set/clear OC0B(PB.1)
        ; \               ++---------- Com0A1, Com0A0: 0=off, 3=clear/set, 2=set/clear OC0A(PB.0)
        OUT     TCCR0A,temp

        ;- - - - - - - - - - - - - - - - - -
        LDI     temp,   0b00000010      ;33
        ; /               !/  !+++--- CS02..CS00: 0=no 1=clk/1 2=/8 3=/64 4=/256 5=/1024 6+7=extClk
        ;|                !   +------ WGM02: WaveGenMode2: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
        ; \               +---------- FOC0A,FOC0B: ForceOutputCompare
        OUT     TCCR0B,temp

        ;- - - - - - - - - - - - - - - - - -
        LDI     temp,TimerReloadValue
        OUT     TCNT0,temp              ;Zeit-Zaehler Anfangswert neu setzen

        ;- - - - - - - - - - - - - - - - - -
        LDI     temp,   0b00000010      ;39
        ; /               \nc/!!!L---- nc
        ;|                    !!L---- TOIE0: Tmr0Ovfl_IntEnable
        ;|                    !L----- OCIE0A: OutputCompareMatch_A_IntEna
        ; \                   L----- OCIE0B: OutputCompareMatch_B_IntEna
        OUT     TIMSK0,temp
        ;- - - - - - - - - - - - - - - - - -

        LDI     temp,Low(RAMEND)
        OUT     SPL,temp
        ;- - - - - - - - - - - - - - - - - -
        SEI             ;allgemeine Interrupt-Sperre aufheben
;
MainLoop:
        ;-------------------------------------------------
        ;LifeControl-SquareWave, auch fuer charge pumps (LadungsPumpen) geeignet:
        IN      temp,   PT_LED
        CBI     PT_LED, Pn_LED
        SBRS    temp,   Pn_LED
        SBI     PT_LED, Pn_LED
        ;--------------------------------
        ; HIER:
        ; Programme fuer andere Aufgaben
        ; ...
        ;--------------------------------
        ;
        RJMP    MainLoop
        ;
        .EXIT
```

20.8 PWM per Timer-PWM-Module

TBD.

20.9 PWM per Software (Timer-ISR)

PWM mit uC
Anwendung: PWM-gesteuerte Bohrmaschine:

- o das *leere* AVR-Programm
- o *Timer - Interrupt* erzeugt *Grund-Takt*, zB 104,2us
  (104.2us ≡ 1/9600 Baud)
- o Timer-ISR macht Software-PWM:
  (ISR ... Interrupt-Service-Routine)
  - · PWM-Counter 'PwmCtr'
  - · PWM-MaxLim 'PwmMax'
  - · PWM-Tastverhaeltnis 'PwmDuty' od. 'Pwm_t_ein'

- o bei Timer-Interrupt fuer jeden *PWM-Channel*
  erhoehe PwmCtr
  if(PwmCtr==PwmDuty) then Output:=Low;
  if(PwmCtr==PwmMax) then PwmCtr:=0; Output:=High;

```
Ipl3a 22Apr13-1524h
-----------------------

PWM mit AVR-uC:
Unser AVR-Maschinenprogramm hat die Bestandteile:
--Interrupt-Vektoren-Tabelle
--Initialisierung
--Hauptprogramm mit 'Main-Loop' (endlos-Schleife)
--Timer-Interrupt-Service-Routine (ISR)


normalerweise laeuft das HAUPT-Programm (Main-Loop);
-> wenn nun
   (von Hardware)
   ein "Interrupt" ausgeloest wird,
   dann wird das Hauptprogramm unterbrochen (Stelle merken)
   und das ISR ausgefuehrt.
   ISR endet mit dem Befehl "RETI", wodurch das
   Hauptprogramm an der Unterbrechungsstelle fortgesetzt wird.

->fuer das "Merken" der Unterbrechungsstelle
   benoetigt man einen "Stack"-Speicher (Stapelspeicher)
   mit den Funktionen "PUSH"  and "POP"

WHG: Was ist ein uC-Timer?
--> ein Binaer-Zaehler (Kette aus D-FF)
--> Prozessortakt als Clock "CLK"
--> irgendwann erreicht es "11111111"
--> (1) beginnt wieder bei "00000000";
    (2) Uebertrag verursacht "Interrupt"="Timer-Overflow-Interrupt"
    (3) Interrupt-Sperr-Flag (1 Bit) wird gesetzt




;AVR-Programm: elektr.Bohrmaschine
;-----------------------------------
;       Motor: PWM (staendiges, rasches ein/aus) ..... PortB.1
;       Soll-Drehzahl-Sensor: "Pistolen-Abzug" ....... PortB.4 == ADC2
;       Ist-Wert: Tacho-Generator (Drehzahlsensor) ... PortB.3 == ADC3

;       uC: AVR AT Tiny13v
:       hat 11 Interrupt-Sources:
;Int-Vect-Tbl:
        RJMP    INIT    ;RESET als Interrupt
        RETI            ;ExternInterrupt
        RETI            ;PinChangeInterrupt
        RJMP    T0OVFL  ;Timer0-Overflow-Interrupt
        RETI            ;EEready-Int
        RETI            ;AnaComp-Int
        RETI            ;T0CompA-Int
        RETI            ;T0CompB-Int
        RETI            ;Watchdog-Int
        RETI            ;ADC-Int

;Register-Namen statt fixer Register-Nummern vergeben:
; (=guter Stil)
.def    Rtemp   =R16    ;"temporary" fuer kurzes Zwischenspeichern
.def    R_soll  =R17
.def    R_ist   =R18
.def    R_T0RV  =R25    ;"Timer 0 Reload Value"
.set    PwmBit  =1      ;BitNummer des PWM-Ausgangs als "symbolische Konstante"
.def    RflagSave=R2    ;brauchma in der ISR


INIT:
        ;Stackpointer einstellen:
        LDI     Rtemp,Low(RAMEND)
        OUT     SPL,Rtemp

        ;Ein/Ausgaenge einstellen:
        LDI     Rtemp,0b00000011        ;PB0..PB1... output
        OUT     DDRB,Rtemp
        CLR     Rtemp
        OUT     PORTB,Rtemp             ;Outputs auf "Low" schalten

        ;Timer-Mechanismus einstellen+starten:-------------------------------------------------
; (1)
        LDI     Rtemp,  0b00000000      ;2F
        ;                 !!!!  ++---- WGM01,00: WaveGenMode: 3=FastPWM
        ;                !!++--------- Com0B1, Com0B0: 2=clear/set, 3=set/clear OC0B(PB.1)
        ;                ++----------- Com0A1, Com0A0: 2=clear/set, 3=set/clear OC0A(PB.0)
        OUT     TCCR0A,Rtemp
; (2)
        LDI     Rtemp,  0b00000010      ;33
```

```
        ;                   !/  !+++-- CS02..CS00: 1=clk/1, 2=/8, /64, /256, /1024, 6+7=extClk
        ;                   !   +----- WGM02: WaveGenMode2
        ;                   +--------- FOC0A,FOC0B: ForceOutputCompare
        OUT     TCCR0B,Rtemp
; (3)
.equ    C_RT0RV= (256-104)      ;152
        LDI     Rtemp,C_RT0RV
        OUT     TCNT0,Rtemp

        LDI     Rtemp,  0b00000010      ;39
        ;                   ....|||.
        ;                       ||+-------TOIE0: Tmr0IntEnable
        ;                       |+--------OCIE0A:
        ;                       +---------OCIE0B:
        OUT     TIMSK0,Rtemp
; (4)
        LDI     Rtemp,  0b00000000      ;PCIE, PinChangeInterrupt
        ;                   |+----------PCIE: PinChangeInterrupt (see PCMSK0)
        ;                   +-----------INT0: extIntEnable
        OUT     GIMSK,Rtemp
        ;---Ende-Timer-Mechanismus-einstellen-----------------------------------------------------



        ;ADC einstellen:
        LDI     Rtemp,0b00100011        ;set AdcChannel=3
        OUT     ADCMUX,Rtemp
        LDI     Rtemp,0b11000011        ;start conversion
        OUT     ADCSRA,Rtemp


        ;sonstige Anfangswerte:
        LDI     R_PwmCtr,0
        LDI     R_PwmMax,100
        LDI     R_PwmDuty,0     ;PWM-Wert anfangs "0" == Stillstand

        SEI     ;general Interrupt enable ... ab jetzt "rennt" das Interrupt-Zeug

MAINLOOP:
        ;ADC fertig?
        SBIC    ADCSRA,ADSC     ;"ADSC" = Name des "conversion in progress"-Bit
        RJMP    LM_nixAdc

        ;welcher?
        IN      Rtemp,ADCMUX
        ANDI    Rtemp,0b00000011        ;andere Bits ausblenden

        ;wenn Sollwert-ADC:
        CP      Rtemp,2
        BRNE    LM_IST
        ;Sollwert einlesen (ADC)
        IN      R_soll,ADCH
        LDI     Rtemp,0b00100011        ;set AdcChannel=3
        OUT     ADCMUX,Rtemp
        LDI     Rtemp,0b11000011        ;start conversion
        OUT     ADCSRA,Rtemp
        RJMP    LM_AdcFertig
LM_IST:

        ;wenn Istwert-ADC:
          ;CP    Rtemp,3
          ;BRNE LM_AdcFertig
        ;Istwert einlesen
        IN      R_ist,ADCH
        LDI     Rtemp,0b00100010
        OUT     ADCMUX,Rtemp
        LDI     Rtemp,0b11000011
        OUT     ADCSRA,Rtemp
LM_AdcFertig:
LM_nixAdc:

        ;WENN ( IST &lt; SOLL ){
        ;       erhoehen: Tastverhaeltnis (PulsBreite/Pulsgesamtdauer)
        ;}else{
        ;       vermindern: Tastverhaeltnis (PulsBreite/Pulsgesamtdauer)
        ;}
        CP      R_ist,R_soll    ;if...
        BRSH    LM_1
        ;then:
          INC   R_PwmDuty
          CP    R_PwmDuty,R_PwmMax
          BRSH  LM_1
          DEC   R_PwmDuty
LM_1:   ;else:
          OR    R_PwmDuty,R_PwmDuty     ;schauen, ob "Null"
          BRNE  LM_2
          DEC   R_PwmDuty
```

```
LM_2:    ;end



;LEBENSZEICHEN:
         IN       Rtemp,PORTB      ;PINB...InputRegister von Port-B
         CBI      PORTB,0          ;Bit Nr.0 auf "Low"
         SBRS     Rtemp,0          ;Ueberspringe, wenn das Bit==1 war
         SBI      PORTB,0          ;Bit Nr.0 auf "High"


         RJMP     MAINLOOP




;

T0OVFL: ;-------------------------------------------------------------------------------
         ;da in der ISR die CPU-Flags verstellt werden
         ;(was Auswirkungen auf das MainProgramm haette),
         ;muessen wir diese CPU-Flags (sie sind im Speicher "SREG")
         ;vorher in ein sonst unbenutztes Register kopieren,
         ;um sie nachher wieder "restaurieren" zu koennen:
         IN       RflagSave,SREG

         ;so! Jetzt das eigentliche ISR-Zeug:
         OUT      TCNT0,R_T0RV     ;...(verstellt keine Flags)
         ;
         INC      R_PwmCtr
         CP       R_PwmCtr,R_PwmDuty
         BRNE     L_OVFL1
         CBI      PortB,PwmBit     ;PWM auf "Lo"
L_OVFL1:
         CP       R_PwmCtr,PwmMax
         BRNE     L_OVFL2
         CLR      R_PwmCtr         ;wieder bei "0" zaehlen anfangen
         SBI      PortB,PwmBit     ;PWM auf "High"
L_OVFL2:

         OUT      SREG,RflagSave   ;zerstoerte Flags des MainProgramms restaurieren
         RETI                      ;(1) zurueck zur Hauptprog.Unterbrechg.stelle
                                   ;(2) Interrupt-Sperr-Flag := 0
;-------------------------------------------------------------------------------------
```

5-Kanal-SW-PWM mit uC
Anwendung:
zB.Fahrzeug: 4xFahrmotor + 1xLenkung mit PWM → 5-Pwm-Channels:
Konzept

- *Timer - Interrupt* erzeugt *Grund-Takt*, zB 104,2us
  (104.2us ≡ 1/9600 Baud)
- Timer-ISR macht 5-Kanal-Software-PWM:
  (ISR ... Interrupt-Service-Routine)
  - 5 Stk PWM-Counter 'PwmCtr1', 'PwmCtr2', ..., 'PwmCtr5'
  - 5 Stk PWM-MaxLim 'PwmMax1', 'PwmMax2', ..., 'PwmMax5'
  - 5 Stk PWM-Tastverhaeltnis 't_ein1', 't_ein2', ..., 't_ein5'

bei Timer-Interrupt:

```
T0OVFL_ISR:

 erhoehe PwmCtr1
 if( PwmCtr1 == t_ein1 ) then Output1:=Low;
 if( PwmCtr1 == PwmMax1 and t_ein1>0 ) then PwmCtr1:=0; Output1:=High;
    !
    v
 erhoehe PwmCtr2
 if( PwmCtr2 == t_ein2 ) then Output2:=Low;
 if( PwmCtr2 == PwmMax2 and t_ein2>0 ) then PwmCtr2:=0; Output2:=High;
    !
    v
    ...

    !
```

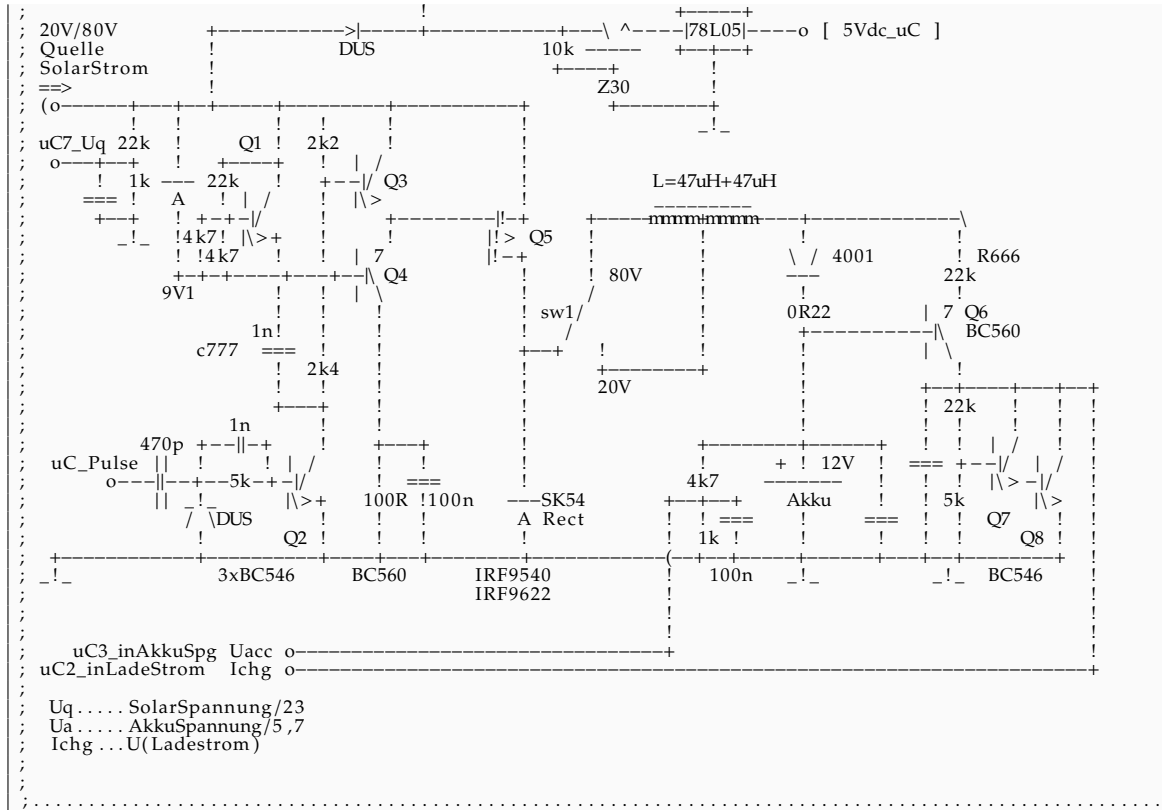
```
    v
erhoehe PwmCtr5
if( PwmCtr5 == t_ein5 ) then Output5:=Low;
if( PwmCtr5 == PwmMax5 and t_ein5>0 ) then PwmCtr5:=0; Output5:=High;

  RETI
```

im Hauptprogramm (MAIN_LOOP) werden Bedienungs-Inputs wahrgenommen und dementsprechend die Tastverhaeltnisse 't_ein1' ... 't_ein5' verstellt.

20.10 Fischis Solar-Lader

Listing 8: Fischis Solar Lader

```
;2do:
;       c777 Entleerung fehlt
;*********************************************************************
; AT-Tiny13 Program
; "TDsoLadLe826.ASM" == TN13.solarLader.Le826
; TN13-solarLadeControllerFuerFS 20Mar11 LcKbt
;       Aufgabe: 12V-Akku aufladen aus Solarzellen
;       Probleme:
;               -20V oder 80V Solarpanel (umloetbar)
;               -absacken der Vsolar im Dunkeln - keine Vcc fuer uC
;               -vertauschen der Anschluesse Akku/Panel
;               -optimaler dutyCycle fuer Switcher
;       Mittelung: Messwerte Vsolar, Vlad, Ilad werden je 8-fach gemittelt
;                       (Adc0+Adc1+Adc2+Adc3+Adc4+Adc5+Adc6+Adc7)/8
;
;*****************************************************************************************************
;bedingte Assemblierung:
 .EQU   IF_mitRegInit    =0
 .EQU   IF_mitRegler     =1
 .EQU   IF_mitAdcMittelung=1
 .EQU   IF_mitCwTiming   =1
 .EQU   IF_mitCwChar     =0
;*****************************************************************************************************
        .equ LadeSchlussSpannung= (256*145*1000/(1000+4700)/5)/10       ;Spannungsteiler an Akku
        .equ LadeGrenzSpannung  = (256*150*1000/(1000+4700)/5)/10       ;Spannungsteiler an Akku
        .equ Unterspannung      = (256*130*1000/(1000+22000)/5)/10      ;Spannungsteiler an Vsolar
        .equ Ilad_max           = 256*1500/5000 ;mal raten: 1500mV ???

        .equ DC_INI_PRZ         = (5+(2*13*96)/100)/10
                                ;=0.02(%)*13us*9.6MHz ;13us/t=0.000013*9600000
;*****************************************************************************************************
.device ATtiny13
;RAM=64Byte      0x60..0x9F
;EEPROM=64 Byte (40H)
;FLASH=512 Word (200H) = 1kByte (0400H: 0x0000..0x03FF)
;.....................................................................
; solLader-ATtiny13 Pin Assignment:
;                       +---u---+
;       -Reset, PB5, ADC0   1       8   Vcc
;  AdcIlad ---- PB3, ADC3   2       7   PB2, SCK  ADC1      ---AdcVsol
;  AdcVlad ---- PB4, ADC2   3       6   PB1, MISO AC1- OC0B ---PwmOut
;  Gnd ----------------     4       5   PB0, MOSI AC0+ OC0A ---liveSQ, LED, cwOUT
;                       +-------+
;
;---------------------------------------------------------------------
;       01      PB5, -Reset, ADC0
;       02      PB3, ADC3: meas Ilad Ladestrom / Isol SolPanelStrom
;       03      PB4, ADC2: meas Vlad Ladespannung
;       04      Gnd
;-------------------------
;       08      Vcc
;       07      PB2, SCK,  ADC1: meas Vin SolPanelSpannung
;       06      PB1, MISO, AC1:  SwitcherPwmOut
;       05      PB0, MOSI, AC0:  liveSQ+CwOut
;---------------------------------------------------------------------
;
.equ    Pn_Pwm          =1      ;PB1
.equ    PT_Pwm          =PORTB
.equ    Pn_Ilad         =3      ;PB3
.equ    PT_Ilad         =PORTB
.equ    Pn_Vlad         =4      ;PB4
.equ    PT_Vlad         =PORTB
.equ    Pn_Vsol         =2      ;PB2
.equ    PT_Vsol         =PORTB
.equ    Pn_LED          =0      ;PB0
.equ    PT_LED          =PORTB
.equ    Pi_LED          =PINB
.equ    Pn_LS           =0      ;PB0
.equ    PT_LS           =PORTB
;
;.....................................................................................................;
; Schaltung:
;.....................................................................................................;
;                                                                                                     ;
;                                                                                                     ;
;               an_Akku+        DUS                                                                   ;
;                  o----------->|-----+              BC639                                           ;
```




```
;
; Uq.....SolarSpannung/23
; Ua.....AkkuSpannung/5,7
; Ichg...U(Ladestrom)
;
;
;.....................................................................................
; R ..... 4k7
; C ..... 100n
; DUS ... 1N4148
; TUN ... BC546, BC550 ... zu schwach fr 3Amp Schaltspitzen!
; TUP ... BC560
; Akku... 12Vdc Blei- od. Gel-Akku
;.....................................................................................
;
;.....................................................................................
; Test-Puls-Generator (LkJ):
;.....................................................................................
;
;     ^                                   ^
;     !                                   !
;     !  ________________                 1k
;    10k   !                !   +-4k7-+      +------o
;     +----!6,2,7    NE555 3!---+      !    | /
;     !    !________________!    +--||-+---|/
;    ===1n                          1n      |\>+
;    _!_                                    !
;                                          _!_
;
;.....................................................................................
;
;*************************************************************************
; uC Register Map:
;-------------------
;R0
;R1
;R2
;R3
.def    RadcAvgCt=R4    ;... ADC conversion Counter = 4_Channels x 8_Mittelungen init= 32_Messungen
.def    R_Ilad=   R5    ;... gemittelter ILad init=0
.def    RaltIlad= R6    ;... alter Ilad init=1
.def    RpwmDc= R7      ;... DutyCylce of OC0B PWM init=2
.def    RT0dv16=R8      ;... Divisor by 16 in T0 timer
.def    Rcwdiv= R9      ;... CW speed divisor
.def    Rdit=   R10     ;... dit duration
.def    Rdah=   R11     ;... dah=3*dit duration
.def    Rcwcod= R12     ;... active CW code
.def    Rcwcnt= R13     ;... active CW tones count
.def    R_Vsol= R14     ;...
.def    R_Vlad= R15     ;...
;
.def    R1flg=  R16     ;... BitFlagMem: CWnbl3:CWpau5:CWnbl2:CWtone : CWbsy:_:T1667:T104
 .def   Rflgt=  R16     ;... BitFlagMem: CWnbl3:CWpau5:CWnbl2:CWtone : CWbsy:_:T1667:T104
.def    R2flg=  R17     ;... BitFlagMem: 0:0:pwmDir:runPwm : _:_:_:_
 .def   Rflgp=  R17     ;... BitFlagMem: 0:0:pwmDir:runPwm : _:_:_:_
.def    Rwk1=   R18     ;... gen.purp.aux.reg., CWCHAR sendchar parameter
.def    Rtmp1=  R18     ;... gen.purp.aux.reg., CWCHAR sendchar parameter
.def    Rint1=  R19     ;... aux @ int.level (T0INT, AdcInt, T1INT, PINCHG,..)
.def    Rwk2=   R20     ;... gen.purp. aux (in MAIN LOOP level)
.def    Rtmp2=  R20     ;... Rwk2
.def    Radcmux=R21     ;... AdMux 0..3
.def    R_Idiv0=R22     ;... T104InterruptCounter/divisor
```

```
;R23
;R24
;R25
;R26 (X1)
;R27 (Xh)
;R28 (Y1)
;R29 (Yh)
;R30... used as "Z"(lo)
;R31... used as "Z"(hi)
;
.equ T104       =0      ;  104us flag
.equ T1667      =1      ; 1667us flag (16 x 104us)
.equ R1flg2     =2      ;bit 2 in R1flg
.equ CWbsy      =3      ;bit 3 in R1flg
.equ CWtone     =4      ;bit 4 in R1flg
.equ CWnbl2     =5      ;bit 5 in R1flg
.equ CWpau5     =6      ;bit 6 in R1flg
.equ CWnbl3     =7      ;bit 7 in R1flg
;
.equ R2flg0     =0      ;bit 0 in R2flg
.equ R2flg1     =1      ;bit 1 in R2flg
.equ R2flg2     =2      ;bit 2 in R2flg
.equ R2flg3     =3      ;bit 3 in R2flg
.equ runPwm     =4      ;bit 4 in Rflgp
.equ pwmDir     =5      ;bit 5 in Rflgp
.equ pwmIsOn    =6      ;bit 6 in R2flg
.equ R2flg7     =7      ;bit 7 in R2flg
;
;*****************************************************************************
;
        .ESEG
        .ORG    0x00
REGTAB:
.equ    DIT_DUR=        30
.equ    C_RT0RV=        152     ;TMR0_RV, 1MHz/104=1M/(256-152)...9615 Bd
;
        .DB     -1              ;R0
        .DB     -1              ;R1
        .DB     -1              ;R2
        .DB     -1              ;R3:
        .DB     32              ;R4: RadcAvgCt: 8meas x 4Chan =32 ... 2.78ms cycle
        .DB     2               ;R5: R_Ilad
        .DB     2               ;R6: RaltIlad
        .DB     DC_INI_PRZ      ;R7: RpwmDc
        .DB     16              ;R8: RT0dv16
        .DB     DIT_DUR         ;R9: Rcwdiv
        .DB     DIT_DUR         ;R10:Rdit
        .DB     3*DIT_DUR       ;R11:Rdah
        .DB     0               ;R12:Rcwcod
        .DB     0               ;R13:Rcwcnt
        .DB     -1              ;R14:
        .DB     -1              ;R15            ;C_RT0RV ;R15:RT0RV
        .DB     0b00000000      ;R16:R1flg:BitFlagMem: 0:CWpau5:CWnbl2:CWtone :: CWbsy:_:T1667:T104
        .DB     0b00100000      ;R17:R2flg:BitFlagMem: 0:pwmIsOn:pwmDir:runPwm : _:_:_:_
        .DB     -1              ;R18:Rwk1:  gen.purp.auxReg, CWCHAR sendchar parameter
        .DB     -1              ;R19:Rint1: auxReg @ int.level (T0INT, AdcInt, T1INT, PINCHG,..)
        .DB     -1              ;R20:Rwk2:  gen.purp. auxReg (in MAIN LOOP level)
        .DB     0               ;R21:Radcmux
        .DB     8               ;R22:R_Idiv0
        .DB     -1              ;R23
        .DB     -1              ;R24
        .DB     -1              ;R25
;       .DB     -1              ;R26-Xlo
;       .DB     -1              ;R27-Xhi
;       .DB     -1              ;R28-Ylo
;       .DB     -1              ;R29-Yhi
;       .DB     30              ;R30-Zlo
;       .DB     0               ;R31-Zhi
;
        .ORG    0x20

;*****************************************************************************
        .DSEG
;.............................................................................
;       TN13 contains 64 BYTE SRAM
;        a) processor stack (subroutine_calls(RCALL/RET) + register_salvage(PUSH/POP))
;        b) SRAM data storage
;.............................................................................
;;;;;;;;        .ORG    0x0060
LORAM:
;
M_AdcMeans:     ;4 x Word:
 M_Adc0Mean:    .BYTE   2       ;unbenutzt:Chan0==Reset(Pin1)
 M_VsolMean:    .BYTE   2       ;ADC1
 M_VladMean:    .BYTE   2       ;ADC2
 M_IladMean:    .BYTE   2       ;ADC3
;
DCWMEM:         .BYTE   2
.equ    CWMEM   =DCWMEM-LORAM
.equ    STACKTOP=LORAM+64
;
;*****************************************************************************
;*****************************************************************************
; TN13 interrupt vectors:
;       JumpTable for InterruptServiceRoutines ("SprungLeiste")
;       see ATtiny13-InterruptSystem-description
;.............................................................................
        .CSEG
```

```
          .ORG    0x0000
          rjmp    MAIN            ;reset handle
          rjmp    EXT_INT0
          rjmp    PINCHG
          rjmp    T0OVF
          rjmp    EE_RDY
          rjmp    ANACOMP
          rjmp    TMR0_CMPA
          rjmp    TMR0_CMPB
          rjmp    WDR
          rjmp    ADC
;***************************************************************************
EXT_INT0:
PINCHG:
;T0OVF
EE_RDY:
ANACOMP:
TMR0_CMPA:
TMR0_CMPB:
WDR:
ADC:
          RETI
;***************************************************************************
T0OVF:    ;alle 100 Instruct's ... 13us (=104,2/8)
          IN      Rint1,SREG              ;1
          ;Timer_0_Jobs - 104,2us :
          ;-----------------------------------
          ;CLKG104:
          ;divide 13us-clocking by 8:
          DEC     R_Idiv0                 ;1
          BRNE    IT01X                   ;2
          LDI     R_Idiv0,0x08            ;1
          CBI     PT_LS,Pn_LS             ;2;CWtone
          ORI     Rflgt,(1<<T104)         ;1
;         SBI     Pi_LED,Pn_LED           ;forCheckPurposes
IT01X:
          OUT     SREG,Rint1              ;1
          RETI                    ;return from interrupt - frees other int's
;
;***************************************************************************
;***************************************************************************
;
;
;.........................................................................
;         this contains the morse code values for ASCII symbols
;         0x30 through 0x5E; some specials in between (0x3A..0x40)
;         b7,b6,b5          are morse code length
;         b4..b0            are morse signal (0..dot, 1..dash)
;.........................................................................
CWTBL:
;compressed code:
; B7+B6+B5 = Len ([0..7])
; B4..B0= CW code (0..dit, 1..dah)
          .DB     0b10111111,0b10101111   ;0, 1
          .DB     0b10100111,0b10100011   ;2 ;3
          .DB     0b10100001,0b10100000   ;4 ;5
          .DB     0b10110000,0b10111000   ;6 ;7
          .DB     0b10111100,0b10111110   ;8 ;9
;
          .DB     0b11011100,0b10110010   ;3AH=':' ;3BH=';'(/)
          .DB     0b10101000,0b10110001   ;3CH='<' ;3DH='='
          .DB     0b10100010,0b11000110   ;3EH='>' ;3FH='?'
          .DB     0b11100000,0b01001000   ;40H='@' ;41H='A'
;
          .DB     0b10010000,0b10010100   ;B ;C
          .DB     0b01110000,0b00100000   ;D ;E
          .DB     0b10000100,0b01111000   ;F ;G
          .DB     0b10000000,0b01000000   ;H ;I
          .DB     0b10001110,0b01110100   ;J ;K
          .DB     0b10001000,0b01011000   ;L ;M
          .DB     0b01010000,0b01111100   ;N ;O
          .DB     0b10001100,0b10011010   ;P ;Q
          .DB     0b01101000,0b01100000   ;R ;S
          .DB     0b00110000,0b01100100   ;T ;U
          .DB     0b10000010,0b01101100   ;V ;W
          .DB     0b10010010,0b10010110   ;X ;Y
          .DB     0b10011000,'Z'            ;Z
;
;-------------------------------------------------------------------------
IOTBL:    ;(TN13)
;0x03     ADCSRB   -:ACME:-:-::-:ADTS2=0:ADTS1=0:ADTS0=0
;0x04     ADCL
;0x05     ADCH
IADCSRB:  .DB     0x23,    0b00000000      ;
;                            !   +++------ADTS: AD-TriggerSelect: 000=FreeRun(if ADATE)
;                            +------------ACME: AnaCompMuxEnable
IADCSRA:  .DB     0x26,    0b10010110      ;ADEN,ADSC,ADATE,ADIF=1,ADIE,ADPS=110
;                          |||||+++-------ADPS presclrSlct: 0=/2 1=/2 2=/4 3=/8 4=/16 5=/32 6=/64 7=/128 (50..200kc)
;                          ||||+----------ADIE=1 InterruptEnable
;                          |||+-----------ADIF=1 InterruptFlag
;                          ||+------------ADATE=1 AutoTriggerEnable
;                          |+-------------ADSC=1 StartConversion command
;                          +--------------ADEN=1 ADC Enable
IADMUX:   .DB     0x27,    0b00100000      ;REFS=0,ADLAR=0,MUX=00
;                            ||   ++------MUX=00 (ADC chan0)
;                            |+------------ADLAR=1 (AD LeftAdjustResult)
;                            +-------------REFS=0 (0=Vcc, 1=int:1.1V)
IACSR:    .DB     0x28,    0b00000000      ;ANACOMP ctrl+stat reg.
```

```
;                           ||||||++--------00 Interrupt on toggle, 01 reserved,
;                           ||||||             10 on falling edge, 11 on rising edge (ACO)
;                           |||||+----------ACME: AnaComp MuxEnable
;                           ||||+-----------ACIE: AnaCompIntEnable
;                           |||+------------ACIF: AnaCompIntFlag(1==reset)
;                           ||+-------------ACO:  AnaCompOutput
;                           |+--------------ACBS: AnaComp BandgapSelect (1,18V=nonInvInput)
;                           +---------------ACD:  AnaComp Disable
IDIDR0: .DB     0x34,   0b00111110      ;14     ;Digital Input Disable Register
;                           !!!!!+--- AIN0D PB0 Pin5
;                           !!!!+---- AIN1D PB1 Pin6
;                           !!!+----- ADC1D PB2 Pin7
;                           !!+------ ADC3D PB3 Pin2
;                           !+------- ADC2D PB4 Pin3
;                           +-------- ADC0D PB5 Pin1 -Reset
;IPCMSK:.DB     0x35,   0               ;15 PinChangeMaskRegister:DFT=0
;IPINB: .DB     0x36,   0               ;16
IDDRB:  .DB     0x37,   0b00000011      ;17
;                           !!!!!+--- Pin5 Ain0 OC0A    ;Funzel+CW
;                           !!!!+---- Pin6 Ain1 OC0B    ;PWM
;                           !!!+----- Pin7 ADC1 T0      ;AdcVsol
;                           !!+------ Pin2 ADC3         ;AdcVlad
;                           !+------- Pin3 ADC2         ;AdcIlad
;                           +-------- Pin1 RESET
;                         !!!!!!!!
IPORTB: .DB     0x38,   0b00100001      ;18
;IPINA: .DB     0x39,   0               ;19
;IDDRA: .DB     0x3A,   0b00010000      ;1A
;IPORTA: .DB    0x3B,   0b00111000      ;1B
;IEECR: .DB     0x3C,   0               ;1C
;IEEDR: .DB     0x3D,   0               ;1D
;IEEAR: .DB     0x3E,   0               ;1E
;IWDTCR:.DB     0x41,   0               ;21
ICLKPR: .DB     0x46,   0b00000000      ;26     ;CLocKPrescaleRegister
;                         !   !!!+--- CLKPS0=0
;                         !   !!+---- CLKPS1=0
;                         !   !+----- CLKPS2=0
;                         !   +------ CLKPS3=0
;                         +---------- CLKPCE...CLKPR ChangeEnable
IGTCCR: .DB     0x48,   0b00000000      ;28
;                         !\____/+--- PSR10:Prescaler reset
;                         !   +------ nc
;                         +---------- TSM:TimerSynchronizationMode
IOCR0B: .DB     0x49,   15      ;29
;                         \\__--------PwmDutyCycle
;;;IOCR0B:        .DB     0x49,   C_RT0RV+1       ;29
;IDWDR: .DB     0x4E,   0               ;2E DebugWireDataRegister:notAccessible
;ITCNT1: .DB    0x4E,   128             ;2E
ITCCR0A: .DB    0x4F,   0b00000011      ;2F
;                         !!!!  ++--- WGM01+00: WaveGenMode: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
;                         !!++------- Com0B1, Com0B0: 0=off, 3=clear/set, 2=set/clear OC0B(PB.1)
;                         ++--------- Com0A1, Com0A0: 0=off, 3=clear/set, 2=set/clear OC0A(PB.0)
IOSCCAL:    .DB     0x51,   $53             ;31 ($00-$7F)
;;;ITCNT0:     .DB        0x52,   C_RT0RV         ;32 ensure startbit duration
ITCNT0:     .DB     0x52,   0x01            ;32 kleinster PWM puls
ITCCR0B:    .DB     0x53,   0b00001001      ;33
;                         !/  !+++--- CS02..CS00: 0=no 1=clk/1 2=/8 3=/64 4=/256 5=/1024 6+7=extClk
;                         !   +------ WGM02: WaveGenMode2: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
;                         +---------- FOC0A,FOC0B: ForceOutputCompare
;IMCUSR:    .DB     0x54,   0b00000000      ;34
;                         |||||||+---PORF:PowerOn reset flag
;                         ||||||+---EXTRF:external reset flag
;                         |||||+---BORF:brownout reset flag
;                         ||||+---WDRF:watchdog reset flag
;                         ++++----nc
;IMCUCR:    .DB     0x55,   0b00000000      ;35
;                         ||||||++--ISC01:00: Int0SenseControl: 0=Level 1=Change 2=falling 3=rising
;                         |||||+---nc
;                         |||++----SM1:0 SleepMode 0=idle 1=ADC_NR 2=PowerDown 3=reserved
;                         ||+-----SE sleep enable
;                         |+------PUD: PullupDisable
;                         +-------nc
IOCR0A:     .DB     0x56,   100             ;36     ;FastPwmMode=7,fclk=9M6, ->fpwm=96k
;ISPMCSR:   .DB     0x57,   0               ;37
;ITIFR:     .DB     0x58,   0               ;38
ITIMSK0:    .DB     0x59,   0b00000010      ;39
;                         \nc/|||!L------ nc
;                              ||+------- TOIE0: Tmr0Ovfl_IntEnable
;                              |L-------- OCIE0A: OutputCompareMatch_A_IntEna
;                              L--------- OCIE0B: OutputCompareMatch_B_IntEna
;IGIFR: .DB     0x5A,   0               ;3A
IGIMSK: .DB     0x5B,   0b00000000      ;3B GeneralInterruptMask
;                            |+----------PCIE: PinChangeInterrupt (see PCMSK)
;                            +-----------INT0: extIntEnable
ISP:        .DB     0x5D,   Low(RAMEND)     ;3D
;ISREG: .DB     0x5F,   0               ;3F
IOTBLE:
;-------------------------------------------------------------------------------
;*****************************************************************************
        .LISTMAC
        .MACRO  PwmOffMacro
        SBRS    Rflgp,  pwmIsOn
        RJMP    L_PWmOffX
        ANDI    Rflgp,  ~(1<<pwmIsOn)
        CBI     PT_Pwm, Pn_Pwm
        LDI     Rtmp1,  0b00000011
;       ;       ;           !!!!  ++--- WGM01+00: WaveGenMode: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
;       ;       ;           !!++------- Com0B1, Com0B0: 0=off, 2=clear/set, 3=set/clear OC0B(PB.1)
```

```
;       ;       ;           ++---------- Com0A1, Com0A0: 0=off, 2=clear/set, 3=set/clear OC0A(PB.0)
        OUT     TCCR0A, Rtmp1
;       ;       ;
;       ANDI    TCCR0B, 0b00001001
;       ;       ;           !/  !+++--- CS02..CS00: 0=no 1=clk/1 2=/8 3=/64 4=/256 5=/1024 6+7=extClk
;       ;       ;           !   +------ WGM02: WaveGenMode2: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
;       ;       ;           +---------- FOC0A,FOC0B: ForceOutputCompare
L_PwmOffX:
        .ENDMACRO
;---------------------------------
        .MACRO  PwmOnMacro
        OUT     OCR0B,  RpwmDc
        SBRC    Rflgp,  pwmIsOn
        RJMP    L_PwmOnX
        ORI     Rflgp,  (1<<pwmIsOn)
        CBI     PT_Pwm, Pn_Pwm
        LDI     Rtmp1,  0b00100011
;       ;       ;           !!!!  ++--- WGM01+00: WaveGenMode: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
;       ;       ;          !!++-------- Com0B1, Com0B0: 0=off, 3=clear/set, 2=set/clear OC0B(PB.1)
;       ;       ;           ++---------- Com0A1, Com0A0: 0=off, 2=clear/set, 3=set/clear OC0A(PB.0)
        OUT     TCCR0A, Rtmp1
        LDI     Rtmp1,  0b00001001
;       ;       ;           !/  !+++--- CS02..CS00: 0=no 1=clk/1 2=/8 3=/64 4=/256 5=/1024 6+7=extClk
;       ;       ;           !   +------ WGM02: WaveGenMode2: 3=FastPWM/TOP=FF 7=FastPWM/Top=OCR0A
;       ;       ;           +---------- FOC0A,FOC0B: ForceOutputCompare
        OUT     TCCR0B, Rtmp1
L_PwmOnX:
        .ENDMACRO
;-------------------------------------------------------------------------------
;*****************************************************************************
;       INIT Program:
;*****************************************************************************
;       this init sequence copies EEPROM contents to
;       Registers R0...R31 and the SFRs (SpecialFunctionRegisters)
;       ie. memory locations 0x00 through 0x5D
;.............................................................................
MAIN:
        ;cpu_REG_init:
        ;(1) copy EEPROM into Registers(32 Byte)
        LDI     R30,  25
        CLR     R31
I_EE1:  OUT     EEAR, R30
        SBI     EECR, EERE      ;EEPROM read enable
        NOP
        IN      Rwk1, EEDR
        ST      Z,Rwk1
        DEC     R30             ;ACHTUNG: DEC setzt kein CY-flag
        BRPL    I_EE1           ;signed test
;
        ;SFR_init:
        ;(2) copy table from PGM-FLASH into I/O-Registers (0x20..0x5F)
        CLR     R29
        LDI     Rwk1,LOW(IOTBLE-IOTBL)
        LDI     R30,LOW(2*IOTBLE)
I_EE2:  DEC     R30
        LPM     Rwk2,Z          ;addressing flash in #Bytes not words!
        DEC     R30
        LPM     R28,Z
        ST      Y,Rwk2
        DEC     Rwk1            ;ACHTUNG: DEC setzt kein CY-flag
        BRPL    I_EE2
;
;
;
        ;set cpu clock prescaler = 1 (statt 8)
        ;-------------------------------------
;ICLKPR: .DB     0x46,   0b00000000      ;26     ;CLocKPrescaleRegister
;                       !   !!!+--- CLKPS0=0
;                       !   !!+---- CLKPS1=0
;                       !   !+----- CLKPS2=0
;                       !   +------ CLKPS3=0
;                       +---------- CLKPCE...CLKPR ChangeEnable
        LDI     Rtmp1,  0x80    ;CLKPCE: clk prescaler change enable
        LDI     Rtmp2,  0x00    ;0000 = osc/1
        OUT     CLKPR,  Rtmp1
        OUT     CLKPR,  Rtmp2
;
        ;LDI    R30,LORAM
        ;SEI                     ;SETB   IE      enable Int's


;APPLICATION_INIT:
;----------------------


.IF(IF_mitRegInit)
        ;INIT_VAR:
        ;------------------------
        LDI     R29,    32      ;.DB    32              ;R4: RadcAvgCt: 8meas x Chan =32 ... 2.78ms cycle
        MOV     R4,     R29
        CLR     R5              ;.DB    2               ;R5: R_Ilad
        CLR     R6              ;.DB    2               ;R6: RaltIlad
        LDI     R29,    DC_INI_PRZ
        MOV     R7,     R29     ;.DB    DC_INI_PRZ      ;R7: RpwmDc
        ;
        LDI     R29,    16
        MOV     R8,     R29     ;.DB    16              ;R8: RT0dv16
        LDI     R29,    DIT_DUR
```

```
        MOV     R9,     R29     ;.DB    DIT_DUR         ;R9: Rcwdiv
        LDI     R29,    DIT_DUR
        MOV     R10,    R29     ;.DB    DIT_DUR         ;R10:Rdit
        LDI     R29,    3*DIT_DUR
        MOV     R11,    R29     ;.DB    3*DIT_DUR       ;R11:Rdah
        CLR     R12             ;.DB    0               ;R12:Rcwcod
        CLR     R13             ;.DB    0               ;R13:Rcwcnt
        ;
        LDI     R16,    0b00000000      ;R16:R1flg:BitFlagMem: 0:CWpau5:CWnbl2:CWtone :: CWbsy:_:T1667:T104
        LDI     R17,    0b00100000      ;R17:R2flg:BitFlagMem: 0:0:pwmDir:runPwm : _:_:_:_
        CLR     R21             ;.DB    0               ;R21:Radcmux
        LDI     R22,    8       ;.DB    8               ;R22:R_Idiv0
.ENDIF


        ;INIT_SRAM:
        ;-------------------
        ;CLR     Rtmp1
        ;STS     M_VsolMean,Rtmp1        ;Mean_Vsol=0
        ;SER     Rtmp1
        ;STS     M_VladMean,Rtmp1        ;Mean_Vlad=255 ... damit er RunterRegelt
        ;STS     M_IladMean,Rtmp1        ;Mean_Ilad=255


        ;INIT_ADC:
        ;-----------------------
        ;ADMUX:
        LDI     Rtmp1,  0b00100000      ;REFS=0,ADLAR=1,MUX=00
;                       ||    ++------MUX=00 (ADC chan0)
;                       |+------------ADLAR=1 (AD LeftAdjustResult)
;                       +-------------REFS=0 (0=Vcc, 1=int:1.1V)
        OUT     ADMUX, Rtmp1
        ;-----------------------------------
        ;ADCSRB:
        LDI     Rtmp1,  0b00000000      ;
;                       !   +++-------ADTS: AD-TriggerSelect: 000=FreeRun(if ADATE)
;                       +-------------ACME: AnaCompMuxEnable
        OUT     ADCSRB, Rtmp1
        ;-----------------------------------
        ;ADCSRA:
        LDI     Rtmp1,  0b11010110      ;ADEN=0,ADSC=0,ADATE=0,ADIF=1,ADIE=0,ADPS=010
;                       |||||+++-------ADPS presclrSlct: 0=/2 1=/2 2=/4 3=/8 4=/16 5=/32 6=/64 7=/128 (50..200kc)
;                       ||||+----------ADIE=1 InterruptEnable
;                       |||+-----------ADIF=1 InterruptFlag
;                       ||+------------ADATE=1 AutoTriggerEnable
;                       |+-------------ADSC=1 StartConversion command
;                       +--------------ADEN=1 ADC Enable
        OUT     ADCSRA, Rtmp1
        ;-----------------------------------



        ;INIT_PWM: (2%)
        ;-------------------
        ;PwmOnMacro

        SEI                     ;SETB   IE      enable Int's

;
;*************************************************************************
;       MAIN PROGRAM LOOP:
;*************************************************************************
;
LOOP:

;-----------------------------------------------
;LifeControl-SquareWave ( +4 chargePump?):
;       IN      Rwk1,   PT_LED
;       CBI     PT_LED, Pn_LED
;       SBRS    Rwk1,   Pn_LED
;       SBI     PT_LED, Pn_LED
;[...] Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of
;DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port. [...]
;       SBI     Pi_LED,Pn_LED           ;forCheckPurposes
;
;
;-----------------------------------------------
; 104,2us-Jobs ? (16x 104,2us = 1667,2 us)
;-----------------------------------------------
CLKG1667:
        SBRS    Rflgt,  T104
        RJMP    LM_16Z
        ;SBI    Pi_LED,Pn_LED           ;forCheckPurposes
;-----------------------------------------------
; 1667us-Jobs ? (16x104,2us = 1667,2 us)
;-----------------------------------------------
        ;divide 104,2us-clocking by 16:
        DEC     RT0dv16
        BRNE    LM_16X
        LDI     Rtmp1,  16
        MOV     RT0dv16,Rtmp1
        ORI     Rflgt,  (1<<T1667)
        ;SBI    Pi_LED,Pn_LED           ;forCheckPurposes
;
.IF( IF_mitCwTiming )
;-----------------------------------------------
;CW signal generator:
;-----------------------------------------------
```

```
          SBRS    R1flg,CWbsy             ;CW-BSY-flag
          RJMP    L0CWX
          DEC     Rcwdiv                  ;sets flags Z,V,N
          BRNE    L0CWX
          TST     Rcwcnt
          BREQ    L0L_CW9
          MOV     Rcwdiv,Rdit             ;Rdit=DIT duration
          SBRC    R1flg,CWtone            ;CW tone?
          RJMP    L0L_PAU
L0L_DIT:;OUT      TCCR1B,RT1slow
          ORI     R1flg,(1<<CWtone)       ;CWtone=1
          SBRC    Rcwcod,7
          MOV     Rcwdiv,Rdah             ;Rdah=DAH duration
          RJMP    L0CWX
L0L_PAU:;OUT      TCCR1B,RT1fast
          ANDI    R1flg,~(1<<CWtone)      ;CWtone=0
          LSL     Rcwcod
          DEC     Rcwcnt
          BRNE    L0CWX
          MOV     Rcwdiv,Rdah             ;Char-Pause
          SBRC    R1flg,CWpau5            ;CWpau5-long pause
          ADD     Rcwdiv,Rdah
          ANDI    R1flg,~(1<<CWpau5)      ;clr CWpau5
          RJMP    L0CWX
L0L_CW9: ANDI     R1flg,~(1<<CWbsy)       ;R1flg.3==CW-busy-flag
L0CWX:
;
;cwTone 9600/16= abt 600Hz pulse gen (CBI is in CLKG104):
          SBRC    R1flg,CWtone
          SBI     PT_LS,Pn_LS
;
.ENDIF
;
          ANDI    Rflgt,~(1<<T1667)
LM_16X:
          ANDI    Rflgt,~(1<<T104)
LM_16Z:
;
;
;
;=============================================================================+
; ADC jobs?                                                                  !
;=============================================================================+
;         ALGORITHM:                                                         !
;.............................................................................+
;  ADCrunctr=8x4=32                                                          !
;  Radcmux=0                                                                 !
;.............................................................................+
;  for(;;){                                                                  !
;  .........................................................................+
;  ! if(ADC[Radcmux] rdy?){                                                  !
;  !     read ADC[Radcmux]                                                   !
;  !     ADCmean[Radcmux] += ADC[Radcmux]                                    !
;  !     Radcmux = Radcmux++ %4                                              !
;  !     ADCrunCtr--                                                         !
;  ! }                                                                       !
;  !.........................................................................+
;  ! if(ADCrunctr ==0){                                                      !
;  !     ADCrunctr =8x4=32                                                   !                    !
;  !     R_Vsol= ADCmean[Vsol] >> 3                                          !
;  !     R_Vlad= ADCmean[Vlad] >> 3                                          !
;  !     alterLadeStrom = Ladestrom R_Ilad;                                  !
;  !     R_Ilad= ADCmean[Ilad] >> 3                                          !
;  !     .....................................................................+
;  !     runPwm=1;                                                           !
;  !     if(AkkuSpannung > LadeGrenzSpannung   )  runPwm=0;                  !
;  !     if(QuellSpannung < UnterSpannung      )  runPwm=0;                  !
;  !     .....................................................................+
;  !     if(runPwm){                                                         !
;  !       if(Ladestrom < alterLadeStrom){ CPL PWMrichtung }                 !
;  !       .....................................................................+
;  !       if(  (AkkuSpannung > LadeSchlussSpannung)                         !
;  !         || (LadeStrom > Ilad_max)){                                     !
;  !         PWMduty = 2                                                     !
;  !         PWMrichtung = 1                                                 !
;  !       }else{                                                            !
;  !         PWMduty+= (PWMrichtung ? +1:-1);                                !
;  !         PWMduty = max(0, min(PWMPeriode, PWMduty));                     !
;  !       }                                                                 !
;  !       .....................................................................+
;  !       PWM_ON(OCR0B = PWMduty)                                           !
;  !     }else{                                                              !
;  !       PWM_OFF                                                           !
;  !     }                                                                   !
;  !     ADCmean[Vsol] =0                                                    !
;  !     ADCmean[Vlad] =0                                                    !
;  !     ADCmean[Ilad] =0                                                    !
;  ! }                                                                       !
;  !.........................................................................+
;  }                                                                         !
;_____________________________________________________________________________!

.IF(IF_mitAdcMittelung)
LAdc10:
          ; ADC jobs?
          SBIS    ADCSRA, ADIF
          RJMP    LADC_END
          SBI     ADCSRA, ADIF            ;clear ADC ready flag
```

```
          ;ADCREAD_AND_ADD:------------------
          ANDI    Radcmux,0x03
          LDI     R30,   M_AdcMeans       ;CLR     R31
          ADD     R30,   Radcmux
          ADD     R30,   Radcmux
          LD      Rtmp1,Z                 ;Z+0..LoByte
          IN      Rtmp2,ADCH
          ADD     Rtmp1,Rtmp2
          ST      Z,Rtmp1
          LDD     Rtmp1,Z+1               ;Z+1..Hibyte
          CLR     Rtmp2                   ;doesnt set CY!
          ADC     Rtmp1,Rtmp2
          STD     Z+1,   Rtmp1
          ;-------------------------
          INC     Radcmux                 ;1
          ANDI    Radcmux,0x03            ;1
          ORI     Radcmux,0b00100000      ;1
;IADMUX:.DB       0x27,    0b00100000     ;REFS=0,ADLAR=1,MUX=00
;                                 ||    ++------MUX=00 (ADC chan0)
;                                 |+------------ADLAR=1 (AD LeftAdjustResult)
;                                 +-------------REFS=0 (0=Vcc, 1=int:1.1V)
          OUT     ADMUX,   Radcmux        ;1
          SBI     ADCSRA, ADSC            ;1 ADC start conversion
;
;AdcMeanValues >>3
          DEC     RadcAvgCt               ;1
          BREQ    LADC_1                  ;BRNE zkurz
          RJMP    LADC_END
LADC_1:   LDI     Rtmp1,(8*4)
          MOV     RadcAvgCt,Rtmp1
          ;
                  .MACRO  LSR3macro
                  LSR     @0              ;1
                  ROR     @1              ;1
                  LSR     @0              ;1
                  ROR     @1              ;1
                  LSR     @0              ;1
                  ROR     @1              ;1    SUM=6
                  .ENDMACRO
          ;
          LDI     R30,M_AdcMeans
          ;
          LDD     R_Vsol, Z+2             ;2 LoByte
          LDD     Rtmp1,  Z+3             ;2 HiByte
          LSR3macro Rtmp1,R_Vsol          ;6
          ;
          LDD     R_Vlad, Z+4             ;2
          LDD     Rtmp1,  Z+5             ;2
          LSR3macro Rtmp1,R_Vlad          ;6
          ;
          MOV     RaltIlad,R_Ilad         ;1 alterLadeStrom = LadeStrom;
          LDD     R_Ilad, Z+6             ;2
          LDD     Rtmp1,  Z+7             ;2
          LSR3macro Rtmp1,R_Ilad          ;6
          ;-------------------------
.ENDIF
;
LAdc20:
.IF(IF_mitRegler && IF_mitAdcMittelung)
;
          ORI     Rflgp,   (1<<runPwm)            ;runPwm=1 (sonst);
          MOV     Rtmp1,   R_Vlad
          CPI     Rtmp1,   LadeGrenzSpannung      ;if(AkkuSpannung > LadeGrenzSpannung   )  runPwm=0;
          BRCC    L_NOPWM
          MOV     Rtmp1,   R_Vsol
          CPI     Rtmp1,   Unterspannung          ;if(QuellSpannung < UnterSpannung      )  runPwm=0;
          BRCS    L_NOPWM
          ;-----------------------------

                                          ;if(runPwm){                                  !
          CP      R_Ilad, RaltIlad        ;  if(Ladestrom < alterLadeStrom){            !
          BRCC    L_NoCplDir              ;
          LDI     Rtmp1,   (1<<pwmDir)    ;    CPL PWMrichtung                          !
          EOR     Rflgp,   Rtmp1          ;  }                                          !
L_NoCplDir:                               ;
          ;-------------------------------;
                                          ;
          MOV     Rtmp1,   R_Vlad         ;  if(  (AkkuSpannung > LadeSchlussSpannung)  !
          CPI     Rtmp1,   LadeSchlussSpannung
          BRCC    L_PwmMINIMAL            ;
          MOV     Rtmp1,   R_Ilad         ;    || (LadeStrom > Ilad_max)){             !
          CPI     Rtmp1,   Ilad_max       ;
          BRCC    L_PwmMINIMAL            ;
          ;                               ;  }else{                                     !
          IN      Rtmp1,   OCR0A          ;  . Rtmp1=PeriodenDauer
          DEC     Rtmp1                   ;  .
          CP      RpwmDc,  Rtmp1          ;  . PWMduty = min(PWMperiode, PWMduty);      !
          BRCC    L_Pwm01                 ;  .
          SBRC    Rflgp,   pwmDir         ;  .
          INC     RpwmDc                  ;  . PWMduty+= (PWMrichtung ? +1:-1);         !
L_Pwm01:                                  ;  .
          TST     RpwmDc                  ;  .
          BREQ    L_Pwm02                 ;  . PWMduty = max(0, PWMduty));              !
          SBRS    Rflgp,   pwmDir         ;  .
          DEC     RpwmDc                  ;  .
L_Pwm02:;                                 ;  .
          RJMP    L_SetPwm
          ;                               ;  .
```

```
L_PwmMINIMAL:                                   ;   .
            LDI     Rtmp1,  2                   ;   . PWMduty = 2                                   !
            MOV     RpwmDc, Rtmp1               ;   .
            ORI     Rflgp,  (1<<pwmDir)         ;   . PWMrichtung = 1                               !
            ;RJMP   L_SetPwm                    ;   }                                               !
L_SetPwm:
            ;OUT    OCR0B,  RpwmDc
            PwmOnMacro
            RJMP    L_ClrMeans
            ;                                   ;
L_NOPWM:                                        ;   !     }else{                                    !
            ANDI    Rflgp,  ~(1<<runPwm)                ;runPwm=0;
            PwmOffMacro                         ;   !       PWM_OFF                                 !
            ;                                   ;   !     }                                         !
            ;
L_ClrMeans:
            CLR     Rtmp2
            LDI     Rtmp1,8
            LDI     R30,    M_AdcMeans          ;   !     ADCmean[Vsol] =0                          !
L_ClM1:     ST      Z+,     Rtmp2               ;   !     ADCmean[Vlad] =0                          !
            ;INC    R30                         ;   !     ADCmean[Ilad] =0                          !
            DEC     Rtmp1
            BRNE    L_ClM1                      ;   ! }                                             !
            ;
.ENDIF
LADC_END:
;
;-----------------------------------
            RJMP    LOOP                ;SJMP   LOOP
;*****************************************************************************
;       end of MAIN
;*****************************************************************************
;
;
;
;-----------------------------------
; hex output in CW (2 digits):
;-----------------------------------
.IF(1==0)
CW_ADC1:
            LDI     R30,DCWMEM
            ST      Z,Rwk2
            ORI     R1flg,(1<<CWnbl3 | 1<<CWnbl2)   ;set R1flg.5=CWnibble2, R1flg.7=CWnibble3
            RJMP    CWHEX
CW_ADC2:LD  Rwk1,Z
            SBRS    R1flg,CWnbl3            ;R1flg.7==CWnbl3
            RJMP    LCW_1
            SWAP    Rwk1
            RJMP    LCW_2
LCW_1:      ANDI    R1flg,~(1<<CWnbl2)      ;clear R1flg.5=CWnibble2
            ORI     R1flg,(1<<CWpau5)       ;signal CWpau5-long Pause
LCW_2:      ANDI    R1flg,~(1<<CWnbl3)      ;clear R1flg.7=CWnbl3
CWHEX:
            ANDI    Rwk1,0x0F
            ORI     Rwk1,0x30
            CPI     Rwk1,0x3A
            BRCS    L_CWH9
            LDI     Rwk2,7
            ADD     Rwk1,Rwk2
L_CWH9:
.ENDIF
.IF( IF_mitCwTiming && IF_mitCwChar )
;
CWCHAR:             ;char in Rwk1
            PUSH    Rwk1
            SUBI    Rwk1,0x30               ;.lt. 30H
            BRCS    CWC_X                   ;underflow (borrow on CY)
            CPI     Rwk1,0x2B               ;'Z'==5AH
            BRCC    CWC_X
            LDI     R30,LOW(2*CWTBL)
            ADD     R30,Rwk1
            ;CLR     R31
            LPM     Rcwcod,Z                ;addressing flash in #Bytes not words!
            ;LDI    R30,LORAM
            LSL     Rcwcod
            ROL     Rcwcnt
            LSL     Rcwcod
            ROL     Rcwcnt
            LSL     Rcwcod
            ROL     Rcwcnt
            INC     Rcwdiv                  ;fake an END_OF_DIT/DAH
            ORI     R1flg,(1<<CWbsy)        ;0x08 set R1flg.3 bit-CWflag
CWC_X:
            POP     Rwk1
            RET
.ENDIF
;
;
;*****************************************************************************
            .EXIT
;*****************************************************************************
;*****************************************************************************
AnaComp-PropagationDelay:
                        750ns typ (2.7Vcc)
                        500ns typ (4.0Vcc)
;*****************************************************************************
; ATtiny13 Pin Assignment:
;                       +---u---+
```

```
; -Reset, ADC0, PB5      1        8    Vcc
;  PB.3,  ADC3           2        7    PB2, SCK, ADC1
;  PB.4,  ADC2           3        6    PB1, MISO, AC1
;  Gnd                   4        5    PB0, MOSI, AC0
;                        +--------+
;*****************************************************************************
;-----------------------------------------------------------------------------
;MOD HISTORY:
;-----------------------------------------------------------------------------
; V000-LcKb0:
;       FileCreate, PinAssignment, SFRvalues
;       LcKcb: FS_Tel
;*****************************************************************************
; V002-LdV:
;       SwitcherSchaltung
;       Pinbelegung
;       SPICE-Simulation
;*****************************************************************************
; V003-Le1:
;       LochrasterPlatine 100x80, Kuehlkoerper, IRF9540, Bohrungen, Schraubklemmen, Fuesse
;       SPICE-Simulationen mit Mess-Stromspiegel
;       SPICE-Simulationen mit Darlington-FetTreiber
;*****************************************************************************
; V004-Le2
*s9olLad1.cir SolarLade-Switcher Le1lg
*SPICE-Netzliste

Vcc     Vcc     0       DC 60V
Vin     in      0       DC 0 PULSE( 0 5 500nsDelay 20nsTr 20nsTf 2usPW 10usPeriod )

*Schalter:
*M1     D1 G1 S1 S1             IRF9622 W=1.4 L=2u
M1      D1 G1 S1 S1             IRF9540 W=1.5 L=2u
 V_ivcc Vcc     S1      DC 0

*PegelShifter_Q1:
 V_iin  in      inP1    DC 0
Rq1b    inP1    B1      5k
Q1      C1 B1 E1        BC546
 Vq1e   E1      0       DC 0

*PushPull-Treiber_Q2Q3:
R2      Vcc     B2      5k
R4      B2      C1      5k
XD3     B2      Vcc     Z07
*
Rq2c    Vcc     C2x     2.2
Q2      C2x B2  B2a     BC546
Q2a     C2x B2a E2      BC546
 Vq2e   E2      G1      DC 0
Rb2a    B2      B3a     1
Q3a     B3b B3a  G1     BC560
Q3b     G1 B3b  C3      BD139
Rq3be   B3b     C3      470
Rc3     C3      R_c3    22
 Vq3c   R_c3    0       DC 0
*

*Drossel/Trafo-Pumpe:
L2      D1      L1a     47uH
K12     L2      L1      0.95
C12     D1      0       10p
R12     D1      0       5k
*
Drect1  0       D1k     SK54
 VD1a   D1k     L1a     DC 0
L1      L1a     Akku    47uH
CL1b    Akku    0       1n
Rakku   Akku    AkkuP   100mOhm
Vakku   AkkuP   AkkuM   DC 12V
 VakkuM AkkuM   0       DC 0

.control
alter   Vcc     60V
TRAN    50ns    40us
*plot   tran1.V(D1)     tran1.V(Akku)   tran1.V(G1)     tran1.V(in)
plot    tran1.v(B1)     tran1.v(B2)     tran1.v(C1)     tran1.V(G1)
*plot   tran1.i(V_ivcc) tran1.i(VakkuM) tran1.i(VD1a)
plot    tran1.i(Vq1e)   tran1.i(Vq2e)   tran1.i(Vq3c)

alter   Vcc     80V
TRAN    25ns    20us
plot    tran2.V(D1)     tran2.V(Akku)   tran2.V(G1)     tran2.V(in)
*plot   tran2.i(V_ivcc) tran2.i(VakkuM) tran2.i(VD1a)
plot    tran2.i(Vq1e)   tran2.i(Vq2e)   tran2.i(Vq3c)


alter Vcc 60V
TRAN    20ns 800ns 500ns
plot    tran3.V(D1)     tran3.V(Akku)   tran3.V(G1)

TRAN    20ns 3000ns 2400ns
plot    tran4.V(D1)/50  10*(tran4.V(G1)-50)     tran4.V(B2)     tran4.V(in)
.endc


.model 1N4148   D ( IS=2.495E-9 BV=100 RS=0.4755 N=1.679 tt=3.030E-9 cjo=1.700E-12 Vj=1 M=0.1959 Ibv=0.0001 )
.model SK54     D
```

```
.model ZPD10     D BV=6.8
.model ZPD15     D BV=15.0
.MODEL BC546 NPN ( IS=1.8E-14 BF=300 VAF=80 VAR=12.5)
.MODEL BC560 PNP ( IS=1.8E-14 BF=200 VAF=80 VAR=12.5)
.model BD139 NPN(Bf=40 Is=10f Vceo=90 CJE=124p CJC=30p )
.model BD138 NPN(Bf=40 Is=10f Vceo=50 CJE=124p CJC=30p )
.model BD140 PNP(Bf=40 Is=10f Vceo=90 CJE=124p CJC=30p )

.model IRF9540  PMOS( Level=3 Kp=10.18u Vto=-3.646 Is=54.08E-18 Gamma=0 Delta=0 Eta=0 Theta=0 Kappa=0.2 Vmax=0 Xj=0
+                Tox=100n Uo=300 Phi=.6 Rs=64.15m Rd=62.45m Cbd=2.029n Pb=.8 Mj=.5 Fc=.5 Cgso=1.033n Cgdo=469.4p )
****            W=1.5 L=2u Vto=-3.646 Kp=10.18u Rds=444.4K
.model IRF9622  PMOS(Level=3 Kp=10.15u Vto=-3.735 Is=92.08E-18 Gamma=0 Delta=0 Eta=0 Theta=0 Kappa=0.2 Vmax=0 Xj=0
+                Tox=100n Uo=300 Phi=.6 Rs=.339 Rd=1.136  Cbd=426.4p Pb=.8 Mj=.5 Fc=.5 Cgso=1.359n Cgdo=32.22p)

.subckt Z07      minus   plus
Q1       plus  B minus   BC546
Rc       plus    B       22k
*Rc      plus    B       24.5k
*Re      B       minus   2.2k
*Re      B       minus   2.35k
Re       B       minus   2.7k
.ends
.subckt Z07x     minus   plus
Q1       plus  B minus   BC546
Rc       plus    B       22k
Re       B       minus   2.2k
.ends
.end
;
;*************************************************************************
;
*curmi11.cir specialCurrentMirror_4_SolarLader Le1c9
*SPICE-Netzliste

Vcc      Icc     0       DC 0
Icc      Icc     plus    DC 0

D1       plus    Kat     1N4002
Rmess    Kat     B1      0.22R
Rakku    B1      AkuP    0.1R
Vakku    AkuP    0       DC 12V

Re1      plus    E1      11k
Q1       C1  B1  E1      BC560
Rcb2     C1      B2      22k
Q2       C1  B2  E2      BC546b
Q3       C1  E2  Amm2    BC546b
Rbe2     B2      Amm2    5k
VIq1c    Amm2    0       DC 0

.model 1N4148   D
.model 1N4002   D (IS=1n RS=5m N=1.65 TT=6u CJO=10p M=0.333 BV=100 IBV=1E-5 )
.model Z5v1     D BV=5.1
.model BC560    PNP is=50f bf=250 vaf=100
.model BC546b NPN is=50f bf=250 vaf=100

.control
DC       Icc 0 3 0.1
PLOT     i(Vakku)        v(E1,B1)        v(C1)   v(plus,Kat)
PLOT     i(VIq1c)
PLOT     v(C1)

DC       Icc 0 30 0.1
PLOT     v(E1,B1)        v(C1)   v(plus,Kat)
.endc
.end
\end{verbatim}
\begin{comment}
;
;*************************************************************************
; V005:Le826
;       Denkfehler: nichtlineare Strommessung tabellarisch linearisieren
;       richti:     brauchma nit: streng monoton is eh, fyr Optimieren reichts
;       eMail an NeuTh, ParMa, ReiPh, SchAl, WolMa
;       Le826-Le841: SchaltSkizze
; V006:Le8gp-Le8ia: SchaltSkizze+PseudoCode
; V007:LeFgf-Le8hf: Korrekturen
; V008:LfAgk-LeFhd: 96kHz-PWM, RegInit:Chk
; V009:LfAi9-LfAjb: TimerINT-T0OVF
; V010:LfB7f-LfB90: RegMap,RegInit,T0-Timer, R2flg->Rflgt
; V011:LfB9j-LfBc6-LfBeh-LfBig-LfC6p-LfC9f-LfCca-LfCdd
;       LfB(Sa-11) 0715-1040 1100-1434 1815-1900 Int-Config, ISRs, SFR-Init
;       LfC(So-12) 0650-0852 0930-1330 1400-1530 ADC-Mittelung, MainAlgorithm
; V012:LfD(Mo-13)
;       0839-1330 HTL:SoLaderLeata
;       LfD:1620-1830 HTL:SoLaderPgm
;               R2flg->Rflga, Rflgp
;       1935-1950 ersterRun: LifeSqWv_OK, OSCCAL-ClockAdjust, TimingsProbleme...geht!
;               1.62ms --> 617Hz
;               cwChar hat 1 Vogel
;       LfDjp:1950-2201 Fehlersuche:keinePWM
; V013:LfEak(Di-14)
;       PWMgehtNichtFehlerSuche-DivVersuche+PdfLesa
;       1214h: ClkReadjust: OSCCAL=0x53 fuer 5Vdc
;               weder PWM noch TMR0-CMPB-Interrupt sind da!
;       1222h: mit WGM2=0 und TOP=0xFF geht die PWM und der TMR0-CMPB-Interrupt
;       1349h: PWM geht, Fehler im Equipment: Brennt EEPROM nicht
;       LfEdo: CW wieder einbauen OK
```

```
;       Regler verstellt das PWM-Timing
;       ADC-Interrupt kimmt, AdcMittelungOK, ReglerMurXt!
;       -1420h
; V014:LfGlh(Do-16)
;       2133-2230 FehlersucheRegler-LADC20..LADC_END
;  LfH9n(Fr-17)
;       0946-1100 RegInit:aussa, 1052h:Fehler in "ClrMeans":DEC Rtmp2 statt Rtmp1
;       1052-1130 Test der ADCs: Ausgabe auf OC0B(Pin6) als R_Vsol/2 R_Vlad/2 R_Ilad/2
;       1059: AdcChannels ganz durcheinand
;       1150: AdcJob auf ADIF-flag (ohne ADC-INT) umgestellt
;  LfHbt(Fr-17)
;       1212: ADC-Int aussa
;       Vsol ist ??? fix
;       Vlad ist Vsol
;       Ilad ist Vlad
;  LfHe6
;       1403: R_Ilad liest Vsol+Vlad
;       1413: ADC-Versuche zum Fehlersuchen
;             freeRun:AdcMux tut nit recht -> singleConversions(ADSC) statt freeRun
;       1529: AdcChannels einzeln giahn, GesamtDing geaht nit!
;       1539: unklar, ob die floatenden ADCs influenzieren oder die SW alles mixt
;       2209: "adcRdy" gloescht, "Rflga" gloescht
;       2214: "RadcRunCt"->"RadcAvgCt"
;       PwmWiederEinschalten, PwmOnMacro und PwmOffMacro "schoener"
;       bis 2336h
; LfKh1-1702-1930h
;             PulslaengenBegrenzer-Schaltung, CR-Glied
;             dt=dI*L/U=5A*50uH/20V (Kurzschluss) = 250/20=12,5us ... max ~20us
;             15us von 5V auf 0,7V
;             0,7V = 5V exp(-dt/RC)
;             ln( 0,7/5 ) = -dt/RC
;             RC = -15us/ln(0.7/5) = 7,63u
;             (zB R=5k C=1n33    ... die 5k sind als Rbasis=4k7 schon drin!)
;             --> 4k7 10n
; Li58f:0830 SchaltplanBeschriftungen
;       0840 PulsbegrenzerSchaltg
;       0855-1356 Testaufbau+Check mit 5Vdc u. 200ns/10us:0V:5V: Mos schaltet zwiani
; Li5ds:1356 1n=BeschlCap an Q2/Q4
;       +    5k --> 3k3
; Li87c:0724 100n parallel zu den 100R an Q2
; LjA95-1205    78L05: verdraht drin
;             ohne Last: schaltet kaum ab:slope=-8V/taus
;             mit 10 Ohm Last: schaltet fb scharf!
;             Sensoren nit verdrahtet
; LjAg5-1730    Check+SensorenVerdrahten
; LjM92:        TDsoLadLjM92.ASM: PulsBegrenzer verlangert: 470p -> 470p +1n
; LjTa1-1220    Schaltfehler: Q3.Base mit Q4.Base verbinden
;             neu: Q2.B.Beschl.C==1n : verschleift Einschaltflanke von uC_Pulse
;                  --> kleiner machen, R(B,Gnd)?
;             Messung: Vcc_Zulauf.L zu gross -> schwingt -> muss daempfen
;             c777 fraglich? (abschalt?, Abschalt-Klemmdiode?)
; LjTjc-2110    Q2.B.Beschl.C==220p: vy besser
;             longPulses: MieseFallFlanke
; LjU9k-1330    nuj: R_L = 2R2
;             Test: Vcc=20Vdc: p=0.3 FET schaltet fb
;             ZenerUnit:geahtNit: FetGate: -18V-Spikes bei Vcc=30Vdc
;             --> U-Teiler niederohmiger?
;             --> habe 9V1/1.3W parallelgeschaltet: -12V Spikes bei Vdc=31Vdc
;             nextItem: U+I-Messung
; LjUgf-2130    alleSignaleOK
;             Ilad-Messung: vielz klein -> link vergessen
;             -- 4V/6R8 = 590mA -> 1.5Vdc/Ilad
;             Vlad-Messg: eher klein -> OK
;             Vsol-Messung: entspricht
;             Ilad: R666 von 11k auf 22k
; LjVaf-1350    auf 80V umleata (2x WdgZahl)
;             PulsBegrenzer auf 470p verkleinert -> schaltet numma ab (TransistorSpeicherLadg)
;             PulsBegrenzerWieder auf 470p+1n
;             Freilauf/RichtDiode auch fuer 2te Wicklg (SK54) (1N4148 brennt sofort ab!)
;             Test: alls ok
;             2do: 220p-BeschlC no schianer anloeten
;             2do: TestMitAkku
;             2do: Software
; Lk4l8-2228    Test mit aktuellem PGM:
;                     zwiani t_on(PWM)
;                     LED brennt olli: is zugleich PT_LS
;                     wenn Vakku zerst anliegt -> ka PWM
;                     es gibt an intermittierenden Zustand 20%PWM/80%nix
;                     schaltet ba zwiani Vsolar die PWM nit ab
;                     --> "LadeschlussSpannung", "GrenzSpannung" + "Unterspannung" Werte waren falsch!
; Lk6d0-1500    Test mit 6V-Bat:OK, mit 12V-Bat:OK
;             (Vsol=6..31V Lastkurve mit Strombegrenzung simuliert)
;             reagiert auf Unterspannung + IladMax
;             Ladeschluss nicht simuliert
;             uC hat no falsche EEPROM-Vals (nomal burnen)
;             ------------------------------------------
; Lk6ha-1900    DOC
; Lk6i8-2155    uC mit korrektem EEPROM-Inhalt brennt
;             Tests in allen Varianten von Vsol und Vakku: geht supi!
;             Test: LadeGrenzSpannung: OK
;             ------------------------------------------
; LkJj3-1930    Test-Puls-Generator-Schaltung mit NE555
;             ------------------------------------------
;
;
```

21 AVR'ing in 'C'

21.1 Das Leere AVR PGM in 'C'
21.2 Das kleine, leere AVR Programm

Im Assemblercode bestand es aus

- interrupt vector table
- initialization section containing
  - CLI (block all interrupts)
  - SP stack pointer init
  - PORTB DDR and Data init
  - SEI (set general interrupt flag)

- eternal main loop with sign-of-life

Listing 9: dasleereavrpgm.c

```
/* File 'dasleereavrpgm.c', Charles Elwood YEAGER, 13Feb23-???
 * Version: v0.02
 * das kleine, leere AVR Programm in 'C'
 * compile: 'avr-gcc dasleereavrpgm.c'
 * download:(use avrdude)
 * run/stop:(power up/down)
 ****************************************/

#define __AVR_ATmega16__         //uC type
#include <avr/io.h>

//no interrupt vector table to write :-))

int main(void){
Initialization:
  //cli() not necessary here
  DDRB  = 0x03;                  // 0000 0011;  0==input 1==output
  PORTB = 0x03;                  // 0000 0010;  B.7-2: noPullups B.1:Hi B.0:Hi
  //sei(); not necessary here

MainLoopForever:                 // self explainig label
  for( ; ; ){
    SignOfLife:
    PORTB ^= (1<<1);             //invert PortB.1: 0000 0010
  }
}
```

Am 27Okt.15 haben wir es aus der Assembler Version übersetzt. Alle Teilnehmer haben dieses 'leere AVR PGM' auswendig gekonnt.

21.3 ISR f. AM16 in 'C'
21.4 AM16 AVR gcc Interrupt Vectors

modAvr1/ucIntVect

Listing 10: Bsp. mit ISR

```
/* 'testPkF1.c' XH@PkF/Nov15
 * demo programm: deviceType, ISR, sign-of-life
 * compile : 'avr-gcc -S testPkF1.c'
 * download: (use 'avrdude')
 * run     : (power up AM16 device)
 * stop    : (power off AM16 device)
 */
#define __AVR_ATmega16__
#include <avr/io.h>
#include <avr/interrupt.h>

ISR( INT0_vect ){ PORTB ^= (1<<1); }
ISR( TIMER0_OVF_vect ){ PORTB ^= (1<<2); }

int main(void){
  cli();
  DDRB  = (1<<0);
  PORTB = 0x01;
  /* INT0_enable() missing here! */
  /* Timer0_Init() missing here! */
```

```
  sei();
  for(;;){
    PORTB ^= (1<<0);
  }
}
```

Listing 11: avr-gcc int vectors (iom16.h)

```
/* Interrupt vectors */
/* Vector 0 is the reset vector. */
/* External Interrupt Request 0 */
#define INT0_vect                       _VECTOR(1)
#define SIG_INTERRUPT0                  _VECTOR(1)

/* External Interrupt Request 1 */
#define INT1_vect                       _VECTOR(2)
#define SIG_INTERRUPT1                  _VECTOR(2)

/* Timer/Counter2 Compare Match */
#define TIMER2_COMP_vect                _VECTOR(3)
#define SIG_OUTPUT_COMPARE2             _VECTOR(3)

/* Timer/Counter2 Overflow */
#define TIMER2_OVF_vect                 _VECTOR(4)
#define SIG_OVERFLOW2                   _VECTOR(4)

/* Timer/Counter1 Capture Event */
#define TIMER1_CAPT_vect                _VECTOR(5)
#define SIG_INPUT_CAPTURE1              _VECTOR(5)

/* Timer/Counter1 Compare Match A */
#define TIMER1_COMPA_vect               _VECTOR(6)
#define SIG_OUTPUT_COMPARE1A            _VECTOR(6)

/* Timer/Counter1 Compare Match B */
#define TIMER1_COMPB_vect               _VECTOR(7)
#define SIG_OUTPUT_COMPARE1B            _VECTOR(7)

/* Timer/Counter1 Overflow */
#define TIMER1_OVF_vect                 _VECTOR(8)
#define SIG_OVERFLOW1                   _VECTOR(8)

/* Timer/Counter0 Overflow */
#define TIMER0_OVF_vect                 _VECTOR(9)
#define SIG_OVERFLOW0                   _VECTOR(9)

/* Serial Transfer Complete */
#define SPI_STC_vect                    _VECTOR(10)
#define SIG_SPI                         _VECTOR(10)

/* USART, Rx Complete */
#define USART_RXC_vect                  _VECTOR(11)
#define SIG_USART_RECV                  _VECTOR(11)
#define SIG_UART_RECV                   _VECTOR(11)

/* USART Data Register Empty */
#define USART_UDRE_vect                 _VECTOR(12)
#define SIG_USART_DATA                  _VECTOR(12)
#define SIG_UART_DATA                   _VECTOR(12)

/* USART, Tx Complete */
#define USART_TXC_vect                  _VECTOR(13)
#define SIG_USART_TRANS                 _VECTOR(13)
#define SIG_UART_TRANS                  _VECTOR(13)

/* ADC Conversion Complete */
#define ADC_vect                        _VECTOR(14)
#define SIG_ADC                         _VECTOR(14)

/* EEPROM Ready */
#define EE_RDY_vect                     _VECTOR(15)
#define SIG_EEPROM_READY                _VECTOR(15)
```

```
/* Analog Comparator */
#define ANA_COMP_vect                   _VECTOR(16)
#define SIG_COMPARATOR                  _VECTOR(16)

/* 2-wire Serial Interface */
#define TWI_vect                        _VECTOR(17)
#define SIG_2WIRE_SERIAL                _VECTOR(17)

/* External Interrupt Request 2 */
#define INT2_vect                       _VECTOR(18)
#define SIG_INTERRUPT2                  _VECTOR(18)

/* Timer/Counter0 Compare Match */
#define TIMER0_COMP_vect                _VECTOR(19)
#define SIG_OUTPUT_COMPARE0             _VECTOR(19)

/* Store Program Memory Ready */
#define SPM_RDY_vect                    _VECTOR(20)
#define SIG_SPM_READY                   _VECTOR(20)
```


21.5 AVR performance accounting

Eine Messung ist natürlich 'sicherer' (auch da gibts Fehlerquellen) als die Rechnung am Schreibtisch, wo man sich mit 'cycles', 'clock sources' und *Megahertzen* vertun kann. Die Messung erfordert aber Laborausrüstung plus *fälschende* Vorkehrungen in der zu messenden Software (wie zB. das *Lebenszeichen* - indem mans für immer drin lässt, entgeht man der Messwertfälschung) und ist insgesamt zeitaufwendiger.

fach (es ist halt mühselig) die Ausführungszeiten (und den Speicherbedarf) der einzelnen Instruktionen aus dem Datenblatt (*'AVR instruction set pdf'*) heraussuchen und aufsummieren. Als Masseinheit sind 'cycles' angegeben (da die CPU-Taktfrequenz das Tempo bestimmt). Diese sind anschliessend in Ausführungszeiten umzurechnen, zB.

clock:1MHz → 1 cycle == 1us
clock:2MHz → 1 cycle == 0.5us
. . . usw.

21.5.1 perf.acct'ing Assembler

Für Assemblercode- Programme kann man *ein-*

Beispiel:

Listing 12: performance acct. ASM

```
MAINLOOP:
cycles  instruction
---     ------------
2       IN      Rtemp,PORTB     ;PINB...InputRegister von Port-B
2       CBI     PORTB,0         ;Bit Nr.0 auf "Low"
1.5*    SBRS    Rtemp,0         ;Ueberspringe, wenn das Bit==1 war
... 1 if condition is false
... 2 if condition is true
... unsere ist 50% true -> 0.50*1 + 0.50*2 = 1.5
2       SBI     PORTB,0         ;Bit Nr.0 auf "High"
2       RJMP    MAINLOOP
---     ------------
9.5 cycles total
== 9.5us at 1.0 MHz CPU clock
```

Unsere MainLoop dauert im Mittel 9.5us mit der *'factory default internal cpu clock source'*; in Wahrheit abwechselnd 9us / 10us - das ist am Oszilloskop noch unterscheidbar!

21.5.2 perf.acct'ing 'C'

In Hochsprachecode wie 'C' ist der Zeitbedarf nicht mehr direkt aus dem Instructionset-Datenblatt ersichtlich. Entweder hat man zusätzliche Entwicklungswerkzeuge (developement tools) oder man lässt sich vom CrossCompiler (zB. avr-gcc) das erzeugte Assembler-Listing ausgeben und zählt wieder *'cycles'*:

Listing 13: AVR 'C' code meiISR1.c

```
/* 'testPkF1.c' XH@PkF/Nov15
 * demo programm: deviceType, ISR, sign-of-life
 * compile : 'avr-gcc -S testPkF1.c'
 * download: (use 'avrdude')
 * run     : (power up AM16 device)
 * stop    : (power off AM16 device)
 */
#define __AVR_ATmega16__
#include <avr/io.h>
#include <avr/interrupt.h>

ISR( INT0_vect ){ PORTB ^= (1<<1); }
ISR( TIMER0_OVF_vect ){ PORTB ^= (1<<2); }

int main(void){
  cli();
  DDRB  = (1<<0);
  PORTB = 0x01;
  /* INT0_enable() missing here! */
  /* Timer0_Init() missing here! */
  sei();
  for(;;){
    PORTB ^= (1<<0);
  }
}
```

Es ist zudem ganz interessant und **lehrreich**, sich gelegentlich den vom CrossCompiler erzeugten Assemblercode anzusehen (kotz):

'avr-gcc -S meiISR1.c'

Listing 14: 'C' -zu- Assemblercode meiISR1.s

```
        .file   "testPkF1.c"
__SREG__ = 0x3f
__SP_H__ = 0x3e
__SP_L__ = 0x3d
__CCP__  = 0x34
__tmp_reg__ = 0
__zero_reg__ = 1
        .global __do_copy_data
        .global __do_clear_bss
        .text
.global __vector_1
        .type   __vector_1, @function
__vector_1:
        push __zero_reg__
        push r0
        in r0,__SREG__
        push r0
        clr __zero_reg__
        push r24
        push r25
        push r26
        push r27
        push r30
        push r31
        push r29
        push r28
        in r28,__SP_L__
        in r29,__SP_H__
/* prologue: Signal */
/* frame size = 0 */
        ldi r26,lo8(56)
        ldi r27,hi8(56)
        ldi r30,lo8(56)
        ldi r31,hi8(56)
        ld r25,Z
        ldi r24,lo8(2)
        eor r24,r25
        st X,r24
/* epilogue start */
```

```
        pop r28
        pop r29
        pop r31
        pop r30
        pop r27
        pop r26
        pop r25
        pop r24
        pop r0
        out __SREG__,r0
        pop r0
        pop __zero_reg__
        reti
        .size   __vector_1, .-__vector_1
.global __vector_2
        .type   __vector_2, @function
__vector_2:
        push __zero_reg__
        push r0
        in r0,__SREG__
        push r0
        clr __zero_reg__
        push r24
        push r25
        push r26
        push r27
        push r30
        push r31
        push r29
        push r28
        in r28,__SP_L__
        in r29,__SP_H__
/* prologue: Signal */
/* frame size = 0 */
        ldi r26,lo8(56)
        ldi r27,hi8(56)
        ldi r30,lo8(56)
        ldi r31,hi8(56)
        ld r25,Z
        ldi r24,lo8(4)
        eor r24,r25
        st X,r24
/* epilogue start */
        pop r28
        pop r29
        pop r31
        pop r30
        pop r27
        pop r26
        pop r25
        pop r24
        pop r0
        out __SREG__,r0
        pop r0
        pop __zero_reg__
        reti
        .size   __vector_2, .-__vector_2
.global main
        .type   main, @function
main:
        push r29
        push r28
        in r28,__SP_L__
        in r29,__SP_H__
/* prologue: function */
/* frame size = 0 */
/* #APP */
 ;  16 "testPkF1.c" 1
        cli
 ;  0 "" 2
/* #NOAPP */
        ldi r30,lo8(55)
        ldi r31,hi8(55)
```

```
        ldi r24,lo8(1)
        st Z,r24
        ldi r30,lo8(56)
        ldi r31,hi8(56)
        ldi r24,lo8(1)
        st Z,r24
/* #APP */
 ;  19 "testPkF1.c" 1
        sei
 ;  0 "" 2
/* #NOAPP */
.L6:
        ldi r26,lo8(56)
        ldi r27,hi8(56)
        ldi r30,lo8(56)
        ldi r31,hi8(56)
        ld r25,Z
        ldi r24,lo8(1)
        eor r24,r25
        st X,r24
        rjmp .L6
        .size   main, .-main
```

Isses **länger** worn?
Mei, jooooooh . . . (Schmerz)!

Und, kennsch Di no aus?
. . . aehm, . . .

Was isch klarer, ASM oder Hochsprache?
grmmmblll

Was isch ez a 'Bissl' effizienter,
ASM oder Hochsprache? :-)) (ätsch)

→ Bittebitte lassz Enk nie kan 'Shas' einreden,

- selber nachschauen,
- selber checken!
- (bitte)

Danke!

21.6 watchdog timer

21.7 Real Application Example: 'RopeMeasurement' in C

Zugegeben, es ischa optischer Sauhaufen (mess-urement :-),
aber do siehgsch amol,
wie fein soa Chaos
zum Lesen isch:

21.8 AVR AM16 C-Application *'RopeMeasurement'*

Listing 15: RopeMeas.c

```
// state chart ?
// PinBelegung ?
// +Entprellung ?
// +ADLAR ?
// +gespeicherteWerteEinlesen/Schreiben(EEPROM)
// +AdcNullungsFunktion
// -Tarierung?
// +singleEnded
// autoPowerOff
// optische Struktur==Sauerei

// 51.5%pgm 75.7%data mit JeL1
// 48.2%pgm 62.2%data mit JeL3
// 48.2%pgm 62.2%data mit JeL4
// 49.8%pgm 63.0%data 7,8%eeprom mit JeL5
// 51.1%pgm 61.0%data 7,8%eeprom mit JeLC

/************************************************************
* Rope Measurement System - R.M.S.      HTL Diploma Project 2099
* Bemmler Mario                                  04-13-2099
* Last Update: 05-19-2099  15:00
************************************************************
* Update List:
*                               04-13-2099: MenuStruct (Pseudo), LCDNEU.c
*                                           LCDNEU.h
*                               04-14-2099: MenuStruct (Pseudo) new,ADC Sequence
*                               04-16-2099: PowerSupply (PortB0),
*                                           Timer + Interrupt,
*                                           ADC fixes: ADC.h, ADC.c
*                               04-28-2099: Drehencoder Interrupt Routine
*                                           Flashen (FEHLER BEI SIGNATURE)
*                                           Fehler Behoben
*                               05-01-2099: Menu Ausprogrammieren
*                                           Testen (Negative Kontrastspannung FEHLER)
*                               05-03-2099: Ausgang PC7 als Impulsgeber verwendet
*                               05-04-2099: Funktionen puls_init() und power_init()
*                               05-05-2099: LCD Kontrastspannung an PORTC7 ausgeben
*                               05-06-2099: LCD Initialisierung
*                               05-07-2099: LCD funktioniert
*                                           Timer - Interrupts Reperatur
*                               05-10-2099: PWM - Beginn
*                               05-11-2099: PWM - Fertig
*                               05-12-2099: Drehencoder Programmierung
*                               05-13-2099: MenuLogik (State Maschine)
*                               05-14-2099: Formel und Save Funktion
*                               05-15-2099: Formel
*                               05-19-2099: Formel und interner Operationsverstarker
*
*
* ToDo Liste:   - Formel
*
*===========================
* BELEGUNGSPLAN
*===========================
*
*PinB0 - Erhaltung          | ADC Eingang - PinA0
*PinB1 - Drehencoder PhaseA | FREI                - PinA1
*PinB2 - Drehencoder PhaseB | FREI                - PinA2
*PinB3 - Drehencoder Taster | FREI                - PinA3
*PinB4 - FREI               | FREI                - PinA4
*PinB5 - FREI               | FREI                - PinA5
*PinB6 - FREI               | FREI                - PinA6
*PinB7 - FREI               | FREI                - PinA7
*RESET -                    |                     - AREF
*Vcc   -                    |             - GND
*GND   -                    |             - AVcc
*XTAL1 - FREI               | FREI                - PinC7
*XTAL2 - FREI               | FREI                - PinC6
*PinD0 - FREI               | FREI                - PinC5
*PinB1 - RS                 | FREI                - PinC4
*PinB2 - EN                 | FREI                - PinC3
*PinB3 - DB                 | FREI                - PinC2
*PinB4 - DB                 | FREI                - PinC1
*PinB5 - DB                 | FREI                - PinC0
*PinB6 - DB                 | PWM - Port  - PinD7
*
************************************************************/


#define __AVR_ATmega16__        1       \\uC type
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/eeprom.h>
#include "LCDNEU.h"
//#include "ADC.h"
#include <avr/pgmspace.h>
#define E_E__P_R_O_M __attribute__ ((section(".eeprom")))

/*****************************************************
* DEFINE BLOCK
*****************************************************/
#define XTAL            8e6                     // 8MHz
//Externer Interrupt fr Taster
//#define INT0_vect          _VECTOR(2)
//Mittels PWM geloest
```

```
//Drehencoder Vector
//#define TIMER0_COMP_vect   _VECTOR(20)

//Timer/Counter0 Overflow Vector fr Interrupt definieren
//#define TIMER1_OVF_VECT    _VECTOR(9)

//Phasen fr Drehencoder definieren
//#define PHASE_A               (PINB & 1<<PB1)
//#define PHASE_B               (PINB & 1<<PB2)
//#define TASTER                (PINB & 1<<PB3)

lcd_string_pgm(void *p, int);
void lcd_string_p(char *s){lcd_string(s);}
void calc_and_store();

#define MAX(a,b) ( ((a)<(b)) ? (b):(a) )
#define MIN(a,b) ( ((a)<(b)) ? (a):(b) )
#define LIM(a,m) ( ((a)<0)? (0): ( ((a)>=(m))? (m-1):(a) ) )
#define LcdLINELEN 16
#define LcdSTRSIZE (LcdLINELEN+1)

enum SMstatsTyp { SMmain, SMcontrast, SMsynth, SMmetal, SMshow, SMnull, SMoff};
#define SMsize (SMoff+1)
enum SMstatsTyp SMstate =SMmain;
//const char PROGMEM *pMainMsg[ SMsize ] = {"Main","Contrast","Synth","Metal","Show","PowerDown...", "AdcNull"};
char *MainMsg[ SMsize ]  = {"Main","Contrast","Synth","Metal","Show","Nullung","PowerDown..."};

#define nSynthAnz 18
int8_t nSynthNr=0;
/*PGM_P char  *SynthMsg[ nSynthAnz ] PROGMEM = { "Exit",
  "8.0mm","8.5mm","9.0mm","9.5mm","10.0mm","10.5mm","11.0mm","11.5mm","12.0mm",
  "12.5mm","13.0mm","13.5mm","14.0mm","14.5mm","15.0mm","15.5mm","16.0mm"};
  */
char  *SynthMsg[ nSynthAnz ]  = { "Exit",
  "8.0mm","8.5mm","9.0mm","9.5mm","10.0mm","10.5mm","11.0mm","11.5mm","12.0mm",
  "12.5mm","13.0mm","13.5mm","14.0mm","14.5mm","15.0mm","15.5mm","16.0mm"};

#define nMetalAnz 19
int8_t nMetalNr=0;
/*prog_char *pMetalMsg[ nMetalAnz ] = { "Exit",
  "8.0mm","8.5mm","9.0mm","9.5mm","10.0mm","10.5mm","11.0mm","11.5mm","12.0mm",
  "12.5mm","13.0mm","13.5mm","14.0mm","14.5mm","15.0mm","15.5mm","16.0mm", "Nullung" };
 */
char  *MetalMsg[ nMetalAnz ]  =  { "Exit",
  "8.0mm","8.5mm","9.0mm","9.5mm","10.0mm","10.5mm","11.0mm","11.5mm","12.0mm",
  "12.5mm","13.0mm","13.5mm","14.0mm","14.5mm","15.0mm","15.5mm","16.0mm", "Nullung" };

#define ShowAnz 10
int8_t nShowNr=0;
char show_string[LcdSTRSIZE] = "Wert##=00.000 kN";
#define ShowIdxPos 4
#define ShowNumPos 7

char contrast_string[LcdSTRSIZE] = "Contrast: XXX ";
#define ContrastNumPos 10
#define nContrastAnz 255

uint8_t nContrastNr=0x080; //??

int8_t MDdebug=0;
int8_t encRotated=0;
int8_t enc_dir;
int8_t nMainMenuNr=0;
int iSVganze,iSVkomma;
float fStoredValues[ShowAnz]={
        0.0, 0.0, 0.0, 0.0, 0.0,
        0.0, 0.0, 0.0, 0.0, 0.0
        };
uint16_t uNullWert=26;
/*im EEPROM: */
float ee_fStoredValues[10] E_E__P_R_O_M;
uint16_t ee_uNullWert      E_E__P_R_O_M;

volatile uint8_t PbAlt;

/*********************************************
* MAIN BLOCK
*********************************************/
int main(void)
{
  uint8_t       firstLoop=1;
  uint8_t       PbXor;
  register      PbNeu;
  char          s2[9];

       DDRA  = 0;              //alles Input (ADC)
       PORTA = 0;              //Pullups ausschalten
       DDRB  = 0x01;           //PinB0 als Ausgang (POWER=ON) und high
       PORTB = 0x0F;           //Pull Ups einschalten
       DDRC  = 0x0F0;          //PORTC7 bis 4 ist OUT, checkout PINS
       PORTC = 0x0F0;
       //PORTD init ist im "lcd_init()":
       lcd_init();             //PortD init und LCD init; ACHTUNG: PD7 = LCD Kontrast
       lcd_string("StartUp...");

       PbAlt   = PINB;         //Encoder Initialisieren
       enc_dir = 1;

       sei();                  //Globale Interrupts aktivieren (what 4?)

       nMainMenuNr = SMstate = SMcontrast;
       MDdebug     = 0;
gespeicherteMesswerteEinlesen: /*10 Werte "fStoredValues[]" aus EEPROM*/
  eeprom_read_block((void*)(fStoredValues), &ee_fStoredValues, sizeof(float)*10);
  eeprom_read_block((void*)(uNullWert), &ee_uNullWert, sizeof(uNullWert));

MainLoop:
  for(;;){
    encRotated =0;
    PbXor      = (PbNeu=PINB) ^ PbAlt;
    PbAlt      = PbNeu;
    _delay_ms(1);
    if(PbXor & 0x08 & PbNeu){   //Taster aktiv
      PORTC ^= 0x20;
        /* (1) react to Encoder button press AND(!) release? */
      switch(SMstate){
```

```
      case SMmain:
        SMstate = nMainMenuNr;
        if(SMoff==nMainMenuNr){
          lcd_string("PowerDown...");
          /*schreib 10 messerte "fStoredValues[]" ins EEPROM*/
          eeprom_write_block((void*)(fStoredValues), &ee_fStoredValues, sizeof(float)*10);
          eeprom_write_block((void*)(uNullWert), &ee_uNullWert, sizeof(uNullWert));
          for(;;){PORTB = 0;} //hangs here 'forever' (until power is down)
        }
        if(SMnull==nMainMenuNr){
          SMstate=SMmetal; nMetalNr=18; calc_and_store();
          nMainMenuNr=SMstate=SMshow;
          nShowNr=0;
        }
        break;
      case SMsynth:
        if(nSynthNr){
          calc_and_store();
          nMainMenuNr=SMstate=SMshow;
          nShowNr=0;
        }else nMainMenuNr=SMstate=SMmain;
        break;
      case SMmetal:
        if(nMetalNr){
          calc_and_store();
          nMainMenuNr=SMstate=SMshow;
          nShowNr=0;
        }else nMainMenuNr=SMstate=SMmain;
        break;
      default://lcd_string("main:default");
      case SMshow:
      case SMcontrast:
        nMainMenuNr=SMstate=SMmain;
        break;
      }
    }else{
      if( PbXor & 0x02 & PbAlt ){ //drehencoder PhaseA verandert
        enc_dir    = (PbAlt & 0x04) ? 1 : -1;  //0x04...PhaseB (1 oder 0)
        encRotated = 1;
        if(enc_dir < 0){
          PORTC ^= 0x080;
          //lcd_string("DuHaschRecht");
        }else{
          PORTC ^= 0x040;
          //lcd_string("duLinkerHund");
        }
        /* (2) react to encoder rotation? */
        switch(SMstate){
        case SMmain:
          nMainMenuNr += enc_dir;
          nMainMenuNr =  LIM(nMainMenuNr, SMsize);
          break;
        case SMsynth:
          nSynthNr += enc_dir;
          nSynthNr =  LIM(nSynthNr, nSynthAnz);
          break;
        case SMmetal:
          nMetalNr += enc_dir;
          nMetalNr =  LIM(nMetalNr, nMetalAnz);
          break;
        case SMshow:
          nShowNr += enc_dir;
          nShowNr =  LIM(nShowNr, ShowAnz);
          break;
        case SMcontrast:
          if(nContrastNr<10) nContrastNr=10;
          if(nContrastNr>245) nContrastNr=245;
          nContrastNr += (10*enc_dir);
          OCR2=nContrastNr =  LIM(nContrastNr, nContrastAnz);
          break;
        default:        SMstate=SMmain;
          break;
        }//switch
      }//if
    }//else

        /* (3) update se Display */
    if( (PbXor & 0x08 /*Taster aktiv*/) || encRotated || firstLoop){
      switch(SMstate){
      case SMmain:    lcd_string_p(MainMsg[ nMainMenuNr ]); break;
      case SMsynth:   lcd_string_p(SynthMsg[ nSynthNr ]); break;
      case SMmetal:   lcd_string_p(MetalMsg[ nMetalNr ]); break;
      //case SMmetal:   sprintf(s2,('E'==*s)?"%s":"%6smm",MetalMsg[ nMetalNr ]);lcd_string_p(s2); break;
      case SMshow:
        /* 1. N-ten Wert aus fStoredValues[] in den "show_string" kopieren
         * 2. N in den ShowString schreiben */
        iSVganze= fStoredValues[nShowNr];
        iSVkomma= (fStoredValues[nShowNr]-iSVganze)*1000;
        snprintf(show_string+ShowIdxPos, LcdLINELEN-ShowIdxPos+1, "%02u=%02d.%03u kN", nShowNr, iSVganze, iSVkomma );
        //sprintf(show_string+ShowIdxPos, "%02u=%06.3f kN", nShowNr, fStoredValues[nShowNr] );
        lcd_string( show_string );
        break;
      case SMcontrast:
        sprintf(contrast_string+ContrastNumPos, "%03u", nContrastNr );
        lcd_string( contrast_string );
        break;
      default:      lcd_string("unknownState"); break;
      }\\switch
    }//if
    firstLoop=0;
  }/*end MainLoop*/
}

/*Interrupt Service Routines (ISR):*/
ISR(INT0_vect){}
ISR(INT1_vect){}
ISR(TIMER0_vect){/*Timer Interrupt Service Routine*/}



lcd_string_pgm(void *p,int nr){
/*
  int8_t i;
  void *s;
  char ch;
```

```
  s=p+nr;
  lcd_clear();
  for(i=0;(i<LcdSTRSIZE) && (ch=pgm_read_byte(s));i++,s++){
    lcd_data(ch);
  }
*/
}


void calc_and_store(){
/* 1.Messen
 * 2.Ausrechnen
 * 3.Speichern ins fStoredValues[]
 */
        #define ADchannel 0
        #define ACpre02 ((0<<ADPS0) | (0<<ADPS1) | (1<<ADPS2))
        #define ACpre04 ((0<<ADPS0) | (1<<ADPS1) | (0<<ADPS2))
        #define ACpre08 ((0<<ADPS0) | (1<<ADPS1) | (1<<ADPS2))
        #define ACpre16 ((1<<ADPS0) | (0<<ADPS1) | (0<<ADPS2))
        #define ACpre32 ((1<<ADPS0) | (0<<ADPS1) | (1<<ADPS2))
        #define ACpre64 ((1<<ADPS0) | (1<<ADPS1) | (0<<ADPS2))
        #define ACpre128 ((1<<ADPS0) | (1<<ADPS1) | (1<<ADPS2))
        #define ADref256 ((1<<REFS1)|(1<<REFS0))
        #define START_ADC (ADCSRA |= (1<<ADSC))
        #define WAIT4ADC  (ADCSRA & (1<<ADSC))
        //ADC1 = Positiv
        //ADC0 = Negativ
        //Gain = 200
        //ADLeft = AD Left Adjust Result
        #define ADLeft  (1 << ADLAR)
        #define ADGain200 (1 << MUX3 | 1 << MUX1 | 1 << MUX0)
  register int8_t       i;
  uint16_t      avg; //averaged ADconversionResult
  int16_t       adw;
  float         dResult;
//int8_t korr = 1; //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
#define KORR 0.008F

//double dMetalSeilDurch[] = {
#define dMetalSeilDurch01       0.269518909F
#define dMetalSeilDurch02       0.261729658F
#define dMetalSeilDurch03       0.253928271F
#define dMetalSeilDurch04       0.246115085F
#define dMetalSeilDurch05       0.238290441F
#define dMetalSeilDurch06       0.230454682F
#define dMetalSeilDurch07       0.222608153F
#define dMetalSeilDurch08       0.214751202F
#define dMetalSeilDurch09       0.206884179F
#define dMetalSeilDurch10       0.199007438F
#define dMetalSeilDurch11       0.191121333F
#define dMetalSeilDurch12       0.183226221F
#define dMetalSeilDurch13       0.175322461F
#define dMetalSeilDurch14       0.167410414F
#define dMetalSeilDurch15       0.159490445F
#define dMetalSeilDurch16       0.151562917F
#define dMetalSeilDurch17       0.143628197F

/*double dSynthSeilDurch[] = {
    0.315981002, 0.311164233, 0.306341909, 0.301514107,
    0.296680906, 0.291842384, 0.286998621, 0.282149696,
    0.277295689, 0.272436681, 0.267572753, 0.262703985,
    0.257830461, 0.252952261, 0.248069469, 0.243182168,
    0.238290441}; */
#define  dSynthSeilDurch01      0.315981002F
#define  dSynthSeilDurch02      0.311164233F
#define  dSynthSeilDurch03      0.306341909F
#define  dSynthSeilDurch04      0.301514107F
#define  dSynthSeilDurch05      0.296680906F
#define  dSynthSeilDurch06      0.291842384F
#define  dSynthSeilDurch07      0.286998621F
#define  dSynthSeilDurch08      0.282149696F
#define  dSynthSeilDurch09      0.277295689F
#define  dSynthSeilDurch10      0.272436681F
#define  dSynthSeilDurch11      0.267572753F
#define  dSynthSeilDurch12      0.262703985F
#define  dSynthSeilDurch13      0.257830461F
#define  dSynthSeilDurch14      0.252952261F
#define  dSynthSeilDurch15      0.248069469F
#define  dSynthSeilDurch16      0.243182168F
#define  dSynthSeilDurch17      0.238290441F

  //ADPS0-2(clock E(20..50kHz) = Prescaler=32=31.25kHz=1MHz/32), ADEN (Enable)
  ADCSRA=  ((1<<ADEN) | ACpre32);
  ADMUX = (ADchannel | ADref256 );

  //one throw-away-conversion:
  for( START_ADC ; WAIT4ADC ; ){};

  avg=0;
  for(i=0;i<16;i++){
   for( START_ADC ; WAIT4ADC ; ); //one conversion
                /*"read ADCL first"*/
   avg+= ((unsigned)ADCL);
   avg+= (((unsigned) ((unsigned)ADCH)<<8) );
  }

  avg = (avg >> 4); //div by 8
  adw = (avg<uNullWert)?0:(avg-uNullWert);

  /*diese Konstruktion spart SRAM: */
  if(SMsynth==SMstate){
   switch(nSynthNr){
        case  1  : dResult= (adw * KORR)/ dSynthSeilDurch01; break;
        case  2  : dResult= (adw * KORR)/ dSynthSeilDurch02; break;
        case  3  : dResult= (adw * KORR)/ dSynthSeilDurch03; break;
        case  4  : dResult= (adw * KORR)/ dSynthSeilDurch04; break;
        case  5  : dResult= (adw * KORR)/ dSynthSeilDurch05; break;
        case  6  : dResult= (adw * KORR)/ dSynthSeilDurch06; break;
        case  7  : dResult= (adw * KORR)/ dSynthSeilDurch07; break;
        case  8  : dResult= (adw * KORR)/ dSynthSeilDurch08; break;
        case  9  : dResult= (adw * KORR)/ dSynthSeilDurch09; break;
        case  10 : dResult= (adw * KORR)/ dSynthSeilDurch10; break;
        case  11 : dResult= (adw * KORR)/ dSynthSeilDurch11; break;
        case  12 : dResult= (adw * KORR)/ dSynthSeilDurch12; break;
        case  13 : dResult= (adw * KORR)/ dSynthSeilDurch13; break;
        case  14 : dResult= (adw * KORR)/ dSynthSeilDurch14; break;
```

```
        case  15 : dResult= (adw * KORR)/ dSynthSeilDurch15; break;
        case  16 : dResult= (adw * KORR)/ dSynthSeilDurch16; break;
        case  17 : dResult= (adw * KORR)/ dSynthSeilDurch17; break;
        default: dResult=0.0;
    }
   }else{
    switch(nMetalNr){
        case  1  : dResult= (adw * KORR)/ dMetalSeilDurch01; break;
        case  2  : dResult= (adw * KORR)/ dMetalSeilDurch02; break;
        case  3  : dResult= (adw * KORR)/ dMetalSeilDurch03; break;
        case  4  : dResult= (adw * KORR)/ dMetalSeilDurch04; break;
        case  5  : dResult= (adw * KORR)/ dMetalSeilDurch05; break;
        case  6  : dResult= (adw * KORR)/ dMetalSeilDurch06; break;
        case  7  : dResult= (adw * KORR)/ dMetalSeilDurch07; break;
        case  8  : dResult= (adw * KORR)/ dMetalSeilDurch08; break;
        case  9  : dResult= (adw * KORR)/ dMetalSeilDurch09; break;
        case  10 : dResult= (adw * KORR)/ dMetalSeilDurch10; break;
        case  11 : dResult= (adw * KORR)/ dMetalSeilDurch11; break;
        case  12 : dResult= (adw * KORR)/ dMetalSeilDurch12; break;
        case  13 : dResult= (adw * KORR)/ dMetalSeilDurch13; break;
        case  14 : dResult= (adw * KORR)/ dMetalSeilDurch14; break;
        case  15 : dResult= (adw * KORR)/ dMetalSeilDurch15; break;
        case  16 : dResult= (adw * KORR)/ dMetalSeilDurch16; break;
        case  17 : dResult= (adw * KORR)/ dMetalSeilDurch17; break;
        case  18 : dResult= avg/100.0F; uNullWert=avg; nMetalNr=0;break;
        default: dResult=0.0;
    }
   }
   /* store se new value into se array mit se last 10 values: */
   for(i=ShowAnz-1;i>0;i--) fStoredValues[i] = fStoredValues[i-1];
   fStoredValues[0] = dResult;

}//end: "calc_and_store()"
```

Listing 16: adc.c

```
/************************************************
adc.h:
AUTHOR: Thomas Gruebler
Version: 0.2
Beschreibung: Erfasst werte am ADC Port
************************************************/
#ifndef ADC_H
#define ADC_H

/* Hauptfunktionen */
void adc_init(void);            //Startet den ADC.
void adc_deinit(void);            //Stoppt de ADC.
uint8_t adcget(uint8_t kanal);        //Liest wert vom ADC ein.

/*Beschreibt aus wievielen wandlungen der Durchschnittswert gebildet
* werden sollte den adcget() zurueckgibt.
* Hoehere Zahl = Genauer, laengere Wandlungszeit
* Niedrigere Zahl = Nicht so genau, kurze Wandlungszeit.*/

#define wandlungen 4        //Standart: 4
//AREF=internal 2.56Vdc BandGap:
#define referenz ((1<<REFS1)|(1<<REFS0))
#endif //ADC_H


/************************************************
adc.c:
AUTHOR: Thomas Gruebler
Version: 0.2
Beschreibung: Erfasst werte am ADC Port
************************************************/

#include <avr/io.h>
#include "ADC.h"

#define

void adc_init(void) {
        //ADPS0-2(Vorteiler 128), ADEN (Enable)
        ADCSRA |= ((1<<ADPS0) | (1<<ADPS1) | (1<<ADPS2) | (1<<ADEN));

        //Referenz setzen
        ADMUX = 0;
        ADMUX |= referenz;

        //eine ADC-Wandlung als Dummy zum warmlaufen
        ADCSRA |= (1<<ADSC);
        while ( ADCSRA & (1<<ADSC) ); //Warten bis die Wandlung zu ende ist.
}

void adc_deinit(void) {
        ADCSRA &= ~(1<<ADEN);
}

uint8_t adcget(uint8_t kanal) {
        uint8_t i = 0;
        uint16_t ergebnis = 0;
```

```
        ADMUX = kanal;            // Kanal waehlen
        ADMUX |= referenz;           //Referenzspannung setzen

        while (i!=wandlungen) {        //Mehrere wandlungen und durchschnittswert ermitteln
                ADCSRA |= (1<<ADSC);        // eine Wandlung "single conversion"
                while ( ADCSRA & (1<<ADSC) );
                i++;
                ergebnis += ADC;
        }
        ergebnis = ergebnis/wandlungen;    //Durchschnittswert bilden

        return ergebnis;

}
```

Listing 17: adc.h

```
/************************************************
AUTHOR: Thomas Gruebler
Version: 0.2
Beschreibung: Erfasst werte am ADC Port
************************************************/
#ifndef ADC_H
#define ADC_H

/* Hauptfunktionen */
void adc_init(void);            //Startet den ADC.
void adc_deinit(void);            //Stoppt de ADC.
uint8_t adcget(uint8_t kanal);        //Liest wert vom ADC ein.

/*Beschreibt aus wievielen wandlungen der Durchschnittswert gebildet
* werden sollte den adcget() zurueckgibt.
* Hoehere Zahl = Genauer, laengere Wandlungszeit
* Niedrigere Zahl = Nicht so genau, kurze Wandlungszeit.*/

#define wandlungen 4        //Standart: 4
#define referenz ((1<<REFS1)|(1<<REFS0))
#endif //ADC_H
```

Listing 18: lcdneu.c

```
#include <avr/io.h>
#include "LCDNEU.h"
#include <util/delay.h>

// sendet ein Datenbyte an das LCD

void lcd_data(unsigned char ch)
{

   LCD_PORT =((ch & 0x0F0) >> 1) + 2;               // Data7..4 und RS
   lcd_enable();
   LCD_PORT =((ch & 0x0F) << 3) + 2;               // Data7..4 und RS
   lcd_enable();

   _delay_ms(1);
}

// sendet einen Befehl an das LCD

void lcd_command(unsigned char ch)
{
   LCD_PORT =((ch & 0x0F0) >> 1);               // Data7..4 und RS
   lcd_enable();
   LCD_PORT =((ch & 0x0F) << 3);               // Data7..4 und RS
   lcd_enable();

   _delay_ms(1);
}

// erzeugt den Enable-Puls
void lcd_enable(void)
{
   LCD_PORT |= 0x04;
   _delay_us(10);                  // kurze Pause
   PORTC ^= 0x80;
   PORTC ^= 0x80;
   LCD_PORT &= 0x0FB;
   _delay_us(10);
}

// Initialisierung:
// Muss ganz am Anfang des Programms aufgerufen werden.

void lcd_init(void)
```

```
{

        LCD_DDR = 0xFE;
   //OC2 Ausgang bereits in lcd_init() festgelegt
        TCNT2=0x00;
        OCR2=0x80;
        //Phase-korrekt PWM, non-inverting
        TCCR2 |= ((1<<FOC2)|(1<<COM21)|(1<<COM20)|(1<<WGM20)|(1<<CS20));

   _delay_ms(30);

   LCD_PORT = 0x10;
   lcd_enable();
    _delay_ms(5);

   LCD_PORT = 0x10;
   lcd_enable();
   _delay_ms(5);

   LCD_PORT = 0x00;
   lcd_enable();
   _delay_ms(50);


   LCD_PORT = 0x00;
   lcd_enable();
    _delay_ms(5);

   LCD_PORT = 0x60;
   lcd_enable();
   _delay_ms(5);


   LCD_PORT = 0x00;
   lcd_enable();
    _delay_ms(5);

   LCD_PORT = 0x08;
   lcd_enable();
   _delay_ms(5);


   LCD_PORT = 0x00;
   lcd_enable();
    _delay_ms(5);

   LCD_PORT = 0x10;
   lcd_enable();
   _delay_ms(5);


        lcd_clear();
}

// Sendet den Befehl zur Loeschung des Displays

void lcd_clear(void)
{
   lcd_command(CLEAR_DISPLAY);
   _delay_ms(65);
}

// Sendet den Befehl: Cursor Home

/*void lcd_home(void)
{
   lcd_command(CURSOR_HOME);
   _delay_ms(5);
}*/
//I BRAUCH JA EIGENTLICH KOAN CURSOR SETZEN ODA?
// setzt den Cursor in Zeile y (1..4) Spalte x (0..15)
/*
void set_cursor(uint8_t x, uint8_t y)
{
  uint8_t tmp;

  switch (y) {
    case 1: tmp=0x80+0x00+x; break;    // 1. Zeile
    case 2: tmp=0x80+0x40+x; break;    // 2. Zeile
    case 3: tmp=0x80+0x10+x; break;    // 3. Zeile
    case 4: tmp=0x80+0x50+x; break;    // 4. Zeile
    default: return;                   // fr den Fall einer falschen Zeile
  }
  lcd_command(tmp);
}*/

// Schreibt einen String auf das LCD
```

```
void lcd_string(char *data){
  register i;
  lcd_clear();
  for(i=0;*data && i<16;i++,data++) {
    lcd_data(*data);
  }
}
```

Listing 19: lcdneu.h

```
#include <avr/io.h>
#include <util/delay.h>

void lcd_data(unsigned char temp1);
void lcd_string(char *data);
void lcd_command(unsigned char temp1);
void lcd_enable(void);
void lcd_init(void);
void lcd_home(void);
void lcd_clear(void);
//void set_cursor(uint8_t x, uint8_t y);

// Hier die verwendete Taktfrequenz in Hz eintragen, wichtig!

#define F_CPU 1000000

// LCD Befehle

#define CLEAR_DISPLAY 0x01
#define CURSOR_HOME   0x02

// Pinbelegung fr das LCD, an verwendete Pins anpassen

#define LCD_PORT      PORTD
#define LCD_DDR       DDRD
#define LCD_RS        PD1
#define LCD_EN        PD2
#define LCD_DB4       PD3
#define LCD_DB5       PD4
#define LCD_DB6       PD5
#define LCD_DB7       PD6

// DB4 bis DB7 des LCD sind mit PD3 bis PD6 des AVR verbunden
```

21.9 AVR AM16 *'Adc-Modul'* zu *'RopeMeasurement'*

Das SourceCode Modul 'AdcModul.c'

- includiert *"ADC.h"*
- ist separat zu compilieren *'gcc -c AdcModul01.c'*
- hat also kein *'main()'*
- erfordert beim Hauptprogramm-Compilieren den Zusatz *'-o AdcModul.o*

Listing 20: ADC.h Header zu AdcModul.c

```
/* File 'ADC.h', Johnny Weissmueller, 99.Dez.3001
 * Version: v0.2
 * Erfasst werte am ADC Port
 * Header zu 'AdcModul.c'
 * 'AvgWeight':aus wievielen AdcConversions der return-Wert zu mitteln ist
 *      Hoeher Wert     = feine Abstufung, wenig Stoerung+Rauschen
 *      Niedrigere Zahl = kurze Wartezeit, hohe Sample-Zahl,
 ****************************************************************/
#ifndef ADC_H
#define ADC_H
#define __AVR_ATmega16__        1       //uC type

void    adc_init(void);                 //ADC init and start
void    adc_deinit(void);               //ADC stop
uint8_t adcget(uint8_t kanal);          //read ADC avergaged

#define AvgWeight       4               //default: 4
#define AdcRefSource    ((1<<REFS1)|(1<<REFS0)) //AREF=internal 2.56Vdc BandGap:
#endif //ADC_H
```

Listing 21: AdcModul.c

```
/* File 'AdcModul.c', Johnny Weissmueller, 99.Dez.3001
 * Version: v0.2
 * Erfasst ADC Port Werte
 * compile: avrgcc -c AdcModul.c
 * run    : --- (isa CodeModul ohne 'main')
```

```
 ****************************************/
#define __AVR_ATmega16__         1       //uC type
#include <avr/io.h>
#include "ADC.h"
  /*
   *we try to enhance ligibility
   *by using MACROs with proper names:
****************************************/
#define MacroInitADC()   {\
                         ADMUX = 0 | AdcrefSource;              //set ADC Reference Voltage     \
                         ADCSRA    = ((1<<ADPS0) | (1<<ADPS1) | (1<<ADPS2));                    \
                         //          ==================================--> prescaler:=128 \
                         (ADCSRA |=  (1<<ADEN));          //enable ADC                        \
                         //                       =====--> 'AD enable'                        \
        }
#define MacroStartADC()  {\
                          ADMUX =  kanal | AdcRefSource; //set 'kanal', ADC reference source \
                          ADCSRA |=  (1<<ADSC);          //ADSC == 'AD start conversion' \
        }
#define MacroStopADC()   {(ADCSRA &= ~(1<<ADSC));}
#define ADCisRunning()   {(ADCSRA &   (1<<ADSC));}
                //         ======         =====--> status bit 'ADC running'
                //           +-----------------> 'ADC status register A'
#define MacroDoOneADConversion(){\
                         MacroStartADC();
                         while(ADCSRA & (1<<ADSC)){};    //wait for ADC done             \
        }
/**********************end-of-macros***/


EMPTY_INTERRUPT(ADC_vect);      //keine ISR() erzeugen (nur 'RETI')

void adc_init(void) {/*
*********************/
  MacroInitADC();               //der Name ist zugleich Erklaerung
  MacroDoOneConversion();       //der Name ist zugleich Erklaerung
}

void adc_deinit(void){/*        //der Namen isnid vu mirXH
**********************/
  ADCSRA &= ~(1<<ADEN);         //ADEN=='AD enable'; '~'==invertieren
}


  /*
   * ADC lesen:
   */
uint8_t adcget(uint8_t kanal){/*
*******************************/
  uint8_t       i;
  uint16_t      summe=0;

    /*average some conversion results (Mittelwert):
    ***********************************************/
  for(i=0; i<AvgWeight; i++){
    MacroDoOneADConversion();
    summe += ADC;
  }
  return ((1.0*summe)/AvgWeight);               //compute average
}
```

22 IoT

Das Directory-Trennzeichen ist der **slash '/'** und nicht der backslash (das haben nur MS aus Urheberrechtsgründen beim Klauen des Directory-Prinzips von UNIX so geändert)

Dateibetrachter-Programme nennt man *'pager'* (zB. das *'less'*)

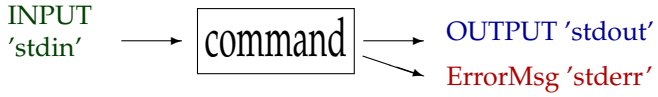
~, . und .. :

→ '~' ist das eigene user- home - Directory
(am Debian `/home/debian`, am Raspbian `/home/pi`, am Knoppix `/home/knoppix`, am obelix `/home/pupil/n17fffvv` usw.)

→ Der '.' ist das aktuelle Verzeichnis
(wo ich grade 'drin' bin — das von 'pwd' gemeldet wird)

→ '..' ist das übergeorndete Verzeichnis 'parent directory'
('cd ..' hebt mich *eine Stufe höher* im Dateibaum)

Viele Konsol-Kommandos arbeiten als *'Filter'*, d.h. sie
· lesen den Konsol-Input (Keyboard) und
· schreiben Resultate auf den Konsol-Output (Monitor).



Bsp:

```
more ..... listet seitenweise (weiter mit Leertaste, stop mit 'q')
grep x ... zeigt nur Zeilen die 'x' enthalten
tr x y ... ersetzt 'x' durch 'y'
sort ..... gibt den Input sortiert wieder aus
```

stdin ist der Konsol-Input als Datei.

stdout ist der Monitor-Output als Datei (Konsole, nicht GUI).
stdout kann man **umleiten (redirect):**

- in eine Datei (mit '>')
  (zB. 'ls -1 > /tmp/liste')
  oder
- an ein anderes Kommando (mit '|')
  (zB. 'du * | sort -n -r
  oder
  sudo iw wlan0 scan | grep -e SSID -e dBm
  -e : chan)

stderr ist Strom der Konsol-Fehlermeldungen (eines Kommandos) als Datei.

/tmp ist das (systemweite) Directory für *'temporary files'*-temporäre Dateien und ist *'world readable'* (alle dürfen Files anlegen, hineinschreiben/löschen aber nur, wer die Datei erzeugt hat - siehe *'sticky bit'*)

Das 'ding' möchti empfehlen: *'ding'* ist ein GUI deutsch/english Vokabel Dictionary — schau Dirs an.

22.1 basic Linux commands

abfragen		Rmk./Bsp.
id	wer bin i	ändern: su
pwd	wo bini	ändern: cd
ls	list directory contents	ls /proc
date	'display or set date/time'	setzen: ***sudo date 030716552018*** (Format: MMDDhhmmYYYY)
hwclock	RTC setzen	hwclock –systohc
df	'display free'	df -h
du	'display used'	du -s *
cat /proc/partitions	partitionen	
fdisk -l	partitionen	sudo fdisk -l
parted -l	partitionen	sudo parted -l
blkid	print block device attributes	
lsblk	block devices (partitions)	
lscpu	processors	
lshal	HAL hardware list	
lsinitramfs	list content of initramfs image	
lsmod	loaded kernel modules	
lsof	list open files	
lspci	list PCI devices	
lsusb	USB devices	
usb-devices	USB devices	mehr Detail
hwinfo	is hardcore!	hwinfo - -short, hwinfo - -cpu, hwinfo - -disk, hwinfo - -netcard, hwinfo - -wlan, hwinfo - -partition, hwinfo - -help
help	f. *builtin commands*	
apropos <Stichwort>	Programme finden	apropos terminal
man -k <Stichwort>	Programme finden	man -k terminal
man <chp> <item>	manual page zu <item> <chp>...chapter 1, 2, 3, ... 8	man 2 open
man man	manpage zu manpages	
man bash	-"- zum command interpreter	
verlassen:		
man	mit 'q' (quit)	
less	mit 'q'	
vi	mit 'ESC' ':' 'q' '!'	
any program	mit 'q' oder mit Strg+C (quit), (cancel)	

22.2 etwas Linux für Files u.Directories

nicht LÖSCHEN!

kein WIEDERHERSTELLEN

ACHTUNG!	die (Linux-) ext-Filesysteme haben keinen *Papierkorb* zur Aufbewahrung von Müll, es gibt auch **kein *Wiederherstellen*!** → Wer seine Files behalten möchte, soll sie **nicht *löschen*!**

22.3 div. utilities

Ma kun installierte Programme einfach mit
'ls /usr/bin/ | less'*
auflisten (sein viele!) Im *'/usr/bin/'* sein alles user applications, CLI und GUI utils gmixt.

```
'|' ....... (pipe) a command used to redirect data into an application.
'pipe' = Datenweiterleitung von Programm A an Programm B
```

zB. *'apropos file | grep utility'*
oder *'apropos file | grep utility |less'*
oder *'cat /dev/urandom | hexdump'*
oder *'echo "0.775*sqrt(2)" | bc -l'*
'>' ..... redirects output data into a file.
zB. *'echo "Hallo"> myFile1'*
'>>' ..... appends output data into a file.
zB. *'echo "10.10.63.61 ox"> /etc/hosts'*
'<' ..... takes input data from a file.
zB. *'netcat obelix.htl 59717 < dings.txt'*
(sendet 'dings.txt' via LAN)

aplay+play audio player
apropos ... help-Uebersicht
arandr .... GUI display/monitor settings (cf. xrandr)
apt-get ... the command used for installing, removing, and finding software
for Debian Linux systems.
sudo apt-get install puredata ... installs the Pure Data program and any dependencies.
sudo apt-get remove puredata ... will remove the program.
sudo apt-cache search image ... will search apt repositories for the keyword search. And so on.

bc ..... CLI Taschenrechner, zB. 'echo "sqrt(2)"|bc -l',
'bc -l' (Ende mit 'quit')

cat ..... "copy and type" = Inhalt auflisten
zB. 'cat /proc/cpuinfo', 'cat /proc/partitions'
..... concatenate. *XH:falsch! Soll 'Copy And Type' hoassn. (Autor isa Pfeifm)*
used to append data to a file. Ex: cat "Last line of text" > sometext.txt.
Merge files: cat append.txt >> main.txt will append all the text in append.txt
into main.txt.
cd ..... change directory. open a folder. ex:
cd ~/Pictures changes your current directory to the home Pictures folder,
so you can easily access the files within.
cp ..... copy file
..... copy a file from one place to another.
Ex: cp this.one this_01.one will copy this.one to another file this_01.one.
Add directories for more fun:
cp ~/Pictures/Vacation/saturn.jpg /Users/otherone/Pictures/Vacation/saturn.jpg.
chmod ..... change mode.
Used for file permissions, which can be important when sharing things on the network,
scripting actions, and many more reasons.

date ..... display or set date/time
df -h ..... display free on Datentraeger
diff ..... files vergleichen
du -s ..... display used space (belegter Speicherbedarf)

echo ..... gibt sein eigenes Argument aus, zB "echo $PATH"
exit ..... Linux/Unix-Logout

file ..... file type info
find ..... look up files in the filesystem.
Ex: *'find ~/Documents -name particular.txt -type f'* will look for the file
with the name particular.txt in the Documents directory.

```
find -H . -newermt 2020-05-13 -name "*" -type f -ls
```

```
find -L /SMUE/KWx/latex -newermt 2020-05-01 -name "*" -type f -ls 2>/dev/null  |grep "Č"
```

```
finger .... user information lookup
flock ..... manage file locks
free ...... freies memory abfragen

gcc ....... C Compiler ('Gnu C Compiler')
g++ ....... derselbe Compiler im C++ Modus
gdb ....... GNU debugger
gimp ...... Bildbearbeitung im GUI
gpg ....... GNU PGP encrypt + sign
grep ...... feines Filter-Programm zum Suchen in files:
            a tool used for searching through files.
            It's quite deep and can be complicated, but if you see the word grep in some command,
            you know it's searching for a match.
pdfgrep.... find text in PDF files

htop ...... display the processes currently alive on the CPU.
            If things seem slow, or you want to see how much CPU or memory a program is using,
            just type htop to see a table of all running processes,
            then type q when you want to exit.
hwinfo .... hardware info, zB.
            hwinfo --short --device, hwinfo --short --usb

id ........ wer bin ich: meine user-ID
ip ........ Info zu LAN-Device und -Config
ip a ...... IP address
ip li ..... MAC adress
ip nei .... ARPtable
ip rou .... routing tables
ip tun .... 'ip tunnels' 'legen'
info ...... see 'info' pages (verhungerte man Alternative)

kpartx .... images von partitions als /dev/... bereitstellen

less ...... Datei-Betrachter ("pager")
less ...... makes it so you can paginate and read a text file.
            Ex: less longtext.txt will fill the screen with the first part of the longtext.txt fil
            Use the space bar to view the next page. Type q to exit.
ls ........ list files (wie "DIR" im Wifdofs-cmd-DOSfenster)
ls -l ..... long format file list
ls ........ list files in the current directory
            Some options are ls -l to list in long format to provide information about permissions
            ls -a to show hidden files that start with the . character.
lsusb ..... list USB devices

man ....... "manual pages" = help (Tip: "man man"!)
mcedit x.c  C source code file [x.c] editieren (mcedit-Editor)
mkdir ..... make directory. create a folder.
            ex: mkdir Vacation makes a folder named Vacation in the current directory.
            mkdir ~/Pictures/Vacation makes a Vacation folder in the home Pictures directory.
more ...... seitenweise anzeigen (Leertaste, 'q')
mount ..... Laufwerke/Drives einbinden
            'mount -o loop <devicefile> <mountpoint>' ... DriveImages einbinden
mv ........ "move" = verschieben und umtaufen
mv ........ move a file from one directory to another, or to give it a new name.
            Ex: mv this.one that.one renames a file.
            mv this.one ~/Pictures/Vacation/ puts the file this.one into the Vacation directory.

nano x.c    C source code file [x.c] editieren (nano-Editor)
nano ...... a text editor.
            You'll often see commands that call nano so you can edit a configuration.
```

```
            Ex: nano /etc/avahi/services/afpd.service to edit the avahi Apple file service file.
ncal ...... calendar info
netcat .... Umleitung via Netzwerk
netdiscover ARP-Netzwerk-Scan
netstat ... print network info (-r:routes, -t:TCP, -u:UDP, -i:interfaces ...)
ss ........ wie 'netstat', aber mehr
nice ...... set process priority

ps ........ process status - Prozessliste
ps afx .... process status - detaillierte Prozessliste
      ps -ek etimes -o pid,tid,cputime,pcpu,etimes,policy,stat,sz,vsz,wchan,comm,cmd
pwd ...    present working directory.
            In case you forget where you are. Not much to it: pwd will output the directory name
            such as /Users/home/chip/Pictures/Vacation/

rm ........ remove a file. delete it, and beware!.
            Use the -r to make it recursive to delete a directory.
            Ex: rm this.one deletes that file. rm -r ~/Pictures/Vacation to forget the good times.
rsync ..... prima Kopier- u.Archivierungsprogramm (prima!)

set ....... gibt das "environment" aus
ssh ....... "secure shell" = remote login terminal, chiffriert
scp ....... secure copy.
            copy a file from one computer to another over a network.
            Ex: scp Pictures/Vacation/motel.jpg Pictures/Vacation/accident.jpg
            chip@otherchip.local:~/Pictures
            copies a couple jpegs to another computer on the network.
ssh ....... secure shell.
            access another computer on the network and use the terminal commands to make
            changes and control it.
            Ex: ssh chip@chip.local to access your CHIP on a local network.
            (bessr: '...to log into machine >chip.local< as user >chip<'
sshfs ..... "secure shell file system" = mount remote directories (prima!)
stat ...... file status, Rechte, Locks
sudo x y .. execute command 'y' as user 'x'
sudo - x y  =execute command 'y' as user 'x' within environment of 'x'
sudo ...... super user do.
            many commands need administrator-like privileges, otherwise they won't work.
            apt-get is a command that needs to be run with sudo to allow files to be written
            to protected directories. You'll see sudo as the first word in a lot of commands
            - all it is doing is giving the command the necessary access.
            You'll be asked for a password the first time you use sudo.

tar ....... Kompressions-Archiv-Tool (-> .tgz), f.Copy u.Backup, sehr vielseitig
test ...... test for condition (file types, value comparison) in shell-scripts
tex ....... TeX text processing, -> LaTeX Text-Satz-System
time y .... execute 'y' with profiling time performance
top ....... display OS processes

uptime .... OS-Betriebsdauer und load average (loadavg) abfragen
usb-devices ... USB report

valgrind .. program profiler
vi x.c     C source code file x.c editieren (insider-Editor; NICHT empfohlen)

vi verlassen: ESC ':' 'q' '!'

w  ........ wer is eingelogt
wc ........ word +line +byte count,
            zB 'ls -1aR ~ |grep -v / |grep -v \^\$ | wc'
```

```
xrandr .... CLI: display settings (zB 'xrandr -o left')
xterm ..... GUI terminal emulator (f. console)

zip, gzip, unzip, unrar
```

22.4 bearige utilities

convert [input-option] <input-file> [output-option] <output-file>

convert between image formats as well as resize an image, blur, crop, despeckle, dither, draw on, flip, join, re-sample, and much more.
konvertiert Bildformate zB. "convert dings.jpg dings.png"

dd 'disk double',

[...] "copies a file (from standard input to standard output, by default) with a changeable I/O block size, while optionally performing conversions on it. [...]

zB. `'dd if=/dev/sdc of=/dev/sdd'`
kopiert den ganzen Drive 'sdc' nach Drive 'sdd'

find

find -L -name "PMevalKdT1.pdf" -type f 2> /dev/null

ip

zB. *ip a* oder *ip ro* oder *ip link*

lsof

mc 'midnight commander'

minicom

netcat

netstat

owncloud(1), owncloudcmd(1)

ding ist ein GUI deutsch/english Vokabel Dictionary

pdflatex

rsync

ssh, sshfs

wine

22.4.1 bash - shell

Listing 22: *'bash'* utility

```
Terminal Tricks: Using Ctrl keys
Ctrl + n : same as Down arrow.
Ctrl + p : same as Up arrow.
Ctrl + r : begins a backward search through command history.(keep pressing Ctrl + r to move backward)
Ctrl + s : to stop output to terminal.
Ctrl + q : to resume output to terminal after Ctrl + s.
Ctrl + a : move to the beginning of line.
Ctrl + e : move to the end of line.
Ctrl + d : if you've type something, Ctrl + d deletes the character under the cursor, else, it escapes the
    current shell.
Ctrl + k : delete all text from the cursor to the end of line.
Ctrl + x + backspace : delete all text from the beginning of line to the cursor.
Ctrl + t : transpose the character before the cursor with the one under the cursor, press Esc + t to transposes
    the two words before the cursor.
Ctrl + w : cut the word before the cursor; then Ctrl + y paste it
Ctrl + u : cut the line before the cursor; then Ctrl + y paste it
Ctrl + _ : undo typing.
Ctrl + l : equivalent to clear.
Ctrl + x + Ctrl + e : launch editor defined by $EDITOR to input your command. Useful for multi-line commands.

Change case
Esc + u
# converts text from cursor to the end of the word to uppercase.

Esc + l
# converts text from cursor to the end of the word to lowercase.

Esc + c
# converts letter under the cursor to uppercase, rest of the word to lowercase.

Run history number (e.g. 53)
!53

Run last command
!!

# run the previous command using sudo
sudo !!
# of course you need to enter your password

Run last command and change some parameter using caret substitution (e.g. last command: echo 'aaa' -> rerun as:
    echo 'bbb')
#last command: echo 'aaa'
^aaa^bbb
#echo 'bbb'
#bbb
#Notice that only the first aaa will be replaced, if you want to replace all 'aaa', use ':&' to repeat it:
^aaa^bbb^:&
#or
!!:gs/aaa/bbb/

Run past command that began with (e.g. cat filename)
!cat
# or
!c
# run cat filename again

Bash globbing:
# '*' serves as a "wild card" for filename expansion.
/etc/pa*wd    #/etc/passwd

# '?' serves as a single-character "wild card" for filename expansion.
/b?n/?at      #/bin/cat

# '[]' serves to match the character from a range.
ls -l [a-z]*   #list all files with alphabet in its filename.

# '{}' can be used to match filenames with more than one patterns
ls {*.sh,*.py}   #list all .sh and .py files
Some handy environment variables
$0   :name of shell or shell script.
$1, $2, $3, ... :positional parameters.
$#   :number of positional parameters.
$?   :most recent foreground pipeline exit status.
$-   :current options set for the shell.
$$   :pid of the current shell (not subshell).
$!   :is the PID of the most recent background command.

$DESKTOP_SESSION     current display manager
$EDITOR   preferred text editor.
$LANG   current language.
$PATH   list of directories to search for executable files (i.e. ready-to-run programs)
```

```
$PWD     current directory
$SHELL   current shell
$USER    current username
$HOSTNAME    current hostname


Variable:
---------

Variable substitution within quotes:
# foo=bar
 echo "'$foo'"
#'bar'
# double/single quotes around single quotes make the inner single quotes expand variables

Get the length of variable
var="some string"
echo ${#var}
# 11

Get the first character of the variable
var=string
echo "${var:0:1}"
#s
# or
echo ${var%%"${var#?}"}

Remove the first or last string from variable
var="some string"
echo ${var:2}
#me string

Replacement (e.g. remove the first leading 0 )
var="0050"
echo ${var[@]#0}
#050

Replacement (e.g. replace 'a' with ',')
{var/a/,}

Replace all (e.g. replace all 'a' with ',')
{var//a/,}

#with grep
 test="god the father"
 grep ${test// /\\\|} file.txt
 # turning the space into 'or' (\|) in grep

To change the case of the string stored in the variable to lowercase (Parameter Expansion)
var=HelloWorld
echo ${var,,}
helloworld

Expand and then execute variable/argument
cmd="bar=foo"
eval "$cmd"
echo "$bar" # foo


Math:
-----

Arithmetic Expansion in Bash (Operators: +, -, *, /, %, etc)
echo $(( 10 + 5 ))  #15
x=1
echo $(( x++ )) #1 , notice that it is still 1, since it's post-incremen
echo $(( x++ )) #2
echo $(( ++x )) #4 , notice that it is not 3 since it's pre-incremen
echo $(( x-- )) #4
echo $(( x-- )) #3
echo $(( --x )) #1
x=2
y=3
echo $(( x ** y )) #8

Print out the prime factors of a number (e.g. 50)
factor 50
# 50: 2 5 5

Sum up input list (e.g. seq 10)
seq 10|paste -sd+|bc

Sum up a file (each line in file contains only one number)
awk '{s+=$1} END {print s}' filename

Column subtraction
cat file| awk -F '\t' 'BEGIN {SUM=0}{SUM+=$3-$2}END{print SUM}'
```

```
Simple math with expr
expr 10+20 #30
expr 10\*20 #600
expr 30 \> 20 #1 (true)

More math with bc
# Number of decimal digit/ significant figure
echo "scale=2;2/3" | bc
#.66

# Exponent operator
echo "10^2" | bc
#100

# Using variables
echo "var=5;--var"| bc
#4
```

22.4.2 `grep` - global regular expression print

Listing 23: *'grep'* utility

```
Grep:
-----

Type of grep
grep = grep -G # Basic Regular Expression (BRE)
fgrep = grep -F # fixed text, ignoring meta-charachetrs
egrep = grep -E # Extended Regular Expression (ERE)
pgrep = grep -P # Perl Compatible Regular Expressions (PCRE)
rgrep = grep -r # recursive

Grep and count number of empty lines
grep -c "^$"

Grep and return only integer
grep -o '[0-9]*'
#or
grep -oP '\d'

Grep integer with certain number of digits (e.g. 3)
grep '[0-9]\{3\}'
# or
grep -E '[0-9]{3}'
# or
grep -P '\d{3}'

Grep only IP address
grep -Eo '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
# or
grep -Po '\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'

Grep whole word (e.g. 'target')
grep -w 'target'
#or using RE
grep '\btarget\b'

Grep returning lines before and after match (e.g. 'bbo')
# return also 3 lines after match
grep -A 3 'bbo'

# return also 3 lines before match
grep -B 3 'bbo'

# return also 3 lines before and after match
grep -C 3 'bbo'

Grep string starting with (e.g. 'S')
grep -o 'S.*'

Extract text between words (e.g. w1,w2)
grep -o -P '(?<=w1).*(?=w2)'

Grep lines without word (e.g. 'bbo')
grep -v bbo filename

Grep lines not begin with string (e.g. #)
grep -v '^#' file.txt

Grep variables with space within it (e.g. myvar="some strings")
grep "$myvar" filename
#remember to quote the variable!
```

```
Grep only one/first match (e.g. 'bbo')
grep -m 1 bbo filename

Grep and return number of matching line(e.g. 'bbo')
grep -c bbo filename

Count occurrence (e.g. three times a line count three times)
grep -o bbo filename |wc -l

Case insensitive grep (e.g. 'bbo'/'BBO'/'Bbo')
grep -i "bbo" filename

COLOR the match (e.g. 'bbo')!
grep --color bbo filename

Grep search all files in a directory(e.g. 'bbo')
grep -R bbo /path/to/directory
# or
grep -r bbo /path/to/directory

Search all files in directory, do not ouput the filenames (e.g. 'bbo')
grep -rh bbo /path/to/directory

Search all files in directory, output ONLY the filenames with matches(e.g. 'bbo')
grep -rl bbo /path/to/directory

Grep OR (e.g. A or B or C or D)
grep 'A\|B\|C\|D'

Grep AND (e.g. A and B)
grep 'A.*B'

Regex any singer character (e.g. ACB or AEB)
grep 'A.B'

Regex with or without a certain character (e.g. color or colour)
grep 'colou?r'

Grep all content of a fileA from fileB
grep -f fileA fileB

Grep a tab
grep $'\t'

Grep variable from variable
$echo "$long_str"|grep -q "$short_str"
if [ $? -eq 0 ]; then echo 'found'; fi
#grep -q will output 0 if match found
#remember to add space between []!

Grep strings between a bracket()
grep -oP '\(\K[^\)]+'

Grep number of characters with known strings in between(e.g. AAEL000001-RA)
grep -o -w "\w\{10\}\-R\w\{1\}"
# \w word character [0-9a-zA-Z_] \W not word character

Skip directory (e.g. 'bbo')
grep -d skip 'bbo' /path/to/files/*
```

22.4.3 sed - stream editor

Listing 24: *'sed'* utility

```
Sed:
----

Remove the 1st line
sed 1d filename

Remove the first 100 lines (remove line 1-100)
sed 1,100d filename

Remove lines with string (e.g. 'bbo')
sed "/bbo/d" filename
# case insensitive:
sed "/bbo/Id" filename

Remove lines whose nth character not equal to a value (e.g. 5th character not equal to 2)
sed -E '/^.{5}[^2]/d'
#aaaa2aaa (you can stay)
#aaaa1aaa (delete!)

Edit infile (edit and save to file), (e.g. deleting the lines with 'bbo' and save to file)
```

```
sed -i "/bbo/d" filename

When using variable (e.g. $i), use double quotes " "
# e.g. add >$i to the first line (to make a bioinformatics FASTA file)
sed "1i >$i"
# notice the double quotes! in other examples, you can use a single quote, but here, no way!
# '1i' means insert to first line

Using environment variable and end-of-line pattern at the same time.
# Use backslash for end-of-line $ pattern, and double quotes for expressing the variable
sed -e "\$s/\$/\n+--$3-----+/"

Delete/remove empty lines
sed '/^\s*$/d'
# or
sed '/^$/d'

Delete/remove last line
sed '$d'

Delete/remove last character from end of file
sed -i '$ s/.$//' filename

Add string to beginning of file (e.g. "[")
sed -i '1s/^/[/' file

Add string at certain line number (e.g. add 'something' to line 1 and line 3)
sed -e '1isomething -e '3isomething'

Add string to end of file (e.g. "]")
sed '$s/$/]/' filename

Add newline to the end
sed '$a\'

Add string to beginning of every line (e.g. 'bbo')
sed -e 's/^/bbo/' file

Add string to end of each line (e.g. "}")
sed -e 's/$/\}\]/' filename

Add \n every nth character (e.g. every 4th character)
sed 's/.\{4\}/&\n/g'

Concatenate/combine/join files with a seperator and next line (e.g separate by ",")
sed -s '$a,' *.json > all.json

Substitution (e.g. replace A by B)
sed 's/A/B/g' filename

Substitution with wildcard (e.g. replace a line start with aaa= by aaa=/my/new/path)
sed "s/aaa=.*/aaa=\/my\/new\/path/g"

Select lines start with string (e.g. 'bbo')
sed -n '/^@S/p'

Delete lines with string (e.g. 'bbo')
sed '/bbo/d' filename

Print/get/trim a range of line (e.g. line 500-5000)
sed -n 500,5000p filename

Print every nth lines
sed -n '0~3p' filename
# catch 0: start; 3: step

Print every odd # lines
sed -n '1~2p'

Print every third line including the first line
sed -n '1p;0~3p'

Remove leading whitespace and tabs
sed -e 's/^[ \t]*//'
# Notice a whitespace before '\t'!!

Remove only leading whitespace
sed 's/ *//'
# notice a whitespace before '*'!!

Remove ending commas
sed 's/,$//g'

Add a column to the end
sed "s/$/\t$i/"
# $i is the valuable you want to add
```

```
# To add the filename to every last column of the file
for i in $(ls);do sed -i "s/$/\t$i/" $i;done

Add extension of filename to last column
for i in T000086_1.02.n T000086_1.02.p;do sed "s/$/\t${i/*./}/" $i;done >T000086_1.02.np

Remove newline\ nextline
sed ':a;N;$!ba;s/\n//g'

Print a particular line (e.g. 123th line)
sed -n -e '123p'

Print a number of lines (e.g. line 10th to line 33 rd)
sed -n '10,33p' <filename

Change delimiter
sed 's=/=\\/=g'

Replace with wildcard (e.g A-1-e or A-2-e or A-3-e....)
sed 's/A-.*-e//g' filename

Remove last character of file
sed '$ s/.$//'

Insert character at specified position of file (e.g. AAAAAA --> AAA#AAA)
sed -r -e 's/^.{3}/&#/' file
```

22.4.4 awk - Aho, Weinberger, Kernighan

Listing 25: *'awk'* utility

```
Awk:
----

Set tab as field separator
awk -F $'\t'

Output as tab separated (also as field separator)
awk -v OFS='\t'

Pass variable
a=bbo;b=obb;
awk -v a="$a" -v b="$b" "$1==a && $10=b" filename

Print line number and number of characters on each line
awk '{print NR,length($0);}' filename

Find number of columns
awk '{print NF}'

Reverse column order
awk '{print $2, $1}'

Check if there is a comma in a column (e.g. column $1)
awk '$1~/,/ {print}'

Split and do for loop
awk '{split($2, a,",");for (i in a) print $1"\t"a[i]}' filename

Print all lines before nth occurrence of a string (e.g stop print lines when 'bbo' appears 7 times)
awk -v N=7 '{print}/bbo/&& --N<=0 {exit}'

Print filename and last line of all files in directory
ls|xargs -n1 -I file awk '{s=$0};END{print FILENAME,s}' file

Add string to the beginning of a column (e.g add "chr" to column $3)
awk 'BEGIN{OFS="\t"}$3="chr"$3'

Remove lines with string (e.g. 'bbo')
awk '!/bbo/' file

Remove last column
awk 'NF{NF-=1};1' file

Usage and meaning of NR and FNR
# For example there are two files:
# fileA:
# a
# b
# c
# fileB:
# d
# e
awk 'print FILENAME, NR,FNR,$0}' fileA fileB
```

```
# fileA    1    1    a
# fileA    2    2    b
# fileA    3    3    c
# fileB    4    1    d
# fileB    5    2    e

AND gate
# For example there are two files:
# fileA:
# 1    0
# 2    1
# 3    1
# 4    0
# fileB:
# 1    0
# 2    1
# 3    0
# 4    1
awk -v OFS='\t' 'NR=FNR{a[$1]=$2;next} NF {print $1,((a[$1]=$2)? $2:"0")}' fileA fileB
# 1    0
# 2    1
# 3    0
# 4    0

Round all numbers of file (e.g. 2 significant figure)
awk '{while (match($0, /[0-9]+\[0-9]+/)){
    \printf "%s%.2f", substr($0,0,RSTART-1),substr($0,RSTART,RLENGTH)
    \$0=substr($0, RSTART+RLENGTH)
    \}
    \print
    \}'

Give number/index to every row
awk '{printf("%s\t%s\n",NR,$0)}'

Break combine column data into rows
# For example, seperate the following content:
# David    cat,dog
# into
# David    cat
# David    dog
awk '{split($2,a,",");for(i in a)print $1"\t"a[i]}' file
# Detail here:
        http://stackoverflow.com/questions/33408762/bash-turning-single-comma-separated-column-into-multi-line-
            string

Average a file (each line in file contains only one number)
awk '{s+=$1}END{print s/NR}'

Print field start with string (e.g Linux)
awk '$1 ~ /^Linux/'

Sort a row (e.g. 1 40 35 12 23 --> 1 12 23 35 40)
awk ' {split( $0, a, "\t" ); asort( a ); for( i = 1; i <= length(a); i++ ) printf( "%s\t", a[i] ); printf( "\n" )
    ; }'

Subtract previous row values (add column6 which equal to column4 minus last column5)
awk '{$6 = $4 - prev5; prev5 = $5; print;}'
```

22.4.5 `xargs` - build and execute command lines

Listing 26: *'xargs'* utility

```
Xargs:
------

Set tab as delimiter (default:space)
xargs -d\t

Prompt commands before running commands
ls|xargs -L1 -p head

Display 3 items per line
echo 1 2 3 4 5 6| xargs -n 3
# 1 2 3
# 4 5 6

Prompt before execution
echo a b c |xargs -p -n 3

Print command along with output
xargs -t abcd
# bin/echo abcd
# abcd
```

```
With find and rm
find . -name "*.html"|xargs rm

# when using a backtick
rm `find . -name "*.html"`

Delete files with whitespace in filename (e.g. "hello 2001")
find . -name "*.c" -print0|xargs -0 rm -rf

Show limits on command-line length
xargs --show-limits
# Output from my Ubuntu:
# Your environment variables take up 3653 bytes
# POSIX upper limit on argument length (this system): 2091451
# POSIX smallest allowable upper limit on argument length (all systems): 4096
# Maximum length of command we could actually use: 2087798
# Size of command buffer we are actually using: 131072
# Maximum parallelism (--max-procs must be no greater): 2147483647

Move files to folder
find . -name "*.bak" -print 0|xargs -0 -I {} mv {} ~/old
# or
find . -name "*.bak" -print 0|xargs -0 -I file mv file ~/old

Move first 100th files to a directory (e.g. d1)
ls |head -100|xargs -I {} mv {} d1

Parallel
time echo {1..5} |xargs -n 1 -P 5 sleep

# a lot faster than:
time echo {1..5} |xargs -n1 sleep

Copy all files from A to B
find /dir/to/A -type f -name "*.py" -print 0| xargs -0 -r -I file cp -v -p file --target-directory=/path/to/B
# v: verbose|
# p: keep detail (e.g. owner)

With sed
ls |xargs -n1 -I file sed -i '/^Pos/d' file

Add the file name to the first line of file
ls |sed 's/.txt//g'|xargs -n1 -I file sed -i -e '1 i\>file\' file.txt

Count all files
ls |xargs -n1 wc -l

Turn output into a single line
ls -1| xargs

Count files within directories
echo mso{1..8}|xargs -n1 bash -c 'echo -n "$1:"; ls -la "$1"| grep -w 74 |wc -l' --
# "--" signals the end of options and display further option processing

Count lines in all file, also count total lines
ls|xargs wc -l

Xargs and grep
cat grep_list |xargs -I{} grep {} filename
Xargs and sed (replace all old ip address with new ip address under /etc directory)
grep -rl '192.168.1.111' /etc | xargs sed -i 's/192.168.1.111/192.168.2.111/g'
```

22.4.6 if statement

Listing 27: *'if'* statement

```
If statement:
-------------

# if and else loop for string matching
if [[ "$c" == "read" ]]; then outputdir="seq"; else outputdir="write" ; fi

# Test if myfile contains the string 'test':
if grep -q hello myfile; then echo -e "file contains the string!" ; fi

# Test if mydir is a directory, change to it and do other stuff:
if cd mydir; then
  echo 'some content' >myfile
else
  echo >&2 "Fatal error. This script requires mydir."
fi

# if variable is null
```

```
if [ ! -s "myvariable" ]; then echo -e "variable is null!" ; fi
#True of the length if "STRING" is zero.

# Using test command (same as []), to test if the length of variable is nonzero
test -n "$myvariable" && echo myvariable is "$myvariable" || echo myvariable is not set

# Test if file exist
if [ -e 'filename' ]
then
  echo -e "file exists!"
fi

# Test if file exist but also including symbolic links:
if [ -e myfile ] || [ -L myfile ]
then
  echo -e "file exists!"
fi

# Test if the value of x is greater or equal than 5
if [ "$x" -ge 5 ]; then echo -e "greater or equal than 5!" ; fi

# Test if the value of x is greater or equal than 5, in bash/ksh/zsh:
if ((x >= 5)); then echo -e "greater or equal than 5!" ; fi

# Use (( )) for arithmetic operation
if ((j==u+2)); then echo -e "j==u+2!!" ; fi

# Use [[ ]] for comparison
if [[ $age -gt 21 ]]; then echo -e "forever 21!!" ; fi
```

22.4.7 `for` loop

Listing 28: *'for'* statement

```
For loop:
---------

# Echo the file name under the current directory
for i in $(ls); do echo file $i;done
#or
for i in *; do echo file $i; done

# Make directories listed in a file (e.g. myfile)
for dir in $(<myfile); do mkdir $dir; done

# Press any key to continue each loop
for i in $(cat tpc_stats_0925.log |grep failed|grep -o '\query\w\{1,2\}');do cat ${i}.log; read -rsp $'Press any
    key to continue...\n' -n1 key;done

# Print a file line by line when a key is pressed,
oifs="$IFS"; IFS=$'\n'; for line in $(cat myfile); do ...; done
while read -r line; do ...; done <myfile

#If only one word a line, simply
for line in $(cat myfile); do echo $line; read -n1; done

#Loop through an array
for i in "${arrayName[@]}"; do echo $i;done
```

22.4.8 `while` loop

Listing 29: *'while'* statement

```
While loop:
-----------

# Column subtraction of a file (e.g. a 3 columns file)
while read a b c; do echo $(($c-$b));done < <(head filename)
#there is a space between the two '<'s

# Sum up column subtraction
i=0; while read a b c; do ((i+=$c-$b)); echo $i; done < <(head filename)

# Keep checking a running process (e.g. perl) and start another new process (e.g. python) immediately after it. (
    BETTER use the wait command! Ctrl+F 'wait')
while [[ $(pidof perl) ]];do echo f;sleep 10;done && python timetorunpython.py
switch (case in bash)
read type;
case $type in
  '0')
```

```
    echo 'how'
    ;;
  '1')
    echo 'are'
    ;;
  '2')
    echo 'you'
    ;;
esac
```

22.4.9 time commands

Listing 30: *time* related

```
Time:
-----

Find out the time require for executing a command
time echo hi

Wait for some time (e.g 10s)
sleep 10

Print date with formatting
date +%F
# 2020-07-19
# or
date +'%d-%b-%Y-%H:%M:%S'
# 10-Apr-2020-21:54:40

# Returns the current time with nanoseconds.
date +"%T.%N"
# 11:42:18.664217000

# Get the seconds since epoch (Jan 1 1970) for a given date (e.g Mar 16 2021)
date -d "Mar 16 2021" +%s
# 1615852800
# or
date -d "Tue Mar 16 00:00:00 UTC 2021"  +%s
# 1615852800

# Convert the number of seconds since epoch back to date
date --date @1615852800
# Tue Mar 16 00:00:00 UTC 2021

wait for random duration (e.g. sleep 1-5 second, like adding a jitter)
sleep $[ ( $RANDOM % 5 ) + 1 ]

Log out your account after a certain period of time (e.g 10 seconds)
TMOUT=10
#once you set this variable, logout timer start running!

Set how long you want to run a command
#This will run the command 'sleep 10' for only 1 second.
timeout 1 sleep 10

Set when you want to run a command (e.g 1 min from now)
at now + 1min  #time-units can be minutes, hours, days, or weeks
warning: commands will be executed using /bin/sh
at> echo hihigithub >~/itworks
at> <EOT>   # press Ctrl + D to exit
job 1 at Wed Apr 18 11:16:00 2018
```

22.4.10 download stuff

Listing 31: *download* things

```
Download:
---------

Download the content of this README.md (the one your are viewing now)
curl https://raw.githubusercontent.com/onceupon/Bash-Oneliner/master/README.md | pandoc -f markdown -t man | man
    -l -

# or w3m (a text based web browser and pager)
curl https://raw.githubusercontent.com/onceupon/Bash-Oneliner/master/README.md | pandoc | w3m -T text/html

# or using emacs (in emac text editor)
emacs --eval '(org-mode)' --insert <(curl https://raw.githubusercontent.com/onceupon/Bash-Oneliner/master/README.
    md | pandoc -t org)
```

```
# or using emacs (on terminal, exit using Ctrl + x then Ctrl + c)
emacs -nw --eval '(org-mode)' --insert <(curl https://raw.githubusercontent.com/onceupon/Bash-Oneliner/master/
    README.md | pandoc -t org)

Download all from a page:
wget -r -l1 -H -t1 -nd -N -np -A mp3 -e robots=off http://example.com
# -r: recursive and download all links on page
# -l1: only one level link
# -H: span host, visit other hosts
# -t1: numbers of retries
# -nd: don't make new directories, download to here
# -N: turn on timestamp
# -nd: no parent
# -A: type (separate by ,)
# -e robots=off: ignore the robots.txt file which stop wget from crashing the site, sorry example.com

Upload a file to web and download (https://transfer.sh/)
#  Upload a file (e.g. filename.txt):
curl --upload-file ./filename.txt https://transfer.sh/filename.txt
# the above command will return a URL, e.g: https://transfer.sh/tG8rM/filename.txt

# Next you can download it by:
curl https://transfer.sh/tG8rM/filename.txt -o filename.txt

Download file if necessary:
data=file.txt
url=http://www.example.com/$data
if [ ! -s $data ];then
    echo "downloading test data..."
    wget $url
fi

Wget to a filename (when a long name)
wget -O filename "http://example.com"

Wget files to a folder
wget -P /path/to/directory "http://example.com"

Instruct curl to follow any redirect until it reaches the final destination:
curl -L google.com
```

22.4.11 random stuff

Listing 32: *random* use

```
Random:
-------

Random generate password (e.g. generate 5 password each of length 13)
sudo apt install pwgen
pwgen 13 5
#sahcahS9dah4a xieXaiJaey7xa UuMeo0ma7eic9 Ahpah9see3zai acerae7Huigh7

Random pick 100 lines from a file:
shuf -n 100 filename

Random order (lucky draw):
for i in a b c d e; do echo $i; done | shuf

Echo series of random numbers between a range (e.g. shuffle numbers from 0-100, then pick 15 of them randomly)
shuf -i 0-100 -n 15

Echo a random number
echo $RANDOM

Random from 0-9
echo $((RANDOM % 10))

Random from 1-10
echo $(((RANDOM %10)+1))
```

22.4.12 Xwindow stuff

Listing 33: *Xwindow* GUI

```
Xwindow:
--------

X11 GUI applications! Here are some GUI tools for you if you get bored by the text-only environment.
```

```
Enable X11 forwarding,in order to use graphical application on servers
ssh -X user_name@ip_address

# or setting through xhost
# --> Install the following for Centos:
# xorg-x11-xauth
# xorg-x11-fonts-*
# xorg-x11-utils


Little xwindow tools:

xclock

xeyes

xcowsay

Open pictures/images from ssh server
1. ssh -X user_name@ip_address
2. apt-get install eog
3. eog picture.png

Watch videos on server
1. ssh -X user_name@ip_address
2. sudo apt install mpv
3. mpv myvideo.mp4

Use gedit on server (GUI editor)
1. ssh -X user_name@ip_address
2. apt-get install gedit
3. gedit filename.txt

Open PDF file from ssh server
1. ssh -X user_name@ip_address
2. apt-get install evince
3. evince filename.pdf

Use google-chrome browser from ssh server
1. ssh -X user_name@ip_address
2. apt-get install libxss1 libappindicator1 libindicator7
3. wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
4. sudo apt-get install -f
5. dpkg -i google-chrome*.deb
6. google-chrome
```

22.4.13 system stuff

Listing 34: *system* admin

```
System:
-------

Work with yum history:
# List yum history (e.g install, update)

sudo yum history
# Example output:
# Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
# ID     | Login user             | Date and time    | Action(s)      | Altered
# -------------------------------------------------------------------------------
#     11 |  ... <myuser>          | 2020-04-10 10:57 | Install        |    1 P<
#     10 |  ... <myuser>          | 2020-03-27 05:21 | Install        |    1 >P
#      9 |  ... <myuser>          | 2020-03-05 11:57 | I, U           |   56 *<
# ...

# Show more details of a yum history (e.g. history #11)
sudo yum history info 11

# Undo a yum history (e.g. history #11, this will uninstall some packages)
sudo yum history undo 11

Audit files to see who made changes to a file [RedHat based system only]
# To audit a directory recursively for changes (e.g. myproject)
auditctl -w /path/to/myproject/ -p wa

# If you delete a file name "VIPfile", the deletion is recorded in /var/log/audit/audit.log
sudo grep VIPfile /var/log/audit/audit.log
#type=PATH msg=audit(1581417313.678:113): item=1 name="VIPfile" inode=300115 dev=ca:01 mode=0100664 ouid=1000
     ogid=1000 rdev=00:00 nametype=DELETE cap_fp=0000000000000000 cap_fi=0000000000000000 cap_fe=0 cap_fver=0

Check out whether SELinux is enabled
sestatus
```

```
# SELinux status:                 enabled
# SELinuxfs mount:                /sys/fs/selinux
# SELinux root directory:         /etc/selinux
# Loaded policy name:             targeted
# Current mode:                   enforcing
# Mode from config file:          enforcing
# Policy MLS status:              enabled
# Policy deny_unknown status:     allowed
# Max kernel policy version:      31

Generate public key from private key
ssh-keygen -y -f ~/.ssh/id_rsa > ~/.ssh/id_rsa.pub

Copy your default public key to remote user
ssh-copy-id <user_name>@<server_IP>
# then you need to enter the password
# and next time you won't need to enter password when ssh to that user
Copy default public key to remote user using the required private key (e.g. use your mykey.pem key to copy your
    id_rsa.pub to the remote user)
# before you need to use mykey.pem to ssh to remote user.
ssh-copy-id -i ~/.ssh/id_rsa.pub -o "IdentityFile ~/Downloads/mykey.pem" <user_name>@<server_IP>
# now you don't need to use key to ssh to that user.

SSH Agent Forwarding
# To bring your key with you when ssh to serverA, then ssh to serverB from serverA using the key.
ssh-agent
ssh-add /path/to/mykey.pem
ssh -A <username>@<IP_of_serverA>
# Next you can ssh to serverB
ssh <username>@<IP_of_serverB>
Set the default user and key for a host when using SSH
# add the following to ~/.ssh/config
Host myserver
  User myuser
  IdentityFile ~/path/to/mykey.pem

# Next, you could run "ssh myserver" instead of "ssh -i ~/path/to/mykey.pem myuser@myserver"

Follow the most recent logs from service
journalctl -u <service_name> -f
Eliminate the zombie
# A zombie is already dead, so you cannot kill it. You can eliminate the zombie by killing its parent.
# First, find PID of the zombie
ps aux| grep 'Z'
# Next find the PID of zombie's parent
pstree -p -s <zombie_PID>
# Then you can kill its parent and you will notice the zombie is gone.
sudo kill 9 <parent_PID>

Show memory usage
free -c 10 -mhs 1
# print 10 times, at 1 second interval

Display CPU and IO statistics for devices and partitions.
# refresh every second
iostat -x -t 1

Display bandwidth usage on an network interface (e.g. enp175s0f0)
iftop -i enp175s0f0

Tell how long the system has been running and number of users
uptime

Check if it's root running
if [ "$EUID" -ne 0 ]; then
        echo "Please run this as root"
        exit 1
fi


Change shell of a user (e.g. bonnie)
chsh -s /bin/sh bonnie
# /etc/shells: valid login shells

Change root / fake root / jail (e.g. change root to newroot)
chroot /home/newroot /bin/bash

# To exit chroot
exit


Display file status (size; access, modify and change time, etc) of a file (e.g. filename.txt)
stat filename.txt

Snapshot of the current processes
ps aux
```

```
Display a tree of processes
pstree

Find maximum number of processes
cat /proc/sys/kernel/pid_max

Print or control the kernel ring buffer
dmesg

Show IP address
$ip add show

# or
ifconfig

Print previous and current SysV runlevel
runlevel
# or
who -r

Change SysV runlevel (e.g. 5)
init 5
#or
telinit 5

Display all available services in all runlevels,
chkconfig --list
# update-rc.d equivalent to chkconfig in ubuntu

Check system version
cat /etc/*-release

Linux Programmer's Manuel: hier- description of the filesystem hierarchy
man hier


Control the systemd system and service manager
# e.g. check the status of cron service
systemctl status cron.service

# e.g. stop cron service
systemctl stop cron.service
List job
jobs -l
Run a program with modified priority (e.g. ./test.sh)
# nice value is adjustable from -20 (most favorable) to +19
# the nicer the application, the lower the priority
# Default niceness: 10; default priority: 80

nice -10 ./test.sh


Export PATH:
export PATH=$PATH:~/path/you/want

Make file executable
chmod +x filename
# you can now ./filename to execute it

Print system information
uname -a

# Check system hardware-platform (x86-64)
uname -i


Surf the net
links www.google.com

Add user, set passwd
useradd username
passwd username

Edit PS1 variable for bash (e.g. displaying the whole path)
1. vi ~/.bash_profile
2. export PS1='\u@\h:\w\$'
# $PS1 is a variable that defines the makeup and style of the command prompt
# You could use emojis and add timestamp to every prompt using the following value:
# export PS1="\t :\w\$ "
3. source ~/.bash_profile

Edit environment setting (e.g. alias)
1. vi ~/.bash_profile
2. alias pd="pwd" //no more need to type that 'w'!
3. source ~/.bash_profile

Print all alias
```

```
alias -p

Unalias (e.g. after alias ls='ls --color=auto')
unalias ls

Set and unset shell options
# print all shell options
shopt

# to unset (or stop) alias
shopt -u expand_aliases

# to set (or start) alias
shopt -s expand_aliases

List environment variables (e.g. PATH)
echo $PATH
# list of directories separated by a colon

List all environment variables for current user
env

Unset environment variable (e.g. unset variable 'MYVAR')
unset MYVAR


Show partition format
lsblk

Inform the OS of partition table changes
partprobe

Soft link program to bin
ln -s /path/to/program /home/usr/bin
# must be the whole path to the program

Show hexadecimal view of data
hexdump -C filename.class

Jump to different node
rsh node_name

Check port (active internet connection)
netstat -tulpn

Print resolved symbolic links or canonical file names
readlink filename

Find out the type of command and where it link to (e.g. python)
type python
# python is /usr/bin/python
# There are 5 different types, check using the 'type -f' flag
# 1. alias     (shell alias)
# 2. function  (shell function, type will also print the function body)
# 3. builtin   (shell builtin)
# 4. file      (disk file)
# 5. keyword   (shell reserved word)

# You can also use `which`
which python
# /usr/bin/python

List all functions names
declare -F

List total size of a directory
du -hs .
# or
du -sb

Copy directory with permission setting
cp -rp /path/to/directory

Store current directory
pushd .

# then pop
popd

#or use dirs to display the list of currently remembered directories.
dirs -l

Show disk usage
df -h
# or
du -h
```

```
#or
du -sk /var/log/* |sort -rn |head -10
check the Inode utilization
df -i
# Filesystem      Inodes  IUsed   IFree IUse% Mounted on
# devtmpfs        492652    304  492348    1% /dev
# tmpfs           497233      2  497231    1% /dev/shm
# tmpfs           497233    439  496794    1% /run
# tmpfs           497233     16  497217    1% /sys/fs/cgroup
# /dev/nvme0n1p1 5037976 370882 4667094    8% /
# tmpfs           497233      1  497232    1% /run/user/1000

Show all file system type
df -TH

Show current runlevel
runlevel

Switch runlevel
init 3
#or
telinit 3

Permanently modify runlevel
1. edit /etc/init/rc-sysinit.conf
2. env DEFAULT_RUNLEVEL=2

Become root
su

Become somebody
su somebody

Report user quotes on device
repquota -auvs

Get entries in a number of important databases
getent database_name
# (e.g. the 'passwd' database)
getent passwd
# list all user account (all local and LDAP)
# (e.g. fetch list of grop accounts)
getent group
# store in database 'group'

Change owner of file
chown user_name filename
chown -R user_name /path/to/directory/
# chown user:group filename

Mount and unmount
# e.g. Mount /dev/sdb to /home/test
mount /dev/sdb /home/test

# e.g. Unmount /home/test
umount /home/test
List current mount detail
mount
# or
df

List current usernames and user-numbers
cat /etc/passwd

Get all username
getent passwd| awk '{FS="[:]"; print $1}'

Show all users
compgen -u

Show all groups
compgen -g

Show group of user
group username

Show uid, gid, group of user
id username

# variable for UID
echo $UID
Check if it's root
if [ $(id -u) -ne 0 ];then
    echo "You are not root!"
    exit;
fi
# 'id -u' output 0 if it's not root
```

```
Find out CPU information
more /proc/cpuinfo
# or
lscpu

Set quota for user (e.g. disk soft limit: 120586240; hard limit: 125829120)
setquota username 120586240 125829120 0 0 /home

Show quota for user
quota -v username

Display current libraries from the cache
ldconfig -p

Print shared library dependencies (e.g. for 'ls')
ldd /bin/ls

Check user login
lastlog

Check last reboot history
last reboot

Edit path for all users
joe /etc/environment
# edit this file

Show and set user limit
ulimit -u

Print out number of cores/ processors
nproc --all

Check status of each core
1. top
2. press '1'

Show jobs and PID
jobs -l

List all running services
service --status-all

Schedule shutdown server
shutdown -r +5 "Server will restart in 5 minutes. Please save your work."

Cancel scheduled shutdown
shutdown -c

Broadcast to all users
wall -n hihi

Kill all process of a user
pkill -U user_name

Kill all process of a program
kill -9 $(ps aux | grep 'program_name' | awk '{print $2}')

Set gedit preference on server
# You might have to install the following:
apt-get install libglib2.0-bin;
# or
yum install dconf dconf-editor;
yum install dbus dbus-x11;

# Check list
gsettings list-recursively

# Change some settings
gsettings set org.gnome.gedit.preferences.editor highlight-current-line true
gsettings set org.gnome.gedit.preferences.editor scheme 'cobalt'
gsettings set org.gnome.gedit.preferences.editor use-default-font false
gsettings set org.gnome.gedit.preferences.editor editor-font 'Cantarell Regular 12'

Add user to a group (e.g add user 'nice' to the group 'docker', so that he can run docker without sudo)
sudo gpasswd -a nice docker

Pip install python package without root
1. pip install --user package_name
2. You might need to export ~/.local/bin/ to PATH: export PATH=$PATH:~/.local/bin/

Removing old linux kernels (when /boot almost full...)
1. uname -a  #check current kernel, which should NOT be removed
2. sudo apt-get purge linux-image-X.X.X-X-generic  #replace old version

Change hostname
sudo hostname your-new-name
```

```
# if not working, do also:
hostnamectl set-hostname your-new-hostname
# then check with:
hostnamectl
# Or check /etc/hostname

# If still not working..., edit:
/etc/sysconfig/network
/etc/sysconfig/network-scripts/ifcfg-ensxxx
#add HOSTNAME="your-new-hostname"

List installed packages
apt list --installed
# or on Red Hat:
yum list installed

Check for package update
apt list --upgradeable
# or
sudo yum check-update

Run yum update excluding a package (e.g. do not update php packages)
sudo yum update --exclude=php*

Check which file make the device busy on umount
lsof /mnt/dir

When sound not working
killall pulseaudio
# then press Alt-F2 and type in pulseaudio
When sound not working
killall pulseaudio

List information about SCSI devices
lsscsi

Tutorial for setting up your own DNS server
http://onceuponmine.blogspot.tw/2017/08/set-up-your-own-dns-server.html

Tutorial for creating a simple daemon
http://onceuponmine.blogspot.tw/2017/07/create-your-first-simple-daemon.html

Tutorial for using your gmail to send email
http://onceuponmine.blogspot.tw/2017/10/setting-up-msmtprc-and-use-your-gmail.html

Using telnet to test open ports, test if you can connect to a port (e.g 53) of a server (e.g 192.168.2.106)
telnet 192.168.2.106 53

Change network maximum transmission unit (mtu) (e.g. change to 9000)
ifconfig eth0 mtu 9000

Get pid of a running process (e.g python)
pidof python
# or
ps aux|grep python

Check status of a process using PID
ps -p <PID>
#or
cat /proc/<PID>/status
cat /proc/<PID>/stack
cat /proc/<PID>/stat

NTP
# Start ntp:
ntpd

# Check ntp:
ntpq -p

Remove unnecessary files to clean your server
sudo apt-get autoremove
sudo apt-get clean
sudo rm -rf ~/.cache/thumbnails/*

# Remove old kernal:
sudo dpkg --list 'linux-image*'
sudo apt-get remove linux-image-OLDER_VERSION

Increase/ resize root partition (root partition is an LVM logical volume)
pvscan
lvextend -L +130G /dev/rhel/root -r
# Adding -r will grow filesystem after resizing the volume.

Create a UEFI Bootable USB drive (e.g. /dev/sdc1)
sudo dd if=~/path/to/isofile.iso of=/dev/sdc1 oflag=direct bs=1048576
```

```
Locate and remove a package
sudo dpkg -l | grep <package_name>
sudo dpkg --purge <package_name>

Create a ssh tunnel
ssh -f -L 9000:targetservername:8088 root@192.168.14.72 -N
#-f: run in background; -L: Listen; -N: do nothing
#the 9000 of your computer is now connected to the 8088 port of the targetservername through 192.168.14.72
#so that you can see the content of targetservername:8088 by entering localhost:9000 from your browser.

Get process ID of a process (e.g. sublime_text)
#pidof
pidof sublime_text

#pgrep, you don't have to type the whole program name
pgrep sublim

#pgrep, echo 1 if process found, echo 0 if no such process
pgrep -q sublime_text && echo 1 || echo 0

#top, takes longer time
top|grep sublime_text


Some benchmarking tools for your server
aio-stress - AIO benchmark.
bandwidth - memory bandwidth benchmark.
bonnie++ - hard drive and file system performance benchmark.
dbench - generate I/O workloads to either a filesystem or to a networked CIFS or NFS server.
dnsperf - authorative and recursing DNS servers.
filebench - model based file system workload generator.
fio - I/O benchmark.
fs_mark - synchronous/async file creation benchmark.
httperf - measure web server performance.
interbench - linux interactivity benchmark.
ioblazer - multi-platform storage stack micro-benchmark.
iozone - filesystem benchmark.
iperf3 - measure TCP/UDP/SCTP performance.
kcbench - kernel compile benchmark, compiles a kernel and measures the time it takes.
lmbench - Suite of simple, portable benchmarks.
netperf - measure network performance, test unidirectional throughput, and end-to-end latency.
netpipe - network protocol independent performance evaluator.
nfsometer - NFS performance framework.
nuttcp - measure network performance.
phoronix-test-suite - comprehensive automated testing and benchmarking platform.
seeker - portable disk seek benchmark.
siege - http load tester and benchmark.
sockperf - network benchmarking utility over socket API.
spew - measures I/O performance and/or generates I/O load.
stress - workload generator for POSIX systems.
sysbench - scriptable database and system performance benchmark.
tiobench - threaded IO benchmark.
unixbench - the original BYTE UNIX benchmark suite, provide a basic indicator of the performance of a Unix-like
     system.
wrk - HTTP benchmark.


Performance monitoring tool - sar
# installation
# It collects the data every 10 minutes and generate its report daily. crontab file (/etc/cron.d/sysstat) is
     responsible for collecting and generating reports.
yum install sysstat
systemctl start sysstat
systemctl enable sysstat

# show CPU utilization 5 times every 2 seconds.
sar 2 5

# show memory  utilization 5 times every 2 seconds.
sar -r 2 5

# show paging statistics 5 times every 2 seconds.
sar -B 2 5

# To generate all network statistic:
sar -n ALL

# reading SAR log file using -f
sar -f /var/log/sa/sa31|tail

Reading from journal file
journalctl --file ./log/journal/a90c18f62af546ccba02fa3734f00a04/system.journal  --since "2020-02-11 00:00:00"

Show a listing of last logged in users.
lastb
```

```
Show a listing of current logged in users, print information of them
who

Show who is logged on and what they are doing
w

Print the user names of users currently logged in to the current host.
users

Stop tailing a file on program terminate
tail -f --pid=<PID> filename.txt
# replace <PID> with the process ID of the program.

List all enabled services
systemctl list-unit-files|grep enabled



Hardware:
---------

Collect and summarize all hardware info of your machine
lshw -json >report.json
# Other options are: [ -html ]  [ -short ]  [ -xml ]  [ -json ]  [ -businfo ]  [ -sanitize ] ,etc

Finding Out memory device detail
sudo dmidecode -t memory

Print detail of CPU hardware
dmidecode -t 4
#           Type   Information
#           0   BIOS
#           1   System
#           2   Base Board
#           3   Chassis
#           4   Processor
#           5   Memory Controller
#           6   Memory Module
#           7   Cache
#           8   Port Connector
#           9   System Slots
#          11   OEM Strings
#          13   BIOS Language
#          15   System Event Log
#          16   Physical Memory Array
#          17   Memory Device
#          18   32-bit Memory Error
#          19   Memory Array Mapped Address
#          20   Memory Device Mapped Address
#          21   Built-in Pointing Device
#          22   Portable Battery
#          23   System Reset
#          24   Hardware Security
#          25   System Power Controls
#          26   Voltage Probe
#          27   Cooling Device
#          28   Temperature Probe
#          29   Electrical Current Probe
#          30   Out-of-band Remote Access
#          31   Boot Integrity Services
#          32   System Boot
#          34   Management Device
#          35   Management Device Component
#          36   Management Device Threshold Data
#          37   Memory Channel
#          38   IPMI Device
#          39   Power Supply

Count the number of Segate hard disks
lsscsi|grep SEAGATE|wc -l
# or
sg_map -i -x|grep SEAGATE|wc -l

Get UUID of a disk (e.g. sdb)
lsblk -f /dev/sdb
# or
sudo blkid /dev/sdb

Generate an UUID
uuidgen

Print detail of all hard disks
lsblk -io KNAME,TYPE,MODEL,VENDOR,SIZE,ROTA
#where ROTA means rotational device / spinning hard disks (1 if true, 0 if false)

List all PCI (Peripheral Component Interconnect) devices
lspci
```

```
# List information about NIC
lspci | egrep -i --color 'network|ethernet'

List all USB devices
lsusb

Linux modules
# Show the status of modules in the Linux Kernel
lsmod

# Add and remove modules from the Linux Kernel
modprobe
# or
# Remove a module
rmmod

# Insert a module
insmod

Controlling IPMI-enabled devices (e.g. BMC)
# Remotely finding out power status of the server
ipmitool -U <bmc_username> -P <bmc_password> -I lanplus -H <bmc_ip_address> power status

# Remotely switching on server
ipmitool -U <bmc_username> -P <bmc_password> -I lanplus -H <bmc_ip_address> power on

# Turn on panel identify light (default 15s)
ipmitool chassis identify 255

# Found out server sensor temperature
ipmitool sensors |grep -i Temp

# Reset BMC
ipmitool bmc reset cold

# Prnt BMC network
ipmitool lan print 1

# Setting BMC network
ipmitool -I bmc lan set 1 ipaddr 192.168.0.55
ipmitool -I bmc lan set 1 netmask 255.255.255.0
ipmitool -I bmc lan set 1 defgw ipaddr 192.168.0.1



Networking:
-----------

Resolve a domain to IP address(es)
dig +short www.example.com
# or
host www.example.com

Get DNS TXT record a of domain
dig -t txt www.example.com
# or
host -t txt www.example.com

Send a ping with a limited TTL to 10 (TTL: Time-To-Live, which is the maximum number of hops that a packet can
     travel across the Internet before it gets discarded.)
ping 8.8.8.8 -t 10

Print the route packets trace to network host
traceroute google.com

Check connection to host (e.g. check connection to port 80 and 22 of google.com)
nc -vw5 google.com 80
# Connection to google.com 80 port [tcp/http] succeeded!
nc -vw5 google.com 22
# nc: connect to google.com port 22 (tcp) timed out: Operation now in progress
# nc: connect to google.com port 22 (tcp) failed: Network is unreachable

Nc as a chat tool!
# From server A:
$ sudo nc -l 80
# then you can connect to the 80 port from another server (e.g. server B):
# e.g. telent <server A IP address> 80
# then type something in server B
# and you will see the result in server A!

Check which ports are listening for TCP connections from the network
#notice that some companies might not like you using nmap
nmap -sT -O localhost

# check port 0-65535
nmap  -p0-65535 localhost
```

```
Check if a host is up and scan for open ports, also skip host discovery.
#skips checking if the host is alive which may sometimes cause a false positive and stop the scan.
$ nmap google.com -Pn

# Example output:
# Starting Nmap 7.01 ( https://nmap.org ) at 2020-07-18 22:59 CST
# Nmap scan report for google.com (172.217.24.14)
# Host is up (0.013s latency).
# Other addresses for google.com (not scanned): 2404:6800:4008:802::200e
# rDNS record for 172.217.24.14: tsa01s07-in-f14.1e100.net
# Not shown: 998 filtered ports
# PORT    STATE SERVICE
# 80/tcp  open  http
# 443/tcp open  https
#
# Nmap done: 1 IP address (1 host up) scanned in 3.99 seconds

Scan for open ports and OS and version detection (e.g. scan the domain "scanme.nmap.org")
$ nmap -A -T4 scanme.nmap.org
# -A to enable OS and version detection, script scanning, and traceroute; -T4 for faster execution

Look up website information (e.g. name server), searches for an object in a RFC 3912 database.
whois google.com

Show the SSL certificate of a domain
openssl s_client -showcerts -connect www.example.com:443

Display IP address
ip a

Display route table
ip r

Display ARP cache (ARP cache displays the MAC addresses of device in the same network that you have connected to)
ip n

Add transient IP addres (reset after reboot) (e.g. add 192.168.140.3/24 to device eno16777736)
ip address add 192.168.140.3/24 dev eno16777736

Persisting network configuration changes
sudo vi /etc/sysconfig/network-scripts/ifcfg-enoxxx
# then edit the fields: BOOTPROT, DEVICE, IPADDR, NETMASK, GATEWAY, DNS1 etc

Refresh NetworkManager
sudo nmcli c reload

Restart all interfaces
sudo systemctl restart network.service

To view hostname, OS, kernal, architecture at the same time!
hostnamectl

Set hostname (set all transient, static, pretty hostname at once)
hostnamectl set-hostname "mynode"

Find out the web server (e.g Nginx or Apache) of a website
curl -I http://example.com/
# HTTP/1.1 200 OK
# Server: nginx
# Date: Thu, 02 Jan 2020 07:01:07 GMT
# Content-Type: text/html
# Content-Length: 1119
# Connection: keep-alive
# Vary: Accept-Encoding
# Last-Modified: Mon, 09 Sep 2019 10:37:49 GMT
# ETag: "xxxxxx"
# Accept-Ranges: bytes
# Vary: Accept-Encoding

Find out the http status code of a URL
curl -s -o /dev/null -w "%{http_code}" https://www.google.com

Unshorten a shortended URL
curl -s -o /dev/null -w "%{redirect_url}" https://bit.ly/34EFwWC

Perform network throughput tests
# server side:
$ sudo iperf -s -p 80
# client side:
iperf -c <server IP address> --parallel 2 -i 1 -t 2 -p 80

To block port 80 (HTTP server) using iptables.
sudo iptables -A INPUT -p tcp --dport 80 -j DROP

# only block connection from an IP address
sudo iptables -A INPUT -s <IP> -p tcp --dport 80 -j DROP
```

22.4.14 data stuff

Listing 35: *data* wrangling

```
Data wrangling:
---------------

Print some words that start with a particular string (e.g. words start with 'phy')
# If file is not specified, the file /usr/share/dict/words is used.
look phy|head -n 10
# Phil
# Philadelphia
# Philadelphia's
# Philby
# Philby's
# Philip
# Philippe
# Philippe's
# Philippians
# Philippine

Repeat printing string n times (e.g. print 'hello world' five times)
printf 'hello world\n%.0s' {1..5}

Do not echo the trailing newline
username=`echo -n "bashoneliner"`

Copy a file to multiple files (e.g copy fileA to file(B-D))
tee <fileA fileB fileC fileD >/dev/null

Delete all non-printing characters
tr -dc '[:print:]' < filename

Remove newline / nextline
tr --delete '\n' <input.txt >output.txt

Replace newline
tr '\n' ' ' <filename

To uppercase/lowercase
tr /a-z/ /A-Z/

Translate a range of characters (e.g. substitute a-z into a)
echo 'something' |tr a-z a
# aaaaaaaaa

Compare two files (e.g. fileA, fileB)
diff fileA fileB
# a: added; d:delete; c:changed
# or
sdiff fileA fileB
# side-to-side merge of file differences

Compare two files, strip trailing carriage return/ nextline (e.g. fileA, fileB)
 diff fileA fileB --strip-trailing-cr

Number a file (e.g. fileA)
nl fileA
#or
nl -nrz fileA
# add leading zeros
#or
nl -w1 -s ' '
# making it simple, blank separate

Join two files field by field with tab (default join by the first column of both file, and default separator is
    space)
# fileA and fileB should have the same ordering of lines.
join -t '\t' fileA fileB

# Join using specified field (e.g. column 3 of fileA and column 5 of fileB)
join -1 3 -2 5 fileA fileB

Combine/ paste two or more files into columns (e.g. fileA, fileB, fileC)
paste fileA fileB fileC
# default tab separate
Group/combine rows into one row
# e.g.
# AAAA
# BBBB
# CCCC
# DDDD
cat filename|paste - -
# AAAABBBB
# CCCCDDDD
cat filename|paste - - - -
```

```
# AAAABBBBCCCCDDDD

Fastq to fasta (fastq and fasta are common file formats for bioinformatics sequence data)
cat file.fastq | paste - - - - | sed 's/^@/>/g'| cut -f1-2 | tr '\t' '\n' >file.fa

Reverse string
echo 12345| rev

Generate sequence 1-10
seq 10

Find average of input list/file of integers
i=`wc -l filename|cut -d ' ' -f1`; cat filename| echo "scale=2;(`paste -sd+`)/"$i|bc

Generate all combination (e.g. 1,2)
echo {1,2}{1,2}
# 1 1, 1 2, 2 1, 2 2

Generate all combination (e.g. A,T,C,G)
set = {A,T,C,G}
group= 5
for ((i=0; i<$group; i++));do
    repetition=$set$repetition;done
    bash -c "echo "$repetition""

Read file content to variable
foo=$(<test1)

Echo size of variable
echo ${#foo}

Echo a tab
echo -e ' \t '

Split file into smaller file
# Split by line (e.g. 1000 lines/smallfile)
split -d -l 1000 largefile.txt

# Split by byte without breaking lines across files
split -C 10 largefile.txt

Create a large amount of dummy files (e.g 100000 files, 10 bytes each):
#1. Create a big file
dd if=/dev/zero of=bigfile bs=1 count=1000000
#2. Split the big file to 100000 10-bytes files
 split -b 10 -a 10 bigfile

Rename all files (e.g. remove ABC from all .gz files)
rename 's/ABC//' *.gz

Remove file extension (e.g remove .gz from filename.gz)
basename filename.gz .gz
zcat filename.gz> $(basename filename.gz .gz).unpacked

Add file extension to all file(e.g add .txt)
rename s/$/.txt/ *
# You can use rename -n s/$/.txt/ * to check the result first, it will only print sth like this:
# rename(a, a.txt)
# rename(b, b.txt)
# rename(c, c.txt)

Squeeze repeat patterns (e.g. /t/t --> /t)
tr -s "/t" < filename

Do not print nextline with echo
echo -e 'text here \c'

View first 50 characters of file
head -c 50 file

Cut and get last column of a file
cat file|rev | cut -d/ -f1 | rev

Add one to variable/increment/ i++ a numeric variable (e.g. $var)
((var++))
# or
var=$((var+1))

Cut the last column
cat filename|rev|cut -f1|rev

Cat to a file
cat >myfile

let me add sth here
exit by control + c
^C
```

```
Clear the contents of a file (e.g. filename)
>filename

Append to file (e.g. hihi)
echo 'hihi' >>filename

Working with json data
#install the useful jq package
#sudo apt-get install jq
#e.g. to get all the values of the 'url' key, simply pipe the json to the following jq command(you can use .[].
     to select inner json, i.e jq '.[].url')
cat file.json | jq '.url'

Decimal to Binary (e.g get binary of 5)
D2B=({0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1}{0..1})
echo -e ${D2B[5]}
#00000101

echo -e ${D2B[255]}
#11111111

Wrap each input line to fit in specified width (e.g 4 integers per line)
echo "00110010101110001101" | fold -w4
# 0011
# 0010
# 1011
# 1000
# 1101

Sort a file by column and keep the original order
sort -k3,3 -s

Right align a column (right align the 2nd column)
cat file.txt|rev|column -t|rev

To both view and store the output
echo 'hihihihi' | tee outputfile.txt
# use '-a' with tee to append to file.

Show non-printing (Ctrl) characters with cat
cat -v filename

Convert tab to space
expand filename

Convert space to tab
unexpand filename

Display file in octal ( you can also use od to display hexadecimal, decimal, etc)
od filename

Reverse cat a file
tac filename

Reverse the result from uniq -c
while read a b; do yes $b |head -n $a ;done <test.txt




Others:
-------

Describe the format and characteristics of image files.
identify myimage.png
#myimage.png PNG 1049x747 1049x747+0+0 8-bit sRGB 1.006MB 0.000u 0:00.000

Bash auto-complete (e.g. show options "now tomorrow never" when you press'tab' after typing "dothis")
More examples

complete -W "now tomorrow never" dothis
# ~$ dothis
# never     now       tomorrow
# press 'tab' again to auto-complete after typing 'n' or 't'

Displays a calendar
# print the current month, today will be highlighted.
cal
# October 2019
# Su Mo Tu We Th Fr Sa
#     1  2  3  4  5
# 6  7  8  9 10 11 12
# 13 14 15 16 17 18 19
# 20 21 22 23 24 25 26
# 27 28 29 30 31
```

```
# only display November
cal -m 11

Convert the hexadecimal MD5 checksum value into its base64-encoded format.
openssl md5 -binary /path/to/file | base64
# NWbeOpeQbtuY0ATWuUeumw==

Forces applications to use the default language for output
export LC_ALL=C

# to revert:
unset LC_ALL

Encode strings as Base64 strings
echo test|base64
#dGVzdAo=

Get parent directory of current directory
dirname 'pwd'

Read .gz file without extracting
zmore filename
# or
zless filename

Run command in background, output error file
some_commands  &>log &
# or
some_commands 2>log &
# or
some_commands 2>&1| tee logfile
# or
some_commands |& tee logfile
# or
some_commands 2>&1 >>outfile
#0: standard input; 1: standard output; 2: standard error

Run multiple commands in background
# run sequentially
(sleep 2; sleep 3) &

# run parallelly
sleep 2 & sleep 3 &

Run process even when logout (immune to hangups, with output to a non-tty)
# e.g. Run myscript.sh even when log out.
nohup bash myscript.sh

Send mail
echo 'heres the content'| mail -a /path/to/attach_file.txt -s 'mail.subject' me@gmail.com
# use -a flag to set send from (-a "From: some@mail.tld")

Convert .xls to csv
xls2csv filename

Make BEEP sound
speaker-test -t sine -f 1000 -l1

Set beep duration
(speaker-test -t sine -f 1000) & pid=$!;sleep 0.1s;kill -9 $pid

Editing your history
history -w
vi ~/.bash_history
history -r
#or
history -d [line_number]

Interacting with history
# list 5 previous command (similar to 'history |tail -n 5' but wont print the history command itself)
fc -l -5

Delete current bash command
Ctrl+U
# or
Ctrl+C
# or
Alt+Shift+#
# to make it to history

Add something to history (e.g. "addmetohistory")
# addmetodistory
# just add a "#" before~~

Get last history/record filename
head !$
```

```
Clean screen
clear
# or simply Ctrl+l

Backup with rsync
rsync -av filename filename.bak
rsync -av directory directory.bak
rsync -av --ignore_existing directory/ directory.bak
rsync -av --update directory directory.bak
rsync -av directory user@ip_address:/path/to/directory.bak
# skip files that are newer on receiver (i prefer this one!)

Make all directories at one time!
mkdir -p project/{lib/ext,bin,src,doc/{html,info,pdf},demo/stat}
# -p: make parent directory
# this will create project/doc/html/; project/doc/info; project/lib/ext ,etc

Run command only if another command returns zero exit status (well done)
cd tmp/ && tar xvf ~/a.tar

Run command only if another command returns non-zero exit status (not finish)
cd tmp/a/b/c ||mkdir -p tmp/a/b/c

Use backslash "" to break long command
cd tmp/a/b/c \
> || \
>mkdir -p tmp/a/b/c

List file type of file (e.g. /tmp/)
file /tmp/
# tmp/: directory

Writing Bash script ('#!'' is called shebang )
#!/bin/bash
file=${1#*.}

# remove string before a "."

Python simple HTTP Server
python -m SimpleHTTPServer
# or when using python3:
python3 -m http.server

Read user input
read input
echo $input

Array
declare -a array=()
# or
declare array=()
# or associative array
declare -A array=()

Send a directory
scp -r directoryname user@ip:/path/to/send

Fork bomb
# Don't try this at home!
# It is a function that calls itself twice every call until you run out of system resources.
# A '# ' is added in front for safety reason, remove it when seriously you are testing it.
# :(){:|:&};:

Use the last argument
!$

Check last exit code
echo $?

Extract .xz
unxz filename.tar.xz
# then
tar -xf filename.tar

Unzip tar.bz2 file (e.g. file.tar.bz2)
tar xvfj file.tar.bz2

Unzip tar.xz file (e.g. file.tar.xz)
unxz file.tar.xz
tar xopf file.tar

Extract to a path
tar xvf -C /path/to/directory filename.gz

Zip the content of a directory without including the directory itself
# First cd to the directory, they run:
zip -r -D ../myzipfile .
```

```
# you will see the myzipfile.zip in the parent directory (cd ..)
Output a y/n repeatedly until killed
# 'y':
yes
# or 'n':
yes n
# or 'anything':
yes anything

# pipe yes to other command
yes | rm -r large_directory

Create large dummy file of certain size instantly (e.g. 10GiB)
fallocate -l 10G 10Gigfile

Create dummy file of certain size (e.g. 200mb)
dd if=/dev/zero of=//dev/shm/200m bs=1024k count=200
# or
dd if=/dev/zero of=//dev/shm/200m bs=1M count=200

# Standard output:
# 200+0 records in
# 200+0 records out
# 209715200 bytes (210 MB) copied, 0.0955679 s, 2.2 GB/s

Keep /repeatedly executing the same command (e.g Repeat 'wc -l filename' every 1 second)
watch -n 1 wc -l filename

Print commands and their arguments when execute (e.g. echo expr 10 + 20 )
set -x; echo `expr 10 + 20 `

Print some meaningful sentences to you (install fortune first)
fortune

Colorful (and useful) version of top (install htop first)
htop

Press any key to continue
read -rsp $'Press any key to continue...\n' -n1 key

Run sql-like command on files from terminal
# download:
# https://github.com/harelba/q
# example:
q -d "," "select c3,c4,c5 from /path/to/file.txt where c3='foo' and c5='boo'"

Using Screen for multiple terminal sessions
# Create session and attach:
screen

# Create a screen and name it 'test'
screen -S test

# Create detached session foo:
screen -S foo -d -m

# Detached session foo:
screen: ^a^d

# List sessions:
screen -ls

# Attach last session:
screen -r

# Attach to session foo:
screen -r foo

# Kill session foo:
screen -r foo -X quit


# Scroll:
# Hit your screen prefix combination (C-a / control+A), then hit Escape.
# Move up/down with the arrow keys.

# Redirect output of an already running process in Screen:
# (C-a / control+A), then hit 'H'

# Store screen output for Screen:
# Ctrl+A, Shift+H
# You will then find a screen.log file under current directory.
Using Tmux for multiple terminal sessions

# Create session and attach:
tmux
```

```
# Attach to session foo:
tmux attach -t foo

# Detached session foo:
^bd

# List sessions:
tmux ls

# Attach last session:
tmux attach

# Kill session foo:
tmux kill-session -t foo

# Create detached session foo:
tmux new -s foo -d

# Send command to all panes in tmux:
Ctrl-B
:setw synchronize-panes

# Some tmux pane control commands:
Ctrl-B
#   Panes (splits), Press Ctrl+B, then input the following symbol:
#   %  horizontal split
#   "  vertical split
#   o  swap panes
#   q  show pane numbers
#   x  kill pane
#   space - toggle between layouts

#   Distribute Vertically (rows):
select-layout even-vertical
#   or
Ctrl+b, Alt+2

# Distribute horizontally (columns):
select-layout even-horizontal
#   or
Ctrl+b, Alt+1

# Scroll
Ctrl-b then \[ then you can use your normal navigation keys to scroll around.
Press q to quit scroll mode.

Pass password to ssh
sshpass -p mypassword ssh root@10.102.14.88 "df -h"

Wait for a pid (job) to complete
wait %1
# or
wait $PID
wait ${!}
#wait ${!} to wait till the last background process ($! is the PID of the last background process)

Convert pdf to txt
sudo apt-get install poppler-utils
pdftotext example.pdf example.txt


List only directory
ls -d */

List one file per line.
ls -1

# or list all, do not ignore entries starting with .
ls -1a

Capture/record/save terminal output (capture everything you type and output)
script output.txt
# start using terminal
# to logout the screen session (stop saving the contents), type exit.


List contents of directories in a tree-like format.
tree
# go to the directory you want to list, and type tree (sudo apt-get install tree)
# output:
# home/
# +- project
#     +- 1
#     +- 2
#     +- 3
#     +- 4
#     +- 5
```

```
#
# set level directories deep (e.g. level 1)
tree -L 1
# home/
# +- project

Set up virtualenv(sandbox) for python
# 1. install virtualenv.
sudo apt-get install virtualenv
# 2. Create a directory (name it .venv or whatever name your want) for your new shiny isolated environment.
virtualenv .venv
# 3. source virtual bin
source .venv/bin/activate
# 4. you can check check if you are now inside a sandbox.
type pip
# 5. Now you can install your pip package, here requirements.txt is simply a txt file containing all the packages
     you want. (e.g tornado==4.5.3).
pip install -r requirements.txt
# 6. Exit virtual environment
deactivate
```

22.4.15 find

Listing 36: *'find'* utility

```
Find:
-----

Hinweis:
   find -name pattern   - - -  pattern immer in "" einschliessen!
suche files 'juenger als ...':
   find -H . -newermt 2020-05-13 -name "*" -type f -ls
   find -L /SMUE/KWx/latex -newermt 2020-05-01 -name "*" -type f -ls 2>/dev/null  |grep "\.C"


List all sub directory/file in the current directory
find .

List all files under the current directory
find . -type f

List all directories under the current directory
find . -type d

Edit all files under current directory (e.g. replace 'www' with 'ww')
find . -name '*.php' -exec sed -i 's/www/w/g' {} \;

# if there are no subdirectory
replace "www" "w" -- *
# a space before *

Find and output only filename (e.g. "mso")
find mso*/ -name M* -printf "%f\n"

Find large files in the system (e.g. >4G)
find / -type f -size +4G

Find and delete file with size less than (e.g. 74 byte)
find . -name "*.mso" -size -74c -delete
# M for MB, etc

Find empty (0 byte) files
find . -type f -empty
# to further delete all the empty files
find . -type f -empty -delete

Recursively count all the files in a directory
find . -type f | wc -l
Condition and loop
```

aus der man-page *man 1 find*:

FIND(1) - search for files in a directory hierarchy

find [-H] [-L] [-P] [-D debugopts] [-Olevel] [path...] [expression]

DESCRIPTION
This manual page documents the GNU version of find. GNU find searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left hand side is false for and operations, true for or), at which point find moves on to the next file name.
If you are using find in an environment where security is important (for example if you are using it to search directories that are writable by other users), you should read the SSecurity Considerations"chapter of the findutils documentation, which is called Finding Files and comes with findutils. That document also includes a lot more detail and discussion than this manual page, so you may find it a more useful source of information.

OPTIONS
The -H, -L and -P options control the treatment of symbolic links. Command-line arguments fol- lowing these are taken to be names of files or directories to be examined, up to the first argument that begins with '-', or the argument '(' or '!'. That argument and any following arguments are taken to be the expression describing what is to be searched for. If no paths are given, the current directory is used. If no expression is given, the expression -print is used (but you should probably consider using -print0 instead, anyway).
This manual page talks about 'options' within the expression list. These options control the behaviour of find but are specified immediately after the last path name. The five 'real' options -H, -L, -P, -D and -O must appear before the first path name, if at all. A double dash - can also be used to signal that any remaining arguments are not options (though ensuring that all start points begin with either './' or '/' is generally safer if you use wildcards in the list of start points).

-P
Never follow symbolic links. This is the default behaviour. When find examines or prints information a file, and the file is a symbolic link, the information used shall be taken from the properties of the symbolic link itself.

-L
Follow symbolic links. When find examines or prints information about files, the infor- mation used shall be taken from the properties of the file to which the link points, not from the link itself (unless it is a broken symbolic link or find is unable to examine the file to which the link points). Use of this option implies -noleaf. If you later use the -P option, -noleaf will still be in effect. If -L is in effect and find discov- ers a symbolic link to a subdirectory during its search, the subdirectory pointed to by the symbolic link will be searched.
When the -L option is in effect, the -type predicate will always match against the type of the file that a symbolic link points to rather than the link itself (unless the sym- bolic link is broken). Using -L causes the -lname and -ilname predicates always to return false.

-H
Do not follow symbolic links, except while processing the command line arguments. When find examines or prints information about files, the information used shall be taken from the properties of the symbolic link itself. The only exception to this behaviour is when a file specified on the command line is a symbolic link, and the link can be resolved. For that situation, the information used is taken from whatever the link points to (that is, the link is followed). The information about the link itself is used as a fallback if the file pointed to by the symbolic link cannot be examined. If -H is in effect and one of the paths specified on the command line is a symbolic link to a directory, the contents of that directory will be examined (though of course -maxdepth 0 would prevent this).

If more than one of -H, -L and -P is specified, each overrides the others; the last one appear- ing on the command line takes effect. Since it is the default, the -P option should be consid- ered to be in effect unless either -H or -L is specified.

GNU find frequently stats files during the processing of the command line itself, before any searching has begun. These options also affect how those arguments are processed. Specifi- cally, there are a number of tests that compare files listed on the command line against a file we are currently considering. In each case, the file specified on the command line will have been examined and some of its properties will have been saved. If the named file is in fact a symbolic link, and the -P option is in effect (or if neither -H nor -L were specified), the information used for the comparison will be taken from the properties of the symbolic link. Otherwise, it will be taken from the properties of the file the link points to. If find cannot follow the link (for example because it has insufficient privileges or the link points to a nonexistent file) the properties of the link itself will be used.

When the -H or -L options are in effect, any symbolic links listed as the argument of -newer will be dereferenced, and the timestamp will be taken from the file to which the symbolic link points. The same consideration applies to -newerXY, -anewer and -cnewer.

The -follow option has a similar effect to -L, though it takes effect at the point where it appears (that is, if -L is not used but -follow is, any symbolic links appearing after -follow on the command line will be dereferenced, and those before it will not).

Listing 37: *'find'* utility

–D debugoptions
Print diagnostic information; this can be helpful to diagnose problems with why find is not doing what you want. The list of debug options should be comma separated. Compati– bility of the debug options is not guaranteed between releases of findutils. For a com– plete list of valid debug options, see the output of find –D help. Valid debug options include

help Explain the debugging options

tree Show the expression tree in its original and optimised form.

stat Print messages as files are examined with the stat and lstat system calls. The find program tries to minimise such calls.

opt Prints diagnostic information relating to the optimisation of the expression tree; see the –O option.

rates Prints a summary indicating how often each predicate succeeded or failed.

–Olevel
Enables query optimisation. The find program reorders tests to speed up execution **while** preserving the overall effect; that is, predicates with side effects are not reordered relative to each other. The optimisations performed at each optimisation level are as follows.

0 Equivalent to optimisation level 1.

1 This is the **default** optimisation level and corresponds to the traditional behav– iour. Expressions are reordered so that tests based only on the names of files (**for** example –name and –regex) are performed first.

2 Any –type or –xtype tests are performed after any tests based only on the names of files, but before any tests that require information from the inode. On many modern versions of Unix, file types are returned by readdir() and so these predi– cates are faster to evaluate than predicates which need to stat the file first.

3 At this optimisation level, the full cost-based query optimiser is enabled. The order of tests is modified so that cheap (i.e. fast) tests are performed first and more expensive ones are performed later, **if** necessary. Within each cost band, predicates are evaluated earlier or later according to whether they are likely to succeed or not. For -o, predicates which are likely to succeed are evaluated earlier, and **for** -a, predicates which are likely to fail are evaluated earlier.

The cost-based optimiser has a fixed idea of how likely any given test is to succeed. In some cases the probability takes account of the specific nature of the test (**for** example, -type f is assumed to be more likely to succeed than -type c). The cost-based optimiser is currently being evaluated. If it does not actually improve the performance of find, it will be removed again. Conversely, optimisations that prove to be reliable, robust and effective may be enabled at lower optimisation levels over time. However, the **default** behaviour (i.e. optimisation level 1) will not be changed in the 4.3.x release series. The findutils test suite runs all the tests on find at each optimisation level and ensures that the result is the same.

EXPRESSIONS

The expression is made up of options (which affect overall operation rather than the processing of a specific file, and always **return** true), tests (which **return** a true or false value), and actions (which have side effects and **return** a true or false value), all separated by operators. -and is assumed where the operator is omitted.

If the expression contains no actions other than -prune, -print is performed on all files **for** which the expression is true.

OPTIONS

All options always **return** true. Except **for** -daystart, -follow and -regextype, the options affect all tests, including tests specified before the option. This is because the options are processed when the command line is parsed, **while** the tests don't do anything until files are examined. The -daystart, -follow and -regextype options are different in this respect, and have an effect only on tests which appear later in the command line. Therefore, for clarity, it is best to place them at the beginning of the expression. A warning is issued if you don't **do** this.

-d A synonym **for** -depth, **for** compatibility with FreeBSD, NetBSD, MacOS X and OpenBSD.

-daystart
Measure times (**for** -amin, -atime, -cmin, -ctime, -mmin, and -mtime) from the beginning of today rather than from 24 hours ago. This option only affects tests which appear later on the command line.

-depth Process each directory's contents before the directory itself. The -delete action also implies -depth.

-follow
Deprecated; use the -L option instead. Dereference symbolic links. Implies -noleaf. The -follow option affects only those tests which appear after it on the command line. Unless the -H or -L option has been specified, the position of the -follow option changes the behaviour of the -newer predicate; any files listed as the argument of -newer will be dereferenced if they are symbolic links. The same consideration applies to -newerXY, -anewer and -cnewer. Similarly, the -type predicate will always match against the type of the file that a symbolic link points to rather than the link itself. Using -follow causes the -lname and -ilname predicates always to return false.

-help, --help
Print a summary of the command-line usage of find and exit.

-ignore_readdir_race
Normally, find will emit an error message when it fails to stat a file. If you give this option and a file is deleted between the time find reads the name of the file from the directory and the time it tries to stat the file, no error message will be issued. This also applies to files or directories whose names are given on the command line. This option takes effect at the time the command line is read, which means that you cannot search one part of the filesystem with this option on and part of it with this option off (if you need to do that, you will need to issue two find commands instead, one with the option and one without it).

-maxdepth levels
Descend at most levels (a non-negative integer) levels of directories below the command line arguments. -maxdepth 0
means only apply the tests and actions to the command line arguments.

-mindepth levels
Do not apply any tests or actions at levels less than levels (a non-negative integer). -mindepth 1 means process all files except the command line arguments.

-mount Don't descend directories on other filesystems. An alternate name **for** -xdev, **for** compatibility with some other versions of find.

-noignore_readdir_race
Turns off the effect of -ignore_readdir_race.

-noleaf
Do not optimize by assuming that directories contain 2 fewer subdirectories than their hard link count. This option is needed when searching filesystems that **do** not follow the Unix directory-link convention, such as CD-ROM or MS-DOS filesystems or AFS volume mount points. Each directory on a normal Unix filesystem has at least 2 hard links: its name and its '.' entry. Additionally, its subdirectories (if any) each have a '..' entry linked to that directory. When find is examining a directory, after it has stated 2 fewer subdirectories than the directory's link count, it knows that the rest of the entries in the directory are non-directories ('leaf' files in the directory tree). If only the files' names need to be examined, there is no need to stat them; this gives a significant increase in search speed.

-regextype type
Changes the regular expression syntax understood by -regex and -iregex tests which occur later on the command line. Currently-implemented types are emacs (this is the default), posix-awk, posix-basic, posix-egrep and posix-extended.

-version, --version
Print the find version number and exit.

-warn, -nowarn
Turn warning messages on or off. These warnings apply only to the command line usage, not to any conditions that find might encounter when it searches directories. The default behaviour corresponds to -warn if standard input is a tty, and to -nowarn otherwise.

-xdev Don't descend directories on other filesystems.

TESTS
Some tests, **for** example -newerXY and -samefile, allow comparison between the file currently being examined and some reference file specified on the command line. When these tests are used, the interpretation of the reference file is determined by the options -H, -L and -P and any previous -follow, but the reference file is only examined once, at the time the command line is parsed. If the reference file cannot be examined (**for** example, the stat(2) system call fails **for** it), an error message is issued, and find exits with a nonzero status.

Numeric arguments can be specified as

+n **for** greater than n,

-n **for** less than n,

n **for** exactly n.

-amin n
File was last accessed n minutes ago.

-anewer file
File was last accessed more recently than file was modified. If file is a symbolic link and the -H option or the -L option is in effect, the access time of the file it points to is always used.

-atime n
File was last accessed n*24 hours ago. When find figures out how many 24-hour periods ago the file was last accessed, any fractional part is ignored, so to match -atime +1, a file has to have been accessed at least two days ago.

-cmin n
File's status was last changed n minutes ago.

-cnewer file
File's status was last changed more recently than file was modified. If file is a symbolic link and the -H option or the -L option is in effect, the status-change time of the file it points to is always used.

-ctime n
File's status was last changed n*24 hours ago. See the comments for -atime to understand how rounding affects the interpretation of file status change times.

-empty File is empty and is either a regular file or a directory.

-executable
Matches files which are executable and directories which are searchable (in a file name resolution sense). This takes into account access control lists and other permissions artefacts which the -perm test ignores. This test makes use of the access(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement access(2) in the client's kernel and so cannot make use of the UID mapping information held on the server. Because this test is based only on the result of the access(2) system call, there is no guarantee that a file **for** which this test succeeds can actually be executed.

-false Always false.

-fstype type
File is on a filesystem of type type. The valid filesystem types vary among different versions of Unix; an incomplete list of filesystem types that are accepted on some version of Unix or another is: ufs, 4.2, 4.3, nfs, tmp, mfs, S51K, S52K. You can use -printf with the %F directive to see the types of your filesystems.

-gid n File's numeric group ID is n.

-group gname
File belongs to group gname (numeric group ID allowed).

-ilname pattern
Like -lname, but the match is case insensitive. If the -L option or the -follow option is in effect, this test returns false unless the symbolic link is broken.

-iname pattern
Like -name, but the match is case insensitive. For example, the patterns 'fo*' and 'F??' match the file names 'Foo', 'FOO', 'foo', 'fOo', etc. In these patterns, unlike filename expansion by the shell, an initial '.' can be matched by '*'. That is, find -name *bar will match the file '.foobar'. Please note that you should quote patterns as a matter of course, otherwise the shell will expand any wildcard characters in them.

-inum n
File has inode number n. It is normally easier to use the -samefile test instead.

-ipath pattern
Behaves in the same way as -iwholename. This option is deprecated, so please do not use it.

-iregex pattern
Like -regex, but the match is case insensitive.

-iwholename pattern
Like -wholename, but the match is case insensitive.

-links n
File has n links.

-lname pattern
File is a symbolic link whose contents match shell pattern pattern. The metacharacters do not treat '/' or '.' specially. If the -L option or the -follow option is in effect, this test returns false unless the symbolic link is broken.

-mmin n
File's data was last modified n minutes ago.

-mtime n
File's data was last modified n*24 hours ago. See the comments for -atime to understand how rounding affects the interpretation of file modification times.

-name pattern
Base of file name (the path with the leading directories removed) matches shell pattern pattern. The metacharacters ('*', '?', and '[]') match a '.' at the start of the base name (this is a change in findutils-4.2.2; see section STANDARDS CONFORMANCE below). To ignore a directory and the files under it, use -prune; see an example in the description

of –path. Braces are not recognised as being special, despite the fact that some shells including Bash imbue braces with a special meaning in shell patterns. The filename matching is performed with the use of the fnmatch(3) library function. Don't forget to enclose the pattern in quotes in order to protect it from expansion by the shell.

–newer file
File was modified more recently than file. If file is a symbolic link and the –H option or the –L option is in effect, the modification time of the file it points to is always used.

–newerXY reference
Compares the timestamp of the current file with reference. The reference argument is normally the name of a file (and one of its timestamps is used **for** the comparison) but it may also be a string describing an absolute time. X and Y are placeholders **for** other letters, and these letters select which time belonging to how reference is used **for** the comparison.

a The access time of the file reference
B The birth time of the file reference
c The inode status change time of reference
m The modification time of the file reference
t reference is interpreted directly as a time

Some combinations are invalid; **for** example, it is invalid **for** X to be t. Some combinations are not implemented on all systems; **for** example B is not supported on all systems. If an invalid or unsupported combination of XY is specified, a fatal error results. Time specifications are interpreted as **for** the argument to the –d option of GNU date. If you try to use the birth time of a reference file, and the birth time cannot be determined, a fatal error message results. If you specify a test which refers to the birth time of files being examined, this test will fail **for** any files where the birth time is unknown.

–nogroup
No group corresponds to file's numeric group ID.

–nouser
No user corresponds to file's numeric user ID.

–path pattern
File name matches shell pattern pattern. The metacharacters **do** not treat '/' or '.' specially; so, **for** example,
find . –path "./sr*sc"
will print an entry **for** a directory called './src/misc' (if one exists). To ignore a whole directory tree, use –prune rather than checking every file in the tree. For example, to skip the directory 'src/emacs' and all files and directories under it, and print the names of the other files found, **do** something like this:
find . –path ./src/emacs –prune –o –print
Note that the pattern match test applies to the whole file name, starting from one of the start points named on the command line. It would only make sense to use an absolute path name here **if** the relevant start point is also an absolute path. This means that this command will never match anything:
find bar –path /foo/bar/myfile –print
The predicate –path is also supported by HP–UX find and will be in a forthcoming version of the POSIX standard.

–perm mode
File's permission bits are exactly mode (octal or symbolic). Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex mode string. For example –perm g=w will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set). It is more likely that you will want to use the '/' or '–' forms, for example –perm –g=w, which matches any file with group write permission. See the EXAMPLES section for some illustrative examples.

–perm –mode
All of the permission bits mode are set for the file. Symbolic modes are accepted in this form, and this is usually the way in which would want to use them. You must specify 'u', 'g' or 'o' **if** you use a symbolic mode. See the EXAMPLES section **for** some illustrative examples.

–perm /mode
Any of the permission bits mode are set **for** the file. Symbolic modes are accepted in this form. You must specify 'u', 'g' or 'o' if you use a symbolic mode. See the EXAMPLES section for some illustrative examples. If no permission bits in mode are set, this test matches any file (the idea here is to be consistent with the behaviour of –perm –000).

–perm +mode

Deprecated, old way of searching for files with any of the permission bits in mode set. You should use -perm /mode instead. Trying to use the '+' syntax with symbolic modes will yield surprising results. For example, '+u+x' is a valid symbolic mode (equivalent to +u,+x, i.e. 0111) and will therefore not be evaluated as -perm +mode but instead as the exact mode specifier -perm mode and so it matches files with exact permissions 0111 instead of files with any execute bit set. If you found this paragraph confusing, you're not alone - just use -perm /mode. This form of the -perm test is deprecated because the POSIX specification requires the interpretation of a leading '+' as being part of a symbolic mode, and so we switched to using '/' instead.

-readable
Matches files which are readable. This takes into account access control lists and other permissions artefacts which the -perm test ignores. This test makes use of the access(2) system call, and so can be fooled by NFS servers which **do** UID mapping (or root-squashing), since many systems implement access(2) in the client's kernel and so cannot make use of the UID mapping information held on the server.

-regex pattern
File name matches regular expression pattern. This is a match on the whole path, not a search. For example, to match a file named './fubar3', you can use the regular expression '.*bar.' or '.*b.*3', but not 'f.*r3'. The regular expressions understood by find are by default Emacs Regular Expressions, but this can be changed with the -regextype option.

-samefile name
File refers to the same inode as name. When -L is in effect, this can include symbolic links.

-size n[cwbkMG]
File uses n units of space. The following suffixes can be used:

'b' **for** 512-byte blocks (this is the **default if** no suffix is used)

'c' for bytes

'w' **for** two-byte words

'k' for Kilobytes (units of 1024 bytes)

'M' **for** Megabytes (units of 1048576 bytes)

'G' for Gigabytes (units of 1073741824 bytes)

The size does not count indirect blocks, but it does count blocks in sparse files that are not actually allocated. Bear in mind that the '%k' and '%b' format specifiers of -printf handle sparse files differently. The 'b' suffix always denotes 512-byte blocks and never 1 Kilobyte blocks, which is different to the behaviour of -ls.

-true Always true.

-type c
File is of type c:

b block (buffered) special

c character (unbuffered) special

d directory

p named pipe (FIFO)

f regular file

l symbolic link; this is never true **if** the -L option or the -follow option is in effect, unless the symbolic link is broken. If you want to search **for** symbolic links when -L is in effect, use -xtype.

s socket

D door (Solaris)

-uid n File's numeric user ID is n.

-used n
File was last accessed n days after its status was last changed.

-user uname
File is owned by user uname (numeric user ID allowed).

```
-wholename pattern
       See -path.  This alternative is less portable than -path.


-writable
       Matches files which are writable.  This takes into account access control lists and
       other permissions artefacts which the -perm test ignores.  This test makes use of the
       access(2) system call, and so can be fooled by NFS servers which do UID mapping (or
       root-squashing), since many systems implement access(2) in the client's kernel and so
       cannot make use of the UID mapping information held on the server.


-xtype c
       The same as -type unless the file is a symbolic link.  For symbolic links: if the -H or
       -P option was specified, true if the file is a link to a file of type c; if the -L
       option has been given, true if c is 'l'.  In other words, for symbolic links, -xtype
       checks the type of the file that -type does not check.


ACTIONS
-delete
       Delete files; true if removal succeeded.  If the removal failed, an error message is
       issued.  If -delete fails, find's exit status will be nonzero (when it eventually
       exits).  Use of -delete automatically turns on the -depth option.

       Warnings: Don't forget that the find command line is evaluated as an expression, so
       putting -delete first will make find try to delete everything below the starting points
       you specified.  When testing a find command line that you later intend to use with
       -delete, you should explicitly specify -depth in order to avoid later surprises.
       Because -delete implies -depth, you cannot usefully use -prune and -delete together.


-exec command ;
       Execute command; true if 0 status is returned.  All following arguments to find are
       taken to be arguments to the command until an argument consisting of ';' is encountered.
       The string '\{\}' is replaced by the current file name being processed everywhere it
       occurs in the arguments to the command, not just in arguments where it is alone, as in
       some versions of find.  Both of these constructions might need to be escaped (with a
       '\') or quoted to protect them from expansion by the shell.  See the EXAMPLES section
       for examples of the use of the -exec option.  The specified command is run once for each
       matched file.  The command is executed in the starting directory.  There are unavoid-
       able security problems surrounding use of the -exec action; you should use the -execdir
       option instead.


-exec command \{\} +
       This variant of the -exec action runs the specified command on the selected files, but
       the command line is built by appending each selected file name at the end; the total
       number of invocations of the command will be much less than the number of matched files.
       The command line is built in much the same way that xargs builds its command lines.
       Only one instance of '\{\}' is allowed within the command.  The command is executed in the
       starting directory.


-execdir command ;

-execdir command \{\} +
       Like -exec, but the specified command is run from the subdirectory containing the
       matched file, which is not normally the directory in which you started find.  This a
       much more secure method for invoking commands, as it avoids race conditions during reso-
       lution of the paths to the matched files.  As with the -exec action, the '+' form of
       -execdir will build a command line to process more than one matched file, but any given
       invocation of command will only list files that exist in the same subdirectory.  If you
       use this option, you must ensure that your \$PATH environment variable does not reference
       '.'; otherwise, an attacker can run any commands they like by leaving an appropriately-
       named file in a directory in which you will run -execdir.  The same applies to having
       entries in \$PATH which are empty or which are not absolute directory names.


-fls file
       True; like -ls but write to file like -fprint.  The output file is always created, even
       if the predicate is never matched.  See the UNUSUAL FILENAMES section for information
       about how unusual characters in filenames are handled.


-fprint file
       True; print the full file name into file file.  If file does not exist when find is run,
       it is created; if it does exist, it is truncated.  The file names ''/dev/stdout'' and
       ''/dev/stderr'' are handled specially; they refer to the standard output and standard
       error output, respectively.  The output file is always created, even if the predicate is
       never matched.  See the UNUSUAL FILENAMES section for information about how unusual
       characters in filenames are handled.


-fprint0 file
       True; like -print0 but write to file like -fprint.  The output file is always created,
```

even **if** the predicate is never matched. See the UNUSUAL FILENAMES section **for** information about how unusual characters in filenames are handled.

-fprintf file format
True; like -printf but write to file like -fprint. The output file is always created, even **if** the predicate is never matched. See the UNUSUAL FILENAMES section **for** information about how unusual characters in filenames are handled.

-ls True; list current file in ls -dils format on standard output. The block counts are of 1K blocks, unless the environment variable POSIXLY_CORRECT is set, in which **case** 512-byte blocks are used. See the UNUSUAL FILENAMES section **for** information about how unusual characters in filenames are handled.

-ok command ;
Like -exec but ask the user first. If the user agrees, run the command. Otherwise just **return** false. If the command is run, its standard input is redirected from /dev/null.

The response to the prompt is matched against a pair of regular expressions to determine **if** it is an affirmative or negative response. This regular expression is obtained from the system **if** the 'POSIXLY_CORRECT' environment variable is set, or otherwise from find's message translations. If the system has no suitable definition, find's own definition will be used. In either case, the interpretation of the regular expression itself will be affected by the environment variables 'LC_CTYPE' (character classes) and 'LC_COLLATE' (character ranges and equivalence classes).

-okdir command ;
Like -execdir but ask the user first in the same way as for -ok. If the user does not agree, just return false. If the command is run, its standard input is redirected from /dev/null.

-print True; print the full file name on the standard output, followed by a newline. If you are piping the output of find into another program and there is the faintest possibility that the files which you are searching for might contain a newline, then you should seriously consider using the -print0 option instead of -print. See the UNUSUAL FILENAMES section for information about how unusual characters in filenames are handled.

-print0
True; print the full file name on the standard output, followed by a null character (instead of the newline character that -print uses). This allows file names that contain newlines or other types of white space to be correctly interpreted by programs that process the find output. This option corresponds to the -0 option of xargs.

-printf format
True; print format on the standard output, interpreting '\' escapes and '%' directives. Field widths and precisions can be specified as with the 'printf' C function. Please note that many of the fields are printed as %s rather than %d, and this may mean that flags don't work as you might expect. This also means that the '-' flag does work (it forces fields to be left-aligned). Unlike -print, -printf does not add a newline at the end of the string. The escapes and directives are:

\a Alarm bell.

\b Backspace.

\c Stop printing from this format immediately and flush the output.

\f Form feed.

\n Newline.

\r Carriage return.

\t Horizontal tab.

\v Vertical tab.

\0 ASCII NUL.

\\ A literal backslash ('\').

\NNN The character whose ASCII code is NNN (octal).

A '\' character followed by any other character is treated as an ordinary character, so they both are printed.

%% A literal percent sign.

\%a File's last access time in the format returned by the C 'ctime' function.

\%Ak File's last access time in the format specified by k, which is either '@' or a directive for the C 'strftime' function. The possible values for k are listed below; some of them might not be available on all systems, due to differences in 'strftime' between systems.

@ seconds since Jan. 1, 1970, 00:00 GMT, with fractional part.

Time fields:

H hour (00..23)

I hour (01..12)

k hour (0..23)

l hour (1..12)

M minute (00..59)

p locale's AM or PM

r time, 12-hour (hh:mm:ss [AP]M)

S Second (00.00 .. 61.00). There is a fractional part.

T time, 24-hour (hh:mm:ss)

+ Date and time, separated by '+', for example '2004-04-28+22:22:05.0'. This is a GNU extension. The time is given in the current timezone (which may be affected by setting the TZ environment variable). The seconds field includes a fractional part.

X locale's time representation (H:M:S)

Z time zone (e.g., EDT), or nothing if no time zone is determinable

Date fields:

a locale's abbreviated weekday name (Sun..Sat)

A locale's full weekday name, variable length (Sunday..Saturday)

b locale's abbreviated month name (Jan..Dec)

B locale's full month name, variable length (January..December)

c locale's date and time (Sat Nov 04 12:02:33 EST 1989). The format is the same as for ctime(3) and so to preserve compatibility with that format, there is no fractional part in the seconds field.

d day of month (01..31)

D date (mm/dd/yy)

h same as b

j day of year (001..366)

m month (01..12)

U week number of year with Sunday as first day of week (00..53)

w day of week (0..6)

W week number of year with Monday as first day of week (00..53)

x locale's date representation (mm/dd/yy)

y last two digits of year (00..99)

Y year (1970...)

\%b The amount of disk space used for this file in 512-byte blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than \%s/512, but it can also be smaller if the file is a sparse file.

\%c File's last status change time in the format returned by the C 'ctime' function.

\%Ck File's last status change time in the format specified by k, which is the same as for %A.

\%d File's depth in the directory tree; 0 means the file is a command line argument.

\%D The device number on which the file exists (the st_dev field of struct stat), in decimal.

\%f File's name with any leading directories removed (only the last element).

\%F Type of the filesystem the file is on; this value can be used **for** -fstype.

\%g File's group name, or numeric group ID if the group has no name.

\%G File's numeric group ID.

\%h Leading directories of file's name (all but the last element). If the file name contains no slashes (since it is in the current directory) the \%h specifier expands to ".".

\%H Command line argument under which file was found.

\%i File's inode number (in decimal).

\%k The amount of disk space used **for** this file in 1K blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than \%s/1024, but it can also be smaller **if** the file is a sparse file.

\%l Object of symbolic link (empty string **if** file is not a symbolic link).

\%m File's permission bits (in octal). This option uses the 'traditional' numbers which most Unix implementations use, but **if** your particular implementation uses an unusual ordering of octal permissions bits, you will see a difference between the actual value of the file's mode and the output of \%m. Normally you will want to have a leading zero on this number, and to do this, you should use the # flag (as in, for example, '\%#m').

\%M File's permissions (in symbolic form, as for ls). This directive is supported in findutils 4.2.5 and later.

\%n Number of hard links to file.

\%p File's name.

\%P File's name with the name of the command line argument under which it was found removed.

\%s File's size in bytes.

\%S File's sparseness. This is calculated as (BLOCKSIZE*st_blocks / st_size). The exact value you will get for an ordinary file of a certain length is system-dependent. However, normally sparse files will have values less than 1.0, and files which use indirect blocks may have a value which is greater than 1.0. The value used for BLOCKSIZE is system-dependent, but is usually 512 bytes. If the file size is zero, the value printed is undefined. On systems which lack support for st_blocks, a file's sparseness is assumed to be 1.0.

\%t File's last modification time in the format returned by the C 'ctime' function.

\%Tk File's last modification time in the format specified by k, which is the same as for %A.

\%u File's user name, or numeric user ID **if** the user has no name.

\%U File's numeric user ID.

\%y File's type (like in ls -l), U=unknown type (shouldn't happen)

\%Y File's type (like %y), plus follow symlinks: L=loop, N=nonexistent

A '\%' character followed by any other character is discarded, but the other character is printed (don't rely on this, as further format characters may be introduced). A '%' at the end of the format argument causes undefined behaviour since there is no following character. In some locales, it may hide your door keys, while in others it may remove the final page from the novel you are reading.

The \%m and \%d directives support the # , 0 and + flags, but the other directives do not, even if they print numbers. Numeric directives that do not support these flags include G, U, b, D, k and n. The '-' format flag is supported and changes the alignment of a field from right-justified (which is the **default**) to left-justified.

See the UNUSUAL FILENAMES section **for** information about how unusual characters in file-names are handled.

-prune True; **if** the file is a directory, **do** not descend into it. If -depth is given, false; no effect. Because -delete implies -depth, you cannot usefully use -prune and -delete together.

-quit Exit immediately. No child processes will be left running, but no more paths specified on the command line will be processed. For example, find /tmp/foo /tmp/bar -print -quit will print only /tmp/foo. Any command lines which have been built up with -execdir ...

{} + will be invoked before find exits. The exit status may or may not be zero, depending on whether an error has already occurred.

UNUSUAL FILENAMES

Many of the actions of find result in the printing of data which is under the control of other users. This includes file names, sizes, modification times and so forth. File names are a potential problem since they can contain any character except '\0' and '/'. Unusual characters in file names can **do** unexpected and often undesirable things to your terminal (**for** example, changing the settings of your function keys on some terminals). Unusual characters are handled differently by various actions, as described below.

-print0, -fprint0
Always print the exact filename, unchanged, even **if** the output is going to a terminal.

-ls, -fls
Unusual characters are always escaped. White space, backslash, and **double** quote characters are printed using C-style escaping (**for** example '\f', '\"'). Other unusual characters are printed using an octal escape. Other printable characters (**for** -ls and -fls these are the characters between octal 041 and 0176) are printed as-is.

-printf, -fprintf
If the output is not going to a terminal, it is printed as-is. Otherwise, the result depends on which directive is in use. The directives %D, %F, %g, %G, %H, %Y, and %y expand to values which are not under control of files' owners, and so are printed as-is. The directives %a, %b, %c, %d, %i, %k, %m, %M, %n, %s, %t, %u and %U have values which are under the control of files' owners but which cannot be used to send arbitrary data to the terminal, and so these are printed as-is. The directives %f, %h, %l, %p and %P are quoted. This quoting is performed in the same way as **for** GNU ls. This is not the same quoting mechanism as the one used **for** -ls and -fls. If you are able to decide what format to use **for** the output of find then it is normally better to use '\0' as a terminator than to use newline, as file names can contain white space and newline characters. The setting of the 'LC_CTYPE' environment variable is used to determine which characters need to be quoted.

-print, -fprint
Quoting is handled in the same way as **for** -printf and -fprintf. If you are using find in a script or in a situation where the matched files might have arbitrary names, you should consider using -print0 instead of -print.

The -ok and -okdir actions print the current filename as-is. This may change in a future release.

OPERATORS

Listed in order of decreasing precedence:

(expr)
Force precedence. Since parentheses are special to the shell, you will normally need to quote them. Many of the examples in this manual page use backslashes **for** this purpose: '\(...\)' instead of '(...)'.

! expr True **if** expr is false. This character will also usually need protection from interpretation by the shell.

-not expr
Same as ! expr, but not POSIX compliant.

expr1 expr2
Two expressions in a row are taken to be joined with an implied "and"; expr2 is not evaluated **if** expr1 is false.

expr1 -a expr2
Same as expr1 expr2.

expr1 -and expr2
Same as expr1 expr2, but not POSIX compliant.

expr1 -o expr2
Or; expr2 is not evaluated **if** expr1 is true.

expr1 -or expr2
Same as expr1 -o expr2, but not POSIX compliant.

expr1 , expr2
List; both expr1 and expr2 are always evaluated. The value of expr1 is discarded; the value of the list is the value of expr2. The comma operator can be useful **for** searching **for** several different types of thing, but traversing the filesystem hierarchy only once. The –fprintf action can be used to list the various matched items into several different output files.

STANDARDS CONFORMANCE

For closest compliance to the POSIX standard, you should set the POSIXLY_CORRECT environment variable. The following options are specified in the POSIX standard (IEEE Std 1003.1, 2003 Edition):

–H This option is supported.

–L This option is supported.

–name This option is supported, but POSIX conformance depends on the POSIX conformance of the system's fnmatch(3) library function. As of findutils –4.2.2, shell metacharacters ('*', '?' or '[]' **for** example) will match a leading '.', because IEEE PASC interpretation 126 requires this. This is a change from previous versions of findutils.

–type Supported. POSIX specifies 'b', 'c', 'd', 'l', 'p', 'f' and 's'. GNU find also supports 'D', representing a Door, where the OS provides these.

–ok Supported. Interpretation of the response is according to the "yes" and "no" patterns selected by setting the 'LC_MESSAGES' environment variable. When the 'POSIXLY_CORRECT' environment variable is set, these patterns are taken system's definition of a positive (yes) or negative (no) response. See the system's documentation for nl_langinfo(3), in particular YESEXPR and NOEXPR. When 'POSIXLY_CORRECT' is not set, the patterns are instead taken from find's own message catalogue.

–newer Supported. If the file specified is a symbolic link, it is always dereferenced. This is a change from previous behaviour, which used to take the relevant time from the symbolic link; see the HISTORY section below.

–perm Supported. If the POSIXLY_CORRECT environment variable is not set, some mode arguments (for example +a+x) which are not valid in POSIX are supported for backward–compatibility.

Other predicates
The predicates –atime, –ctime, –depth, –group, –links, –mtime, –nogroup, –nouser, –print, –prune, –size, –user and –xdev are all supported.

The POSIX standard specifies parentheses '(', ')', negation '!' and the 'and' and 'or' operators (–a, –o).

All other options, predicates, expressions and so forth are extensions beyond the POSIX standard. Many of these extensions are not unique to GNU find, however.

The POSIX standard requires that find detects loops:

> The find utility shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file encountered. When it detects an infinite loop, find shall write a diagnostic message to standard error and shall either recover its position in the hierarchy or terminate.

GNU find complies with these requirements. The link count of directories which contain entries which are hard links to an ancestor will often be lower than they otherwise should be. This can mean that GNU find will sometimes optimise away the visiting of a subdirectory which is actually a link to an ancestor. Since find does not actually enter such a subdirectory, it is allowed to avoid emitting a diagnostic message. Although this behaviour may be somewhat confusing, it is unlikely that anybody actually depends on this behaviour. If the leaf optimisation has been turned off with –noleaf, the directory entry will always be examined and the diagnostic message will be issued where it is appropriate. Symbolic links cannot be used to create filesystem cycles as such, but **if** the –L option or the –follow option is in use, a diagnostic message is issued when find encounters a loop of symbolic links. As with loops containing hard links, the leaf optimisation will often mean that find knows that it doesn't need to call stat() or chdir() on the symbolic link, so this diagnostic is frequently not necessary.

The –d option is supported for compatibility with various BSD systems, but you should use the POSIX–compliant option –depth instead.

The POSIXLY_CORRECT environment variable does not affect the behaviour of the –regex or –iregex tests because those tests aren't specified in the POSIX standard.

ENVIRONMENT VARIABLES

LANG Provides a **default** value **for** the internationalization variables that are unset or null.

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE
The POSIX standard specifies that this variable affects the pattern matching to be used **for** the -name option. GNU find uses the fnmatch(3) library function, and so support **for** 'LC_COLLATE' depends on the system library. This variable also affects the interpretation of the response to -ok; while the 'LC_MESSAGES' variable selects the actual pattern used to interpret the response to -ok, the interpretation of any bracket expressions in the pattern will be affected by 'LC_COLLATE'.

LC_CTYPE
This variable affects the treatment of character classes used in regular expressions and also with the -name test, if the system's fnmatch(3) library function supports this. This variable also affects the interpretation of any character classes in the regular expressions used to interpret the response to the prompt issued by -ok. The 'LC_CTYPE' environment variable will also affect which characters are considered to be unprintable when filenames are printed; see the section UNUSUAL FILENAMES.

LC_MESSAGES
Determines the locale to be used for internationalised messages. If the 'POSIXLY_CORRECT' environment variable is set, this also determines the interpretation of the response to the prompt made by the -ok action.

NLSPATH
Determines the location of the internationalisation message catalogues.

PATH Affects the directories which are searched to find the executables invoked by -exec, -execdir, -ok and -okdir.

POSIXLY_CORRECT
Determines the block size used by -ls and -fls. If POSIXLY_CORRECT is set, blocks are units of 512 bytes. Otherwise they are units of 1024 bytes.

Setting this variable also turns off warning messages (that is, implies -nowarn) by **default**, because POSIX requires that apart from the output **for** -ok, all messages printed on stderr are diagnostics and must result in a non-zero exit status.

When POSIXLY_CORRECT is not set, -perm +zzz is treated just like -perm /zzz **if** +zzz is not a valid symbolic mode. When POSIXLY_CORRECT is set, such constructs are treated as an error.

When POSIXLY_CORRECT is set, the response to the prompt made by the -ok action is interpreted according to the system's message catalogue, as opposed to according to find's own message translations.

TZ Affects the time zone used **for** some of the time-related format directives of -printf and -fprintf.

EXAMPLES

```
find /tmp -name core -type f -print | xargs /bin/rm -f
```

Find files named core in or below the directory /tmp and delete them. Note that this will work incorrectly **if** there are any filenames containing newlines, single or **double** quotes, or spaces.

```
find /tmp -name core -type f -print0 | xargs -0 /bin/rm -f
```

Find files named core in or below the directory /tmp and delete them, processing filenames in such a way that file or directory names containing single or **double** quotes, spaces or newlines are correctly handled. The -name test comes before the -type test in order to avoid having to call stat(2) on every file.

```
find . -type f -exec file '{}' \;
```

Runs 'file' on every file in or below the current directory. Notice that the braces are enclosed in single quote marks to protect them from interpretation as shell script punctuation. The semicolon is similarly protected by the use of a backslash, though single quotes could have been used in that case also.

```
find / \
\( -perm -4000 -fprintf /root/suid.txt %#m %u %p\n \) , \
\( -size +100M -fprintf /root/big.txt %-10s %p\n \)
```

Traverse the filesystem just once, listing setuid files and directories into /root/suid.txt and large files into /root/big.txt.

```
find \$HOME -mtime 0
```

Search for files in your home directory which have been modified in the last twenty-four hours. This command works this way because the time since each file was last modified is divided by 24 hours and any remainder is discarded. That means that to match -mtime 0, a file will have to have a modification in the past which is less than 24 hours ago.

```
find /sbin /usr/sbin -executable \! -readable -print
```

Search for files which are executable but not readable.

```
find . -perm 664
```

Search for files which have read and write permission for their owner, and group, but which other users can read but not write to. Files which meet these criteria but have other permissions bits set (for example if someone can execute the file) will not be matched.

```
find . -perm -664
```

Search for files which have read and write permission for their owner and group, and which other users can read, without regard to the presence of any extra permission bits (for example the executable bit). This will match a file which has mode 0777, for example.

```
find . -perm /222
```

Search for files which are writable by somebody (their owner, or their group, or anybody else).

```
find . -perm /220
find . -perm /u+w,g+w
find . -perm /u=w,g=w
```

All three of these commands do the same thing, but the first one uses the octal representation of the file mode, and the other two use the symbolic form. These commands all search for files which are writable by either their owner or their group. The files don't have to be writable by both the owner and group to be matched; either will **do**.

```
find . -perm -220
find . -perm -g+w,u+w
```

Both these commands **do** the same thing; search **for** files which are writable by both their owner and their group.

```
find . -perm -444 -perm /222 ! -perm /111
find . -perm -a+r -perm /a+w ! -perm /a+x
```

These two commands both search **for** files that are readable **for** everybody (-perm -444 or -perm -a+r), have at least one write bit set (-perm /222 or -perm /a+w) but are not executable **for** anybody (! -perm /111 and ! -perm /a+x respectively).

```
cd /source-dir
find . -name .snapshot -prune -o \( \! -name *~ -print0 \)|
cpio -pmd0 /dest-dir
```

This command copies the contents of /source-dir to /dest-dir, but omits files and directories named .snapshot (and anything in them). It also omits files or directories whose name ends in ~, but not their contents. The construct -prune -o \(... -print0 \) is quite common. The idea here is that the expression before -prune matches things which are to be pruned. However, the -prune action itself returns true, so the following -o ensures that the right hand side is evaluated only **for** those directories which didn't get pruned (the contents of the pruned directories are not even visited, so their contents are irrelevant). The expression on the right hand side of the -o is in parentheses only for clarity. It emphasises that the -print0 action takes place only for things that didn't have -prune applied to them. Because the **default** 'and' condition between tests binds more tightly than -o, this is the default anyway, but the parentheses help to show what is going on.

```
find repo/ -exec test -d {}/.svn -o -d {}/.git -o -d {}/CVS ; \
-print -prune
```

Given the following directory of projects and their associated SCM administrative directories, perform an efficient search for the projects' roots:

```
repo/project1/CVS
repo/gnu/project2/.svn
repo/gnu/project3/.svn
repo/gnu/project3/src/.svn
repo/project4/.git
```

```
       In this example, -prune prevents unnecessary descent into directories that have already been
       discovered (for example we do not search project3/src because we already found project3/.svn),
       but ensures sibling directories (project2 and project3) are found.


EXIT STATUS
       find exits with status 0 if all files are processed successfully, greater than 0 if errors
       occur.  This is deliberately a very broad description, but if the return value is non-zero,
       you should not rely on the correctness of the results of find.


SEE ALSO
       locate(1), locatedb(5), updatedb(1), xargs(1), chmod(1), fnmatch(3), regex(7), stat(2),
       lstat(2), ls(1), printf(3), strftime(3), ctime(3), Finding Files (on-line in Info, or printed).

HISTORY
       As of findutils-4.2.2, shell metacharacters ('*', '?' or '[]' for example) used in filename
       patterns will match a leading '.', because IEEE POSIX interpretation 126 requires this.

       The syntax -perm +MODE was deprecated in findutils-4.2.21, in favour of -perm /MODE.  As of
       findutils-4.3.3, -perm /000 now matches all files instead of none.

       Nanosecond-resolution timestamps were implemented in findutils-4.3.3.

       As of findutils-4.3.11, the -delete action sets find's exit status to a nonzero value when it
       fails.  However, find will not exit immediately.  Previously, find's exit status was unaffected
       by the failure of -delete.

       Feature                 Added in    Also occurs in
       -newerXY                4.3.3       BSD
       -D                      4.3.1
       -O                      4.3.1
       -readable               4.3.0
       -writable               4.3.0
       -executable             4.3.0
       -regextype              4.2.24
       -exec ... +             4.2.12      POSIX
       -execdir                4.2.12      BSD
       -okdir                  4.2.12
       -samefile               4.2.11
       -H                      4.2.5       POSIX
       -L                      4.2.5       POSIX
       -P                      4.2.5       BSD
       -delete                 4.2.3
       -quit                   4.2.3
       -d                      4.2.3       BSD
       -wholename              4.2.0
       -iwholename             4.2.0
       -ignore_readdir_race    4.2.0
       -fls                    4.0
       -ilname                 3.8
       -iname                  3.8

       -ipath                  3.8
       -iregex                 3.8

NON-BUGS
       \$ find . -name *.c -print
       find: paths must precede expression
       Usage: find [-H] [-L] [-P] [-Olevel] [-D help|tree|search|stat|rates|opt|exec] [path...] [expression]

       This happens because *.c has been expanded by the shell resulting in find actually receiving a
       command line like this:

       find . -name bigram.c code.c frcode.c locate.c -print

       That command is of course not going to work.  Instead of doing things this way, you should
       enclose the pattern in quotes or escape the wildcard:
        $ find . -name \*.c -print
        $

BUGS
       There are security problems inherent in the behaviour that the POSIX standard specifies for
       find, which therefore cannot be fixed.  For example, the -exec action is inherently insecure,
       and -execdir should be used instead.  Please see Finding Files for more information.

       The environment variable LC_COLLATE has no effect on the -ok action.

       The best way to report a bug is to use the form at http://savannah.gnu.org/bugs/?group=findu-
       tils.  The reason for this is that you will then be able to track progress in fixing the prob-
       lem.  Other comments about find(1) and about the findutils package in general can be sent to
       the bug-findutils mailing list.  To join the list, send email to bug-findutils-request@gnu.org.

                                                                                              FIND(1)
```

22.5 advanced Linux Commands

Linux Commands???

22.5.1 usermod - Administration

Listing 38: man 8 usermod

```
NAME
       usermod - modify a user account
SYNOPSIS
       usermod [options] LOGIN
DESCRIPTION
       The usermod command modifies the system account files to reflect the changes
           that are specified
       on the command line.
OPTIONS
       The options which apply to the usermod command are:

       -a, --append
           Add the user to the supplementary group(s). Use only with the -G option.

       -c, --comment COMMENT
           The new value of the user's password file comment field. It is normally
               modified using the
           chfn(1) utility.

       -d, --home HOME_DIR
           The user's new login directory.

           If the -m option is given, the contents of the current home directory will
               be moved to the
           new home directory, which is created if it does not already exist.

       -e, --expiredate EXPIRE_DATE
           The date on which the user account will be disabled. The date is specified
               in the format
           YYYY-MM-DD.

           An empty EXPIRE_DATE argument will disable the expiration of the account.

           This option requires a /etc/shadow file. A /etc/shadow entry will be
               created if there were
           none.

       -f, --inactive INACTIVE
           The number of days after a password expires until the account is
               permanently disabled.

           A value of 0 disables the account as soon as the password has expired, and
               a value of -1
           disables the feature.

           This option requires a /etc/shadow file. A /etc/shadow entry will be
               created if there were
           none.

       -g, --gid GROUP
           The group name or number of the user's new initial login group. The group
               must exist.

           Any file from the user's home directory owned by the previous primary group
                of the user
           will be owned by this new group.

           The group ownership of files outside of the user's home directory must be
               fixed manually.

       -G, --groups GROUP1[,GROUP2,...[,GROUPN]]]
           A list of supplementary groups which the user is also a member of. Each
               group is separated
           from the next by a comma, with no intervening whitespace. The groups are
               subject to the
```

```
           same restrictions as the group given with the -g option.

           If the user is currently a member of a group which is not listed, the user
               will be removed
           from the group. This behaviour can be changed via the -a option, which
               appends the user to
           the current supplementary group list.

       -l, --login NEW_LOGIN
           The name of the user will be changed from LOGIN to NEW_LOGIN. Nothing else
               is changed. In
           particular, the user's home directory or mail spool should probably be
               renamed manually to
           reflect the new login name.

       -L, --lock
           Lock a user's password. This puts a '!' in front of the encrypted password,
                effectively
           disabling the password. You can't use this option with -p or -U.

           Note: if you wish to lock the account (not only access with a password),
               you should also
           set the EXPIRE_DATE to 1.

       -m, --move-home
           Move the content of the user's home directory to the new location.

           This option is only valid in combination with the -d (or --home) option.


           usermod will try to adapt the ownership of the files and to copy the modes,
                ACL and
           extended attributes, but manual changes might be needed afterwards.

       -o, --non-unique
           When used with the -u option, this option allows to change the user ID to a
                non-unique
           value.

       -p, --password PASSWORD
           The encrypted password, as returned by crypt(3).


           Note: This option is not recommended because the password (or encrypted
               password) will be
           visible by users listing the processes.

           The password will be written in the local /etc/passwd or /etc/shadow file.
               This might
           differ from the password database configured in your PAM configuration.

           You should make sure the password respects the system's password policy.

       -R, --root CHROOT_DIR
           Apply changes in the CHROOT_DIR directory and use the configuration files
               from the
           CHROOT_DIR directory.

       -s, --shell SHELL
           The name of the user's new login shell. Setting this field to blank causes
               the system to
           select the default login shell.

       -u, --uid UID
           The new numerical value of the user's ID.

           This value must be unique, unless the -o option is used. The value must be
               non-negative.

           The user's mailbox, and any files which the user owns and which are located
                in the user's
           home directory will have the file user ID changed automatically.
```

```
            The ownership of files outside of the user's home directory must be fixed
                manually.

            No checks will be performed with regard to the UID_MIN, UID_MAX,
                SYS_UID_MIN, or
            SYS_UID_MAX from /etc/login.defs.

        -U, --unlock
            Unlock a user's password. This removes the '!' in front of the encrypted
                password. You
            can't use this option with -p or -L.

            Note: if you wish to unlock the account (not only access with a password),
                you should also
            set the EXPIRE_DATE (for example to 99999, or to the EXPIRE value from
            /etc/default/useradd).

        -Z, --selinux-user SEUSER
            The new SELinux user for the user's login.

            A blank SEUSER will remove the SELinux user mapping for user LOGIN (if any)
                .

CAVEATS
        You must make certain that the named user is not executing any processes when
            this command is
        being executed if the user's numerical user ID, the user's name, or the user's
            home directory
        is being changed.  usermod checks this on Linux, but only check if the user is
            logged in
        according to utmp on other architectures.

        You must change the owner of any crontab files or at jobs manually.

        You must make any changes involving NIS on the NIS server.

CONFIGURATION
        The following configuration variables in /etc/login.defs change the behavior of
             this tool:

        MAIL_DIR (string)
            The mail spool directory. This is needed to manipulate the mailbox when its
                 corresponding
            user account is modified or deleted. If not specified, a compile-time
                default is used.

        MAIL_FILE (string)
            Defines the location of the users mail spool files relatively to their home
                 directory.

        The MAIL_DIR and MAIL_FILE variables are used by useradd, usermod, and userdel
            to create, move,
        or delete the user's mail spool.

        MAX_MEMBERS_PER_GROUP (number)
            Maximum members per group entry. When the maximum is reached, a new group
                entry (line) is
            started in /etc/group (with the same name, same password, and same GID).

            The default value is 0, meaning that there are no limits in the number of
                members in a
            group.

            This feature (split group) permits to limit the length of lines in the
                group file. This is
            useful to make sure that lines for NIS groups are not larger than 1024
                characters.

            If you need to enforce such limit, you can use 25.

            Note: split groups may not be supported by all tools (even in the Shadow
                toolsuite). You
            should not use this variable unless you really need it.
```

```
FILES
       /etc/group
           Group account information.

       /etc/gshadow
           Secure group account information.

       /etc/login.defs
           Shadow password suite configuration.

       /etc/passwd
           User account information.

       /etc/shadow
           Secure user account information.

SEE ALSO
       chfn(1), chsh(1), passwd(1), crypt(3), gpasswd(8), groupadd(8), groupdel(8),
           groupmod(8),
       login.defs(5), useradd(8), userdel(8).

shadow-utils 4.1.5.1            05/25/2012              USERMOD(8)
```

22.5.2 Masquerading Gateway (Sc7)

Annahme: *'MeinPC'* sei über das Netzwerkdevice *"wlan0"* ins Internet verbunden, zB. im *'HTL-S'*, und man möchte seine drei *Raspberry Pi 3B* mit den IPs 169.254.3.3, 169.254.3.4 und 169.254.3.5 per *"masquerading"* über *'MeinPC'* als Gateway mit einbinden:

Listing 39: Masquerading auf 'MeinPC'

```
   # (siehe 'man ip-address')
   # das erzeugt auch die 'route' ('ip ro add 169.254.3.0/24 dev eth0'):
/sbin/ip a add 169.254.3.0/24 dev eth0

/sbin/iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE

echo "1" > /proc/sys/net/ipv4/ip_forward
   # auch '/sbin/sysctl -w net.ipv4.ip_forward=1'
```

Die Raspi müssen von dieser neuen Route wissen, um sie zu verwenden:

Listing 40: Route auf 'MeinRaspi1'

```
   # (siehe 'man ip-route')
/sbin/ip ro del default

/sbin/ip ro add default via MeinPC dev eth0
```

22.5.3 sshfs

s.Kap.22.12 S.393

22.5.4 cgroups

cgroups

Listing 41: man 1 systemd-cgls

```
SYSTEMD-CGLS(1)                                systemd-cgls                                SYSTEMD-CGLS
    (1)



NAME
       systemd-cgls - Recursively show control group contents

SYNOPSIS
       systemd-cgls [OPTIONS...] [CGROUP...]

DESCRIPTION
       systemd-cgls recursively shows the contents of the selected Linux control group hierarchy in
            a
       tree. If arguments are specified, shows all member processes of the specified control groups
       plus all their subgroups and their members. The control groups may either be specified by
           their
       full file paths or are assumed in the systemd control group hierarchy. If no argument is
       specified and the current working directory is beneath the control group mount point
       /sys/fs/cgroup, shows the contents of the control group the working directory refers to.
       Otherwise, the full systemd control group hierarchy is shown.

       By default, empty control groups are not shown.

OPTIONS
       The following options are understood:

       --all
           Do not hide empty control groups in the output.

       -l, --full
           Do not ellipsize process tree members.

       -k
           Include kernel threads in output.

       -M MACHINE, --machine=MACHINE
           Limit control groups shown to the part corresponding to the container MACHINE.

       -h, --help
           Print a short help text and exit.

       --version
           Print a short version string and exit.

       --no-pager
           Do not pipe output into a pager.

EXIT STATUS
       On success, 0 is returned, a non-zero failure code otherwise.

SEE ALSO
       systemd(1), systemctl(1), systemd-cgtop(1), systemd-nspawn(1), ps(1)



systemd 215                                                                                SYSTEMD-CGLS
    (1)
```

Listing 42: man 1 systemd-cgtop

```
SYSTEMD-CGTOP(1)                               systemd-cgtop                               SYSTEMD-CGTOP
    (1)



NAME
       systemd-cgtop - Show top control groups by their resource usage
```

```
SYNOPSIS
       systemd-cgtop [OPTIONS...]

DESCRIPTION
       systemd-cgtop shows the top control groups of the local Linux control group hierarchy,
           ordered
       by their CPU, memory, or disk I/O load. The display is refreshed in regular intervals (by
       default every 1s), similar in style to top(1). If systemd-cgtop is not connected to a tty,
           only
       one iteration is performed and no columns headers are printed. This mode is suitable for
       scripting.

       Resource usage is only accounted for control groups in the relevant hierarchy, i.e. CPU
           usage
       is only accounted for control groups in the "cpuacct" hierarchy, memory usage only for those
            in
       "memory" and disk I/O usage for those in "blkio". If resource monitoring for these resources
            is
       required, it is recommended to add the CPUAccounting=1, MemoryAccounting=1 and
       BlockIOAccounting=1 settings in the unit files in question. See systemd.resource-control(5)
           for
       details.

       To emphasize this: unless "CPUAccounting=1", "MemoryAccounting=1" and "BlockIOAccounting=1"
           are
       enabled for the services in question, no resource accounting will be available for system
       services and the data shown by systemd-cgtop will be incomplete.

OPTIONS
       The following options are understood:

       -p
           Order by control group path name.

       -t
           Order by number of tasks in control group (i.e. threads and processes).

       -c
           Order by CPU load.

       -m
           Order by memory usage.

       -i
           Order by disk I/O load.

       -b, --batch
           Run in "batch" mode: do not accept input and run until the iteration limit set with
           --iterations is exhausted or until killed. This mode could be useful for sending output
           from systemd-cgtop to other programs or to a file.

       -n, --iterations=
           Perform only this many iterations.

       -d, --delay=
           Specify refresh delay in seconds (or if one of "ms", "us", "min" is specified as unit in
           this time unit).

       --depth=
           Maximum control group tree traversal depth. Specifies how deep systemd-cgtop shall
               traverse
           the control group hierarchies. If 0 is specified, only the root group is monitored. For
               1,
           only the first level of control groups is monitored, and so on. Defaults to 3.

       -h, --help
           Print a short help text and exit.

       --version
           Print a short version string and exit.

KEYS
       systemd-cgtop is an interactive tool and may be controlled via user input using the
```

```
            following
       keys:

       h
            Shows a short help text.

       SPACE
            Immediately refresh output.

       q
            Terminate the program.

       p, t, c, m, i
            Sort the control groups by path, number of tasks, CPU load, memory usage, or IO load,
            respectively.

       %
            Toggle between showing CPU time as time or percentage.

       +, -
            Increase or decrease refresh delay, respectively.

EXIT STATUS
       On success, 0 is returned, a non-zero failure code otherwise.

SEE ALSO
       systemd(1), systemctl(1), systemd-cgls(1), systemd.resource-control(5), top(1)



systemd 215                                                                    SYSTEMD-CGTOP
    (1)
```

22.5.5 'systemd' service manager

Listing 43: man 1 systemd

```
SYSTEMD(1)                                    systemd                                    SYSTEMD
    (1)



NAME
       systemd, init - systemd system and service manager

SYNOPSIS
       systemd [OPTIONS...]

       init [OPTIONS...] {COMMAND}

DESCRIPTION
       systemd is a system and service manager for Linux operating systems. When run as first
            process
       on boot (as PID 1), it acts as init system that brings up and maintains userspace services.

       For compatibility with SysV, if systemd is called as init and a PID that is not 1, it will
       execute telinit and pass all command line arguments unmodified. That means init and telinit
            are
       mostly equivalent when invoked from normal login sessions. See telinit(8) for more
            information.

       When run as system instance, systemd interprets the configuration file system.conf,
            otherwise
       user.conf. See systemd-system.conf(5) for more information.

OPTIONS
       The following options are understood:

       --test
            Determine startup sequence, dump it and exit. This is an option useful for debugging
                only.
```

```
       --dump-configuration-items
           Dump understood unit configuration items. This outputs a terse but complete list of
           configuration items understood in unit definition files.

       --unit=
           Set default unit to activate on startup. If not specified, defaults to default.target.

       --system, --user
           For --system, tell systemd to run a system instance, even if the process ID is not 1, i.
               e.
           systemd is not run as init process.  --user does the opposite, running a user instance
               even
           if the process ID is 1. Normally it should not be necessary to pass these options, as
           systemd automatically detects the mode it is started in. These options are hence of
               little
           use except for debugging. Note that it is not supported booting and maintaining a full
           system with systemd running in --system mode, but PID not 1. In practice, passing --
               system
           explicitly is only useful in conjunction with --test.

       --dump-core
           Dump core on crash. This switch has no effect when run as user instance.

       --crash-shell
           Run shell on crash. This switch has no effect when run as user instance.

       --confirm-spawn
           Ask for confirmation when spawning processes. This switch has no effect when run as user
           instance.

       --show-status=
           Show terse service status information while booting. This switch has no effect when run
               as
           user instance. Takes a boolean argument which may be omitted which is interpreted as
               true.

       --log-target=
           Set log target. Argument must be one of console, journal, syslog, kmsg, journal-or-kmsg,
           syslog-or-kmsg, null.

       --log-level=
           Set log level. As argument this accepts a numerical log level or the well-known syslog
               (3)
           symbolic names (lowercase): emerg, alert, crit, err, warning, notice, info, debug.

       --log-color=
           Highlight important log messages. Argument is a boolean value. If the argument is
               omitted,
           it defaults to true.

       --log-location=
           Include code location in log messages. This is mostly relevant for debugging purposes.
           Argument is a boolean value. If the argument is omitted it defaults to true.

       --default-standard-output=, --default-standard-error=
           Sets the default output or error output for all services and sockets, respectively. That
           is, controls the default for StandardOutput= and StandardError= (see systemd.exec(5) for
           details). Takes one of inherit, null, tty, journal, journal+console, syslog,
           syslog+console, kmsg, kmsg+console. If the argument is omitted --default-standard-output
               =
           defaults to journal and --default-standard-error= to inherit.

       -h, --help
           Print a short help text and exit.

       --version
           Print a short version string and exit.

CONCEPTS
       systemd provides a dependency system between various entities called "units" of 12 different
       types. Units encapsulate various objects that are relevant for system boot-up and
           maintenance.
```

```
The majority of units are configured in unit configuration files, whose syntax and basic set
     of
options is described in systemd.unit(5), however some are created automatically from other
configuration, dynamically from system state or programmatically at runtime. Units may be
"active" (meaning started, bound, plugged in, ..., depending on the unit type, see below),
    or
"inactive" (meaning stopped, unbound, unplugged, ...), as well as in the process of being
activated or deactivated, i.e. between the two states (these states are called "activating",
"deactivating"). A special "failed" state is available as well, which is very similar to
"inactive" and is entered when the service failed in some way (process returned error code
    on
exit, or crashed, or an operation timed out). If this state is entered, the cause will be
logged, for later reference. Note that the various unit types may have a number of
    additional
substates, which are mapped to the five generalized unit states described here.

The following unit types are available:

 1. Service units, which start and control daemons and the processes they consist of. For
    details see systemd.service(5).

 2. Socket units, which encapsulate local IPC or network sockets in the system, useful for
    socket-based activation. For details about socket units see systemd.socket(5), for
        details
    on socket-based activation and other forms of activation, see daemon(7).

 3. Target units are useful to group units, or provide well-known synchronization points
     during
    boot-up, see systemd.target(5).

 4. Device units expose kernel devices in systemd and may be used to implement device-based
    activation. For details see systemd.device(5).

 5. Mount units control mount points in the file system, for details see systemd.mount(5).

 6. Automount units provide automount capabilities, for on-demand mounting of file systems
     as
    well as parallelized boot-up. See systemd.automount(5).

 7. Snapshot units can be used to temporarily save the state of the set of systemd units,
     which
    later may be restored by activating the saved snapshot unit. For more information see
    systemd.snapshot(5).

 8. Timer units are useful for triggering activation of other units based on timers. You may
    find details in systemd.timer(5).

 9. Swap units are very similar to mount units and encapsulate memory swap partitions or
     files
    of the operating system. They are described in systemd.swap(5).

10. Path units may be used to activate other services when file system objects change or are
    modified. See systemd.path(5).

11. Slice units may be used to group units which manage system processes (such as service
    and
    scope units) in a hierarchical tree for resource management purposes. See systemd.slice
        (5).

12. Scope units are similar to service units, but manage foreign processes instead of
    starting
    them as well. See systemd.scope(5).

Units are named as their configuration files. Some units have special semantics. A detailed
list is available in systemd.special(7).

systemd knows various kinds of dependencies, including positive and negative requirement
dependencies (i.e.  Requires= and Conflicts=) as well as ordering dependencies (After= and
Before=). NB: ordering and requirement dependencies are orthogonal. If only a requirement
dependency exists between two units (e.g.  foo.service requires bar.service), but no
    ordering
dependency (e.g.  foo.service after bar.service) and both are requested to start, they will
    be
```

```
started in parallel. It is a common pattern that both requirement and ordering dependencies
    are
placed between two units. Also note that the majority of dependencies are implicitly created
and maintained by systemd. In most cases, it should be unnecessary to declare additional
dependencies manually, however it is possible to do this.

Application programs and units (via dependencies) may request state changes of units. In
systemd, these requests are encapsulated as 'jobs' and maintained in a job queue. Jobs may
succeed or can fail, their execution is ordered based on the ordering dependencies of the
    units
they have been scheduled for.

On boot systemd activates the target unit default.target whose job is to activate on-boot
services and other on-boot units by pulling them in via dependencies. Usually the unit name
    is
just an alias (symlink) for either graphical.target (for fully-featured boots into the UI)
    or
multi-user.target (for limited console-only boots for use in embedded or server environments
    ,
or similar; a subset of graphical.target). However, it is at the discretion of the
administrator to configure it as an alias to any other target unit. See systemd.special(7)
    for
details about these target units.

Processes systemd spawns are placed in individual Linux control groups named after the unit
which they belong to in the private systemd hierarchy. (see cgroups.txt[1] for more
    information
about control groups, or short "cgroups"). systemd uses this to effectively keep track of
processes. Control group information is maintained in the kernel, and is accessible via the
file system hierarchy (beneath /sys/fs/cgroup/systemd/), or in tools such as ps(1) (ps xawf
    -eo
pid,user,cgroup,args is particularly useful to list all processes and the systemd units they
belong to.).

systemd is compatible with the SysV init system to a large degree: SysV init scripts are
supported and simply read as an alternative (though limited) configuration file format. The
SysV /dev/initctl interface is provided, and compatibility implementations of the various
    SysV
client tools are available. In addition to that, various established Unix functionality such
     as
/etc/fstab or the utmp database are supported.

systemd has a minimal transaction system: if a unit is requested to start up or shut down it
will add it and all its dependencies to a temporary transaction. Then, it will verify if the
transaction is consistent (i.e. whether the ordering of all units is cycle-free). If it is
    not,
systemd will try to fix it up, and removes non-essential jobs from the transaction that
    might
remove the loop. Also, systemd tries to suppress non-essential jobs in the transaction that
would stop a running service. Finally it is checked whether the jobs of the transaction
contradict jobs that have already been queued, and optionally the transaction is aborted
    then.
If all worked out and the transaction is consistent and minimized in its impact it is merged
with all already outstanding jobs and added to the run queue. Effectively this means that
before executing a requested operation, systemd will verify that it makes sense, fixing it
    if
possible, and only failing if it really cannot work.

Systemd contains native implementations of various tasks that need to be executed as part of
the boot process. For example, it sets the hostname or configures the loopback network
    device.
It also sets up and mounts various API file systems, such as /sys or /proc.

For more information about the concepts and ideas behind systemd, please refer to the
    Original
Design Document[2].

Note that some but not all interfaces provided by systemd are covered by the Interface
Stability Promise[3].

Units may be generated dynamically at boot and system manager reload time, for example based
     on
```

```
       other configuration files or parameters passed on the kernel command line. For details see
           the
       Generators Specification[4].

       Systems which invoke systemd in a container or initrd environment should implement the
       Container Interface[5] or initrd Interface[6] specifications, respectively.

DIRECTORIES
       System unit directories
           The systemd system manager reads unit configuration from various directories. Packages
               that
           want to install unit files shall place them in the directory returned by pkg-config
               systemd
           --variable=systemdsystemunitdir. Other directories checked are
           /usr/local/lib/systemd/system and /lib/systemd/system. User configuration always takes
           precedence.  pkg-config systemd --variable=systemdsystemconfdir returns the path of the
           system configuration directory. Packages should alter the content of these directories
               only
           with the enable and disable commands of the systemctl(1) tool. Full list of directories
               is
           provided in systemd.unit(5).

       User unit directories
           Similar rules apply for the user unit directories. However, here the XDG Base Directory
           specification[7] is followed to find units. Applications should place their unit files
               in
           the directory returned by pkg-config systemd --variable=systemduserunitdir. Global
           configuration is done in the directory reported by pkg-config systemd
           --variable=systemduserconfdir. The enable and disable commands of the systemctl(1) tool
               can
           handle both global (i.e. for all users) and private (for one user) enabling/disabling of
           units. Full list of directories is provided in systemd.unit(5).

       SysV init scripts directory
           The location of the SysV init script directory varies between distributions. If systemd
           cannot find a native unit file for a requested service, it will look for a SysV init
               script
           of the same name (with the .service suffix removed).

       SysV runlevel link farm directory
           The location of the SysV runlevel link farm directory varies between distributions.
               systemd
           will take the link farm into account when figuring out whether a service shall be
               enabled.
           Note that a service unit with a native unit configuration file cannot be started by
           activating it in the SysV runlevel link farm.

SIGNALS
       SIGTERM
           Upon receiving this signal the systemd system manager serializes its state, reexecutes
           itself and deserializes the saved state again. This is mostly equivalent to systemctl
           daemon-reexec.

           systemd user managers will start the exit.target unit when this signal is received. This
                is
           mostly equivalent to systemctl --user start exit.target.

       SIGINT
           Upon receiving this signal the systemd system manager will start the ctrl-alt-del.target
           unit. This is mostly equivalent to systemctl start ctl-alt-del.target.

           systemd user managers treat this signal the same way as SIGTERM.

       SIGWINCH
           When this signal is received the systemd system manager will start the kbrequest.target
           unit. This is mostly equivalent to systemctl start kbrequest.target.

           This signal is ignored by systemd user managers.

       SIGPWR
           When this signal is received the systemd manager will start the sigpwr.target unit. This
                is
           mostly equivalent to systemctl start sigpwr.target.
```

```
SIGUSR1
    When this signal is received the systemd manager will try to reconnect to the D-Bus bus.

SIGUSR2
    When this signal is received the systemd manager will log its complete state in human
    readable form. The data logged is the same as printed by systemctl dump.

SIGHUP
    Reloads the complete daemon configuration. This is mostly equivalent to systemctl
    daemon-reload.

SIGRTMIN+0
    Enters default mode, starts the default.target unit. This is mostly equivalent to
        systemctl
    start default.target.

SIGRTMIN+1
    Enters rescue mode, starts the rescue.target unit. This is mostly equivalent to
        systemctl
    isolate rescue.target.

SIGRTMIN+2
    Enters emergency mode, starts the emergency.service unit. This is mostly equivalent to
    systemctl isolate emergency.service.

SIGRTMIN+3
    Halts the machine, starts the halt.target unit. This is mostly equivalent to systemctl
    start halt.target.

SIGRTMIN+4
    Powers off the machine, starts the poweroff.target unit. This is mostly equivalent to
    systemctl start poweroff.target.

SIGRTMIN+5
    Reboots the machine, starts the reboot.target unit. This is mostly equivalent to
        systemctl
    start reboot.target.

SIGRTMIN+6
    Reboots the machine via kexec, starts the kexec.target unit. This is mostly equivalent
        to
    systemctl start kexec.target.

SIGRTMIN+13
    Immediately halts the machine.

SIGRTMIN+14
    Immediately powers off the machine.

SIGRTMIN+15
    Immediately reboots the machine.

SIGRTMIN+16
    Immediately reboots the machine with kexec.

SIGRTMIN+20
    Enables display of status messages on the console, as controlled via systemd.show_status
        =1
    on the kernel command line.

SIGRTMIN+21
    Disables display of status messages on the console, as controlled via systemd.
        show_status=0
    on the kernel command line.

SIGRTMIN+22, SIGRTMIN+23
    Sets the log level to "debug" (or "info" on SIGRTMIN+23), as controlled via
    systemd.log_level=debug (or systemd.log_level=info on SIGRTMIN+23) on the kernel command
    line.

SIGRTMIN+24
    Immediately exits the manager (only available for --user instances).
```

```
       SIGRTMIN+26, SIGRTMIN+27, SIGRTMIN+28, SIGRTMIN+29
           Sets the log level to "journal-or-kmsg" (or "console" on SIGRTMIN+27, "kmsg" on
           SIGRTMIN+28, or "syslog-or-kmsg" on SIGRTMIN+29), as controlled via
           systemd.log_target=journal-or-kmsg (or systemd.log_target=console on SIGRTMIN+27,
           systemd.log_target=kmsg on SIGRTMIN+28, or systemd.log_target=syslog-or-kmsg on
           SIGRTMIN+29) on the kernel command line.

ENVIRONMENT
       $SYSTEMD_LOG_LEVEL
           systemd reads the log level from this environment variable. This can be overridden with
           --log-level=.

       $SYSTEMD_LOG_TARGET
           systemd reads the log target from this environment variable. This can be overridden with
           --log-target=.

       $SYSTEMD_LOG_COLOR
           Controls whether systemd highlights important log messages. This can be overridden with
           --log-color=.

       $SYSTEMD_LOG_LOCATION
           Controls whether systemd prints the code location along with log messages. This can be
           overridden with --log-location=.

       $XDG_CONFIG_HOME, $XDG_CONFIG_DIRS, $XDG_DATA_HOME, $XDG_DATA_DIRS
           The systemd user manager uses these variables in accordance to the XDG Base Directory
           specification[7] to find its configuration.

       $SYSTEMD_UNIT_PATH
           Controls where systemd looks for unit files.

       $SYSTEMD_SYSVINIT_PATH
           Controls where systemd looks for SysV init scripts.

       $SYSTEMD_SYSVRCND_PATH
           Controls where systemd looks for SysV init script runlevel link farms.

       $LISTEN_PID, $LISTEN_FDS
           Set by systemd for supervised processes during socket-based activation. See
           sd_listen_fds(3) for more information.

       $NOTIFY_SOCKET
           Set by systemd for supervised processes for status and start-up completion notification.
           See sd_notify(3) for more information.

KERNEL COMMAND LINE
       When run as system instance systemd parses a number of kernel command line arguments[8]:

       systemd.unit=, rd.systemd.unit=
           Overrides the unit to activate on boot. Defaults to default.target. This may be used to
           temporarily boot into a different boot unit, for example rescue.target or
           emergency.service. See systemd.special(7) for details about these units. The option
           prefixed with "rd."  is honored only in the initial RAM disk (initrd), while the one
               that
           is not prefixed only in the main system.

       systemd.dump_core=
           Takes a boolean argument. If true, systemd dumps core when it crashes. Otherwise, no
               core
           dump is created. Defaults to true.

       systemd.crash_shell=
           Takes a boolean argument. If true, systemd spawns a shell when it crashes. Otherwise, no
           shell is spawned. Defaults to false, for security reasons, as the shell is not protected
                by
           any password authentication.

       systemd.crash_chvt=
           Takes an integer argument. If positive systemd activates the specified virtual terminal
           when it crashes. Defaults to -1.

       systemd.confirm_spawn=
```

```
            Takes a boolean argument. If true, asks for confirmation when spawning processes.
                Defaults
            to false.

       systemd.show_status=
            Takes a boolean argument or the constant auto. If true, shows terse service status
                updates
            on the console during bootup.  auto behaves like false until a service fails or there is
                 a
            significant delay in boot. Defaults to true, unless quiet is passed as kernel command
                line
            option in which case it defaults to auto.

       systemd.log_target=, systemd.log_level=, systemd.log_color=, systemd.log_location=
            Controls log output, with the same effect as the $SYSTEMD_LOG_TARGET, $SYSTEMD_LOG_LEVEL
                ,
            $SYSTEMD_LOG_COLOR, $SYSTEMD_LOG_LOCATION environment variables described above.

       systemd.default_standard_output=, systemd.default_standard_error=
            Controls default standard output and error output for services, with the same effect as
                the
            --default-standard-output= and --default-standard-error= command line arguments
                described
            above, respectively.

       systemd.setenv=
            Takes a string argument in the form VARIABLE=VALUE. May be used to set default
                environment
            variables to add to forked child processes. May be used more than once to set multiple
            variables.

       quiet
            Turn off status output at boot, much like systemd.show_status=false would. Note that
                this
            option is also read by the kernel itself and disables kernel log output. Passing this
            option hence turns off the usual output from both the system manager and the kernel.

       debug
            Turn on debugging output. This is equivalent to systemd.log_level=debug. Note that this
            option is also read by the kernel itself and enables kernel debug output. Passing this
            option hence turns on the debug output from both the system manager and the kernel.

       -b, emergency
            Boot into emergency mode. This is equivalent to systemd.unit=emergency.target and
                provided
            for compatibility reasons and to be easier to type.

       single, s, S, 1
            Boot into rescue mode. This is equivalent to systemd.unit=rescue.target and provided for
            compatibility reasons and to be easier to type.

       2, 3, 4, 5
            Boot into the specified legacy SysV runlevel. These are equivalent to
            systemd.unit=runlevel2.target, systemd.unit=runlevel3.target,
            systemd.unit=runlevel4.target, and systemd.unit=runlevel5.target, respectively, and
            provided for compatibility reasons and to be easier to type.

       locale.LANG=, locale.LANGUAGE=, locale.LC_CTYPE=, locale.LC_NUMERIC=, locale.LC_TIME=,
       locale.LC_COLLATE=, locale.LC_MONETARY=, locale.LC_MESSAGES=, locale.LC_PAPER=,
       locale.LC_NAME=, locale.LC_ADDRESS=, locale.LC_TELEPHONE=, locale.LC_MEASUREMENT=,
       locale.LC_IDENTIFICATION=
            Set the system locale to use. This overrides the settings in /etc/locale.conf. For more
            information see locale.conf(5) and locale(7).

       For other kernel command line parameters understood by components of the core OS, please
            refer
       to kernel-command-line(7).

SOCKETS AND FIFOS
       /run/systemd/notify
            Daemon status notification socket. This is an AF_UNIX datagram socket and is used to
            implement the daemon notification logic as implemented by sd_notify(3).
```

```
       /run/systemd/shutdownd
           Used internally by the shutdown(8) tool to implement delayed shutdowns. This is an
               AF_UNIX
           datagram socket.

       /run/systemd/private
           Used internally as communication channel between systemctl(1) and the systemd process.
               This
           is an AF_UNIX stream socket. This interface is private to systemd and should not be used
                in
           external projects.

       /dev/initctl
           Limited compatibility support for the SysV client interface, as implemented by the
           systemd-initctl.service unit. This is a named pipe in the file system. This interface is
           obsolete and should not be used in new applications.

SEE ALSO
       The systemd Homepage[9], systemd-system.conf(5), locale.conf(5), systemctl(1), journalctl(1)
            ,
       systemd-notify(1), daemon(7), sd-daemon(3), systemd.unit(5), systemd.special(5), pkg-config
            (1),
       kernel-command-line(7), bootup(7), systemd.directives(7)

NOTES
        1. cgroups.txt
           https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt

        2. Original Design Document
           http://0pointer.de/blog/projects/systemd.html

        3. Interface Stability Promise
           http://www.freedesktop.org/wiki/Software/systemd/InterfaceStabilityPromise

        4. Generators Specification
           http://www.freedesktop.org/wiki/Software/systemd/Generators

        5. Container Interface
           http://www.freedesktop.org/wiki/Software/systemd/ContainerInterface

        6. initrd Interface
           http://www.freedesktop.org/wiki/Software/systemd/InitrdInterface

        7. XDG Base Directory specification
           http://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html

        8. If run inside a Linux container these arguments may be passed as command line arguments
            to
           systemd itself, next to any of the command line options listed in the Options section
           above. If run outside of Linux containers, these arguments are parsed from /proc/cmdline
           instead.

        9. systemd Homepage
           http://www.freedesktop.org/wiki/Software/systemd/



systemd 215                                                                              SYSTEMD
    (1)
```

22.5.6 Tunnels mit 'ip tun'

```
ip-tunnel/ipip:
  @tr1: sudo ip tunnel add xyz mode ipip ttl 64 remote 192.168.2.73 local 192.168.1.99
       dev eth0
        sudo ip link set xyz up
        sudo ip ro add 192.168.2.73/32 dev xyz
  @rPi3:sudo ip tun add xyz mode ipip ttl 64 remote 192.168.1.99 local 192.168.2.73
      dev wlan0
        sudo ip link set xyz up
        sudo ip ro add 192.168.1.99/32 dev xyz
```

```
ssh-port-forwarding:
  dings@rPi2:$ "ssh -N -L 10.10.63.72:2223:rPi1:2225 dings@rPi1"
  dings@rPi1:$ "sudo nano /etc/ssh/sshd_config"  ("Port 2225" und "GatewayPorts yes"
      dazu)
  dings@rPi1:$ "sudo /etc/init.d/ssh reload"
  cfs@tr1:$    "ssh dings@rPi2lan -p 2223"

tr1: ssh dings@rPi1 via rPi2lan:
cfs@tr1:$ ip ro
 default via 10.10.77.92 dev eth0
 10.10.0.0/16 dev eth0  proto kernel  scope link  src 10.10.8.37
 192.168.1.0/24 dev eth0  scope link  src 192.168.1.99
 192.168.2.0/24 via 10.10.63.72 dev eth0

dings@rPi2lan:~ $ ip ro
 default via 10.10.8.37 dev eth0
 169.254.0.0/16 dev eth0  proto kernel  scope link  src 169.254.15.224  metric 202
 192.168.2.0/24 dev wlan0  proto kernel  scope link  src 192.168.2.72  metric 303
dings@rPi1:~ $ ip ro
 default via 192.168.2.72 dev wlan0
 192.168.2.0/24 dev wlan0  proto kernel  scope link  src 192.168.2.71  metric 303
dings@rPi2lan:~ $
```

22.5.7 german umlauts auf US-Keyboard

schreib Folgendes ins ~/.Xmodmap:
(genau so,
des ischka 'Datenmull',
huni vor 100000 Jahr ausan Forum)

```
removeLock=Caps_Lock
keycode66=Mode_switch
keycode26=eEEuroSigncent
keycode30=uUudiaeresisUdiaeresis
keycode32=oOodiaeresisOdiaeresis
keycode38=aAadiaeresisAdiaeresis
keycode39=sSssharp
```

dann: `xmodmap  ~/.Xmodmap`

nun sind CAPSLOCK+'a' ==> ä
usw.

22.5.8 Commands, Brainstmg, ungeordnet

To be done: Commands, Utilities

```
"ls -sortg", "ls -soru", "ls -sou"
man create_database
man 3p fflush
pdfjam
patch
texdoc <search/name>
ExitStatus: "echo $?"
adduser
addgroup
usermod
chown
chmod
chroot
env     - run program in modified environment
chattr  - lsattr
chcon   - change file security context
chfn    - change real user name and information (typically printed by finger(1))
chgrp
chpasswd - update passwords in batch mode
newusers - update and create new users in batch
/etc/nologin
chrt    - manipulate the real-time attributes of a process
chvt    - change foreground virtual terminal
xargs
seq
tac             cat reverse
tr              translates characters while copying stdin to stdout
dpipe
```

```
vmstat
mincore - determine whether pages are resident in memory:
          int mincore(void *addr, size_t length, unsigned char *vec);
mmap, munmap - map or unmap files or devices into memory

hwinfo
hwclock
hardinfo
/usr/bin/lshal
pvs - report information about physical volumes
/bin/lsblk
e2label
e2freefrag(8)
findfs - find a filesystem by label or UUID
fintmnt - find a file system
/proc/self/fdinfo
man 2 syscalls - Linux OS system calls
man 7 hier - description of the file system hierarchy
fstype
enscript(1)
libusb.org
usbmon  install nm
readelf
atop
htop
iftop
iotop  - 'sudo iotop -o -a'
ntop
vtop
ionice
wipe
kpartx
partx
losetup
partprobe
/dev/disk/by-uuid by-id by-label ...
cfdisk - manipulate  disk partition table
fdisk - manipulate  disk partition table
parted - manipulate  disk partition table
sfdisk -manipulate  disk partition table
gdisk - GUI GPT manipulator
sgdisk - CLI gdisk
lsblk-findmnt-findfs-blkid-'cat /proc/partitions'
blkid - print block device attributes
lsinitramfs     list content of an initramfs image

xrandr -q -display :0;
xrandr --current -display :0
xrandr -display :0.0 -fb 640x480;
xrandr --output VGA1 --auto
xrandr --output VGA1 --off
xrandr --output VGA1 --transform 1,0,0,0,1,0,0,0,1
xrandr --output VGA1 --transform 1,0,100,0,1,0,0,0,1
X, zB. 'X -query myHost ...'
Xserver
man Xorg
xhost
xrandr
pilot, jpilot
xvidtune
Xdmx
Xvfb
xset
vncviewer BB2:1
vinagre
loadkeys us
setxkbmap us
cvt - calculate VESA CVT mode lines
xvidtune
avahi - ZeroConf-Daemon

"sshfs dings@bb2:/mnt/skb2 /mnt/bbStix"
"ssh -L 1234:localhost:8888 w12acfs@BB2 c/rtos/NWES3/jit4 8888"
```

```
netstat
nmap -n -F --reason -vv --packet-trace  192.168.100.2
nmcli - NetworkManager command line UI
nmtui - NetworkManager 'curses' based text UI
ethtool
iptables -t nat -n -L
stunnel
ip-tunnel/ipip:
  @tr1: sudo ip tunnel add xyz mode ipip ttl 64 remote 192.168.2.73 local 192.168.1.99
       dev eth0
        sudo ip link set xyz up
        sudo ip ro add 192.168.2.73/32 dev xyz
  @rPi3:sudo ip tun add xyz mode ipip ttl 64 remote 192.168.1.99 local 192.168.2.73
      dev wlan0
        sudo ip link set xyz up
        sudo ip ro add 192.168.1.99/32 dev xyz
ssh-port-forwarding:
  dings@rPi2:$ "ssh -N -L 10.10.63.72:2223:rPi1:2225 dings@rPi1"
  dings@rPi1:$ "sudo nano /etc/ssh/sshd_config"  ("Port 2225" und "GatewayPorts yes"
      dazu)
  dings@rPi1:$ "sudo /etc/init.d/ssh reload"
  cfs@tr1:$    "ssh dings@rPi2lan -p 2223"
netcat 4 Arduino/UsbFwd
  clonePC= src: 'netcat pc2 1234 < /dev/sda'    dest: 'netcat -l 1234 | dd of=/dev/sda
      '
socat - serial traffic connecting + sniffing multipurpose relay (SOcket CAT)
kismet - Wireless sniffing and monitoring
www.lntwww.de - Ein Lerntutorial f. die Nachrichtentechnik; Digitalsignaluebertragung

LaTeX:
 rotatebox{45}{text}
 sidewaystable{}
 \stopbreaks...\startbreaks
 \enlargehispage*{hoehe}


noweb  writes the program  source code to the output files and writes a TeX file for
    documentation.
wsdl - Mono's Web Service Proxy Generator
distcc          distributes  compilation of C code across several machines on a
    network
qemu    QEMU PC System emulator, qemu-doc - QEMU Emulator User Documentation
        qemu-system-i386, qemu-system-x86_64, qemu-system-arm, qemu-system-mips, qemu-
            system-cris,
        qemu-system-s390x, qemu-system-m68k, ...
vdeq    vdeq - Virtual Distributed Ethernet wrapper for QEMU/KVM virtual machines
netdiscover...ARP-NetzwerkScan
$
```

22.6 U-Mitschriften

Listing 44: U-Mitschrift1819

```
Der Linuxrechner "obelix" steht im A205
im schwarzen Schrank und hat
  IP-Adresse  10.10.63.61



Grundlagen:
  In UNIX+Linux gibt es keine Laufwerksbuchstaben
  wie A:, C:, ...
  sondern nur das 'root'-Dateisystem '/'
  /
  /etc
  /proc
  /home
  /home/franz
  /home/josef
  usw.
```

```
  Laufwerke tauchen als Directories in /media oder /mnt
  auf.
  So sind zB unsere user-Account-Directories
  in /home/pupil/n...   (wie zB /home/pupil/n17aprela)
  in Wahrheit eine separate Partition.
  (dadurch kann ein versehentliches Vollfuellen der Partition
  das System nicht blockieren)

  Auf meinemXH Notebook hier ist /media/sdb3 ein extra Laufwerk:
ls -l /media/
drwxr-xr-x 2 knoppix knoppix    0 Oct 23 08:21 sda1
drwxr-xr-x 2 knoppix knoppix    0 Oct 23 08:21 sda2
drwxr-xr-x 2 knoppix knoppix    0 Oct 23 08:21 sda3
drwxr-xr-x 2 knoppix knoppix    0 Oct 23 08:21 sda4
drwxr-xr-x 2 knoppix knoppix    0 Oct 23 08:21 sdb1
drwxr-xr-x 2 knoppix knoppix    0 Oct 23 08:21 sdb2
drwxr-xr-x 4 root    root    4096 Nov  5  2015 sdb3
  (es ist die Partition "3" auf der SDcard)

  Das Kommando 'lsblk' zeigt Speicherdevices:
lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0   7.5G  0 disk
|-sda1   8:1    0   3.3G  0 part
|-sda2   8:2    0   4.2G  0 part
|-sda3   8:3    0   7.9M  0 part
'-sda4   8:4    0   7.9M  0 part
sdb      8:16   1    30G  0 disk
|-sdb1   8:17   1    14G  0 part /mnt-system
|-sdb2   8:18   1   498M  0 part /KNOPPIX-DATA
'-sdb3   8:19   1  15.6G  0 part /media/sdb3
zram0  251:0    0   747M  0 disk [SWAP]
cloop0 240:0    0   9.6G  1 disk /KNOPPIX

  Das Kommando 'lsusb' zeigt USB-Devices:
lsusb
Bus 001 Device 002: ID 093a:2700 Pixart Imaging, Inc.
Bus 001 Device 003: ID 058f:6335 Alcor Micro Corp. SD/MMC Card Reader
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub




fuer Admins:
Linux-Ping 'abdrehen':
  echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
bzw.
  echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_all

das /proc-Dateisystem (ein 'pseudo'-Filesystem):
  /proc/cpuinfo
  /proc/meminfo
  /proc/partitions
  u.v.a.m.

  (Inhalt anzeigen mit 'cat' wie zB.
  'cat /proc/partitions'  )

iXH erzeuge jetzt user-Accounts fuer neue Teilnehmer:
  es gibt Kommandos 'adduser', addgroup usw.
  aber den usern sollen weitere Directories
  plus eine Webseite automatisch angelegt werden,
  deshalb lasse ich das ein 'shell-script' machen.
  Ein 'shell-script' ist eine Text-Datei voller Linux-Kommandos.
Shell-Scripte ('Shell-Scripting') 111218
----------------------------------------

Shellscripte sind Dateien mit
```

```
Linux-Kommandos drin
--Du brauchst das "Ausfuehrungsrecht" (X) fuer
  diese Datei (Du, nicht die Datei, Du)
--die erste Zeile des Shellscripts sollte
  "#!/bin/bash" lauten
--die Kommandos in Deinem Shellscript sollte mit
  dem vollstaendigen Pfadnamen angefuehrt werden
  (sonst ist die Funktion von der '$PATH'-Sucheinstellung
  abhaengig)
--'bash' ist der Kommando-Interpreter
  (der echt vieles kann)
--'bash' ist in der man-page 'man bash' (man 1 bash) beschrieben.
  (umfangreich!)
--man tauft ueblicherweise (muss aber nicht) '.sh'

zB:
  iXH schreibe mir ein Shellscript, das mir die Datei IT*pdf
  auf den angeschlossenen UsbStick kopiert.
  Vorher untersuche iXH aber den (fremden) Stick:
  lsblk
  fdisk -l
  parted -l

  #!/bin/bash
  echo "ich bin XHs It*pdf-kopier-Script"


knoppix@Microknoppix:~$ ./ITkopier.sh
bash: ./ITkopier.sh: Permission denied
knoppix@Microknoppix:~$
knoppix@Microknoppix:~$
knoppix@Microknoppix:~$
knoppix@Microknoppix:~$ ==> AETSCH! kein Ausfuehrungsrecht!
Linux 04Dez18
=================

Zwischenfrage:
DNS-Eintraege sind in
  /etc/resolv.conf
Man soll sie aber nicht (mehr) direkt dort eintragen,
sondern in /etc/network/interfaces (bei Verwendung
des 'NetworkManager') bzw. /etc/dhcpcd.conf
(


Der 'Boot'-Vorgang (richtig 'bootstrap loader',
altmodisch eingedeutscht auch 'Urlader'):

In 'FAT'-Partitiontables ist der vorderste
Sector (MBR master boot record, 512 Byte):
  446 Byte Boot Code
   64 Byte 4 Stk Partition tables (1..4)
    2 Byte 'magic number'

Das BIOS laedt diese 446 Byte Bootcode ins RAM
und fuehrt es aus. Typischerweise ladet diese
einen Boot-Manager nach. Die gaengigen Bootmanager
sind teilweise durchaus bekannt (je nach OS
unterscheidet sich der weitere Ablauf)
Linuxe laden dann eine Datei 'initrd'
(siehe '/boot'-Directory) ins RAM,
nachdem ein Teil des RAM wie eine 'Disk' formatiert
wurde (Dateisystem 'initramfs').
Diese 'initrd' (wie 'initial ramdisk') enthaelt
typischerweise Dateisystemtreiber, die dem Kernel
zum Systemstart fehlen (die 'root'-Partition);
falls der Kernel sein 'root'-file system
beim Boot nicht lesen kann, meldet er Dir etwas
von 'kernel panic' oder 'waiting for root file
system to appear...').

Die Kerneldatei nennt sich 'vmlinuz'
(zurzeit sind fuer WindowsPCs meist
```

```
mehrere vmlinuz-Varianten fuer 32-bit
und 64-bit Rechner vorhanden)

Beim Bootmanager 'grub2' werden Boot-Varianten
in der Konfigurationsdatei
  /boot/grub/menu.lst
konfiguriert.

Verbreitet, besonders auf nicht-PC-Systemen,
sind aber auch der 'syslinux' und der
'isolinux' Bootmanager (zB auf CD/DVD, Stick, SD).

Nach dem Laden/Starten von 'initrd' und 'vmlinuz'
wird das Programm '/bin/init' (im Kernelcode
oder in der 'initrd') ausgefuehrt (wird dann
Prozess Nr.1 - siehe 'ps'-Kommando und lauft
bis System-Shutdown).
/bin/init liest (gewoehnlich) /etc/inittab
als Konfigurationsdatei.

Fuer den weiteren System-Startup haben sich
konkurrierende Mechanismen etabliert,
besonders der sog.
'SystemV' (klassisch) und der
'systemd' Mechanismus (juenger).

Zum Grossteil ist der Startup mittels
SHELL-SCRIPTs ('Kommando-Dateien')
und damit in lesbaren ASCII-Textfiles
programmiert,
die man ohne sonderliche Werkzeuge
(wie 'Registry-Editoren') lesen/modifizieren
kann.

Das SystemV kennt 'runlevels':
  runlevel 1:  single user
  runlevel 2:  multi user
  runlevel 3:  +Netzwerk
  runlevel 5:  +GUI
  runlevel 6:  reboot
  runlevel 0:  HALT
Den Runlevel wechelt man mit 'init <runlevel>'
zB. 'init 6' (fuer reboot)Linux Mi21Feb'18
==================

Das Utility "file":
  tests each argument in an attempt to classify it.

  'file' sagt mir,
  was die angegebene Datei fuer ein Dateiformat
  enthaelt (.txt, .png, .pdf, .odt usw.)
  zB.:
  $ file /home
  ==> "/home: directory"

  $ file /etc/hosts
  ==> /etc/hosts: ASCII text

  $ file /usr/bin/python
  ==> "/usr/bin/python: symbolic link to `python2.7'"

  $ file /usr/bin/python3.3
  "/usr/bin/python3.3: ELF 32-bit LSB  executable,
  Intel 80386, version 1 (SYSV), dynamically linked
  (uses shared libs), for GNU/Linux 2.6.32"

  usw.

  'file' achtet NICHT auf die Endigung (.pdf, ...),
  sondern den Inhalt der Datei,
  durch Umtaufen der Datei kann man 'file'
  nicht in die Irre fuehren:
```

```
cp datei.png datei.pdf
cp datei.png datei.exe
$ file datei.png
==> datei.png: PNG image data, 950 x 348
$ mv datei.png datei.pdf
$ file datei.pdf
==> datei.pdf: PNG image data, 950 x 348
mv datei.pdf datei.exe
$  file datei.exe
==> datei.exe: PNG image data, 950 x 348

U-Mitschrift Linux 16Okt18:
---------------------------

Lehrerkuerzel "XH" = Christoph (SCHOENHERR)

1) XHs Linux-Rechner im Schulnetz:
   -"obelix" 10.10.63.61 (da kriegst einen UserAccount)
        (lies Details dazu in meinem "Fst-PDF)
        (bitte "anbrausen": http://10.10.63.61)
   -"bb" 10.10.63.245

2) Beamer (odgl.) ansprechen:
   a) "xrandr -q"
      zeigt die Namen der Video-Ausgaenge
      zB. "LVDS1", "VGA1" odgl.
   b) "xrandr --output <Ausgang> --auto
      (versucht auto-probing zum Video-Ausgang)

3) in Linux gibt es keine Laufwerksbuchstaben (C:, D:,...)
   (die Devices tauchen alle als Unterverzeichnisse,
   meistens in "/media" auf:
        /media/sda2, /media/sdb1, /media/sdc1, ...

4) Linux verwendet "/" statt "\" in Dateipfaden:
   /home/franz/public_html/index.html
   /home/josef/c/hallo.c
   ...

5) in Linux gibt es KEIN "WIEDERHERSTELLEN" !!!
   Loesche daher nur in Ruhe und im Zweifelsfalle
   gar nicht.

6) Mach Backups!
   (auf externe Platten undso).
   UsbSticks sind UNVERLAESSLICH
   SD-Cards sind UNVERLAESSLICH
   (deren sog. "NAND-Flash-Zellen" leiern aus!)

7) MeinXH liebstes Backup- und Transfer-Programm
   ist "rsync" (ein Konsol-Kommando mit vielen
   Optionen, zB
   "rsync -auv <Quelle> <Ziel>"









"Profiling tool": time
  time <command>
  real    0m10.006s
  user    0m0.003s
  sys     0m0.000s


Datei-Komprimieren mit "tar":
Komprimieren: "tar -czvf <archiv> <files>
  zB:  tar -czvf myDocs.tgz  *.pdf
```

```
      ==> myDocs.tgz

"Entpacken: tar -xzvf <archiv>
           ==> alle Files einzeln


netcat (nc):
  a)  Netzwerk --> Monitor
  b)  Netzwerk <-- Keyboard

 zB  cat file  |  nc  10.10.63.61  1234

                sendet den Inhalt von 'file'
                an den Rechner 10.10.63.61 Port 1234

 zB  nc -l 1234   >  rxfile
.               wartet auf Netzwerk-Zusendung und
.               leitet diese auf die Datei 'rxfile' um
.               ("-l" ... "listen")

 oder zB:
 a)  nc -l 1234  |  dd of=/dev/sda4
 b)  dd if=/dev/sdc2 |  nc  zielrechner 1234

.                       b) kopiert die Partition 2 der
.                       Disk'c' aufs Netzwerk
.                       a) wartet ('-l') auf Netzwerkdaten
.                       und schreibt diese auf diese
.                       auf die Partition 4 der Disk'a'

Das Programm "dd":
  Low-Level copy
  Options: "if=..."  Input File
           "of=..."  Output File
  zB:
  dd if=RasPiImage.iso of=/dev/sdb
.               RasPi-Image auf Stick kopieren
20.Nov.18 'sshf' - SSH file system (Directory 'mappen')

1) ich brauche das 'sshfs' Programm (ggf. installieren)
   (workstation seitig)

2) ich brauche einen 'mount point'
   (leeres Directory mit Zugriffsrecht)

   "mkdir mountpoint"

3) das mount-Kommando

   "sshfs   n15alooser@10.10.63.61:   mountpoit"

4) verwenden ... (kopieren, hochladen, editieren...)


5) wieder aufheben/trennen:

   "fusermount -u  <mountpoint>


SMB/CIFS mount des "Luke"-Laufwerks
(das ist ja mittlerweile eine 'samba-share')

1) ich brauche einen 'mount point'
   (leeres Directory mit Zugriffsrecht)

   "mkdir mountpoint"

2) das mount-Kommando

   "mount.cifs //C01/DSK  mountpoint -o user=<Htl-User>"

   (zB. MikTeX-Variante unter
   "LUKE_New/Software/Dienstprogramme/TextEditor/
```

```
    LaTeX/")


zu: Linux Hilfe-System, die 'man' pages

   die man-pages sind unter
   /usr/share/man
   gespeichert:
   /usr/share/man/man1/*
   /usr/share/man/man2/*
   /usr/share/man/man3/*
   /usr/share/man/man4/*
   ...
   die manpage-Files dort drin sind ge'zip't
   (zB. 'printf.3.gz')

   bitte lies die Kapitelstruktur in
   'man man'
   (LinuxCommands in man1, OS-API-calls in man2,
   C-programmierfunktionen in man3,...)


-----------------
Linux 18Dez18
-----------------

utility 'convert':
------------------
  convert between image formats
  as well as resize an image
zB.
     'convert buidl.jpg buidl.png'
  erzeugt (zusaetzlich) die Datei 'buidl.png'
  im '.png' format

in einem shell-script zB alle 'gif' in 'jpg' konvertieren:
  #!/bin/bash
  cd $1
  for i in *.gif ; do
    convert $i $(echo $i | sed s/gif/jpg/)
  done
  exit

Geht alle *.gif-Dateien durch und konvertiert sie
  zu '.jpg'
Erklaerungen:
Wenn Du dieses Script zB unter 'gifjpg' speicherst,
dann musst Du Dir noch die 'x'-Rechte (eXecute)
auf diese Datei geben:
  chmod o+x gifjpg

  '$1' ... ist der erste KommandozeilenPARAMETER

  'for i in *.gif' ... i ist ein Element der
         Liste der Dateien mit '.gif' -Ende

  'convert' ... s.o.

  '$( ... )' ... fuehrt den ()-Inhalt aus und liefert
               davon die Ergebnisse

  'sed s/a/b/' ... ersetzt in der input-Zeile alle 'a'
         durch 'b'


utility 'dd':
-------------
Kopieren von Datentraeger-'Images' mit 'dd':
'dd' ist ein stinksimples, uraltes Utility;
     es kopiert Byte fuer Byte von der mit
     'if=' angegebenen Quelle in das mit
     'of=' angegebene Ziel,
zB.
```

```
    'dd if=quellFile of=zielFile'
if= und of= koennen aber auch DEVICES sein
(dann werden die Dateisysteme umgangen!)
zB.
   'dd if=/dev/sda1  of=myPartitionImage1'
speichert die gesamte 'sda1'-Partition
in die Datei 'myPartitionImage1'.
Damit koennte man die Parition etwa auf eine andere
Platte (oder einen Stick, Sdcard usw.) kopieren:
  'dd if=myPartitionImage1  of=/dev/sdb1'
ABER PASS AUF!
DAS ZERSTOERT NATUERLICH DIE /dev/sdb1 INHALTE!!!



Linux 28Feb18, Netzwerkeinstellungen

Files:
  /etc/hosts
  /etc/resolv.conf
  /etc/network/interfaces   (f.'NetworkManager')
    auto eth0
    iface eth0 static
      address   10.0.0.99
      netmask   255.255.0.0
      gateway   10.0.0.38
  (etc/wpa_supplicant/wpa_supplicant.conf)

Commands:
  ip
  ifconfig
  route

statische Namensaufloesung:
  /etc/hosts



mehrere IP-Adressen auf einem LAN-Device:
1) 'sudo ip a a 10.0.0.99/24 dev eth0'
  oder (alt)
2) 'sudo ifconfig eth0:1 10.0.0.39 netmask 255.255.255.0'



WindowsNetzlaufwerk = "CIFS"
mount -t TYPE DEVICE MOUNTPOINT
mount.cifs
```

22.7 IP Socket Programming

22.7.1 Linux-Übung mit RaspberryPi

Aufgabe:

Verbinde Deinen PC mit Deinem RaspberryPi (RasPi) ('Angry IP Scanner', 'Wireshark', APIPA/Zeroconf, Bonjour, 169.254.1.115 ?, 169.254.15.224 ?) Starte einen primitiv-Webserver mit dem Utility 'netcat'/'ncat'/'nc' s. →supersimpelServerMit'netcat'. Bringe das C-Demo 'Richards kleiner Webserver' (s.u.) auf Dein RasPi, compiliere es dort mit dem 'gcc' (s.Kommentar) und starte es. Kontaktiere es per Webbrowser von Deinem PC. Erweitere Deinen Webserver um die Einblendung der Sensor-Werte aus den device-Files /sys/class/rtc/rtc0/time, /sys/class/rtc/rtc0/date und /sys/class/net/eth0/address

22.7.2 Richards kleiner Webserver

Listing 45: Richards kleiner Webserver

```
/* File Richards 'wpeep.c' , cfs@GcF,PbQ,XcB
 * shows message from Web-Browser
 * +sends very simple reply
 * compile: gcc rwpeepXcBsrc.c -o rwpeep
 * run    : ./rwpeep <portNr>  (zB "./rwpeep 50100")
 * stop   : Strg+C   oder "serverstop" im URLstring
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#ifndef SO_REUSEPORT
 #define SO_REUSEPORT 15
#endif
#define IpPort  (argv[1])
int inhaltAnzeigenSieheUnten(char*);

    /*MAIN:*/
int main(int argc,char **argv){
  int           fdPORT, fdCOMM, lenC, r1=0;
  char          buf[4096], buf2[4096], *s, hostName[256];
  struct sockaddr_in    s_address;
  struct sockaddr       c_address;

  if(argc<2){ printf("usage: %s <PortNumber>",*argv); exit(0); }
  gethostname(hostName, 255);
  fdPORT = socket(AF_INET,SOCK_STREAM,0);
  s_address.sin_family      = AF_INET;
  s_address.sin_addr.s_addr = htonl(INADDR_ANY);        //ex:inet_addr("127.0.0.1");
  s_address.sin_port        = htons(atoi(IpPort));
  int one=1; setsockopt(fdPORT, SOL_SOCKET, SO_REUSEPORT, &one, sizeof one);
  if(bind(fdPORT,(struct sockaddr *) &s_address, sizeof(s_address))){
     perror("bind failed"); exit(EXIT_FAILURE);
  }
  listen(fdPORT,5);
  for(;;){
    printf("server: %s@%s waiting at port:%d\n", argv[0], hostName, atoi(IpPort));
    lenC    = sizeof(c_address);
    fdCOMM  = accept(fdPORT, &c_address, &lenC);
    if(1 > (r1=read(fdCOMM, buf, sizeof(buf)-2 ))) break; /*nix zu lesen*/
    buf[r1]=0; inhaltAnzeigenSieheUnten(buf);
    snprintf(buf2, sizeof(buf2)-1,
        "HTTP/1.0 200 OK\r\nServer: RWS\r\nContent-Type: text/html\r\n\r\n"
        "<html><body bgcolor=tan><h1>RWS - kleiner Webserver</h1>\n"
        "<br>Nur zum Testen, ein kleiner text...\n"
        "<hr><small>(c)2005 richard weinberger</small></body></html>\r\n\r\n"
        );
    write(fdCOMM, buf2, strlen(buf2));
```

```
    close(fdCOMM);
    if(strstr(buf,"serverstop")) break; /*"geheim"-shutdown URLstring*/
  }
  shutdown(fdPORT,SHUT_RDWR);
  close(fdPORT);
  printf("\nserver \"%s\" terminated.\n",*argv);
}/*end of main*/


int inhaltAnzeigenSieheUnten(char *s){
  printf("\n*--------new-message:------------*\n");
  for(;*s;s++){
    if(isprint(*s)) putchar(*s);
    else printf("(0x%02X)",*s);
    if('\n'==*s) putchar('\n');
  }
  printf("\n*--------message-end-------------*\n");
}
```

22.7.3 simpler Browser-Fake

Listing 46: Browser-Fake

```
/* File wclnt2.c =BrowserFake/* Socket-Programmierung, cfs@GcF,Sf9 */

#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <netdb.h>

int main(int argc,char **argv){
  int                   fd1, result, l;
  char                  s[16384], hname[256];
  struct sockaddr_in    address;

  address.sin_family=AF_INET;
  address.sin_addr.s_addr = inet_addr("10.10.10.30");
  address.sin_port = htons(80);  /*(NetworkByteOrder)*/
  printf("client \"%s\" @%s\n",*argv,(gethostname(hname,255),hname));
  fd1=socket( AF_INET, SOCK_STREAM, 0); /* == socket erzeugen und verbinden **/
  if((result=connect(fd1,(struct sockaddr *) &address,sizeof(address))) < 0){
      perror("cant connect\n");
  }else{
      sprintf(s,"GET /SMUE/srcvw.php?fn=weliza/weliza1.c&nr=123 HTTP1.0/\n"
                "Host: localhost:80\n"  "\r\n\r\n" );
      write(fd1,s,l=strlen(s));
      while( read(fd1, s, 16383) > 0 ){
        printf("Reply from server: /%s/\n",s);
        usleep(100000L);
      }
      close(fd1);
  }
}
```

22.7.4 Richards kleiner Webserver mit fork()

Listing 47: Richards kleiner Webserver mit fork()

```
/* File "websrv1.c"
 *
 * minimal-webserver, reentrant (fork())
 * by JESUS.RW 15-Mar-2005
 * cfs 28-Mar-2005
 ********************************************************************************
 */

#include <stdio.h>
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>

/* ****************************************function prototypes: ***/
void send_to_client(const char *, int );
void read_http_request(int );
void child_exit_handler();

/* ****************************************config****************/
#define SERVERNAME      "rws"
#define SERVERVERSION   "0.01"
#define MAXQUEUELEN     5
#define BUFFERLEN       4096
#define LISTENPORT      80

/* ************************************************************************
 * *** M A I N ***
 ************************************************************************/
int main(){
 int    server_socket;
 int    new_client;
 int    client_len;
 pid_t  pid;
 struct sockaddr_in     server, client;
 struct sigaction       mysignal;

 printf("%s v %s"
        "\n----------\n"
        "no features, no performance, no sense, no nothing...\n\n"
        ,SERVERNAME, SERVERVERSION);
 if((server_socket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1){
        printf("socket() failed\n");
        return(-1);
 }

 /* ****************************************socket:***/
 server.sin_family = AF_INET;
 server.sin_addr.s_addr = htonl(0);
 server.sin_port = htons(LISTENPORT);
 client_len = sizeof client;

 /* ****************************************signalling:***/
 mysignal.sa_handler = child_exit_handler;
 sigfillset(&mysignal.sa_mask);
 mysignal.sa_flags = SA_RESTART;
 sigaction(SIGCHLD, &mysignal, 0);

 if(bind(server_socket, (struct sockaddr*)&server, sizeof server) != 0){
        printf("bind() failed\n");
        return(-1);
 }
 if(listen(server_socket, MAXQUEUELEN) != 0){
        printf("listen() failed\n");
        return(-1);
 }
 while( -1 != (new_client = accept(server_socket, (struct sockaddr*)&client, &client_len))){
  if((pid = fork()) < 0){
        printf("fork() failed!\n");
        return(-1);
  }else if(pid == 0){   /* im child */
        close(server_socket);
        read_http_request(new_client);
        send_to_client("HTTP/1.0 200 OK\r\n"
                "Server: RWS\r\n"
                "Content-Type: text/html\r\n\r\n"
                "<html><head><title>RWS - ein kleiner Webserver</title></head>\n"
```

```
                "<body><h1>RWS - ein kleiner Webserver</h1>\n"
                "<p>Nur zum Testen, ein kleiner text...</p><hr>\n"
                "<h9>(c)2005 richard weinberger <richard@nod.at></h9></body></html>\r\n"
                "\r\n"
                , new_client);
        printf("child: beende mich.\n");
        _exit(0);
        printf("child: ich lebe noch -> BUG BUG BUG!\n");
  }else{
        close(new_client);
        printf("close()\n");
  }
 }
 printf("cu...\n");
 return(0);
}


/* ****************************************send_2_client()**
 */
void send_to_client(const char *message, int client){
 unsigned int   len;
 len = strlen(message);
 write(client, message, len);
}

/* ****************************************http_request()**
 */
void read_http_request(int client){
 char                   buffer[BUFFERLEN];
 int                    read_count;
 unsigned short int     ok = 0;
 while(ok < 2){
  read_count = read(client, buffer, BUFFERLEN);
  if(buffer[read_count - 1] == '\n' && buffer[read_count - 2] == '\r'){
        ok++;
        if(buffer[read_count - 3] == '\n') ok++;
  }
 }
}

/* ********************** signal handler: ***************
 * child_exit_handler()
 * ********************************************************/
void child_exit_handler(){
 while (waitpid(-1, NULL, WNOHANG) > 0);
}
```

22.7.5 'IPv4 socket establishment'

kann man zB. mit folgendem 'C'-Code Macro implementieren:

```
#include <sys/socket.h>
#define SO_REUSEPORT 15

#define MAKE_LISTEN_PORT( fdLISTEN, IpPort ) {\
    struct sockaddr_in  s_address;              \
    char        hostName[256];                  \
    gethostname(hostName, 255);                 \
    fdLISTEN = socket(AF_INET,SOCK_STREAM,0);           \
    s_address.sin_family      = AF_INET;                \
    s_address.sin_addr.s_addr = htonl(INADDR_ANY);      \
    s_address.sin_port        = htons(atoi(IpPort));    \
    int one=1;  \
    setsockopt(fdLISTEN, SOL_SOCKET, SO_REUSEADDR, &one, sizeof one); /*+SO_REUSEPORT*/ \
    if(bind(fdLISTEN,(struct sockaddr *) &s_address, sizeof(s_address))){       \
       perror("bind failed"); exit(EXIT_FAILURE);       \
    }   \
    listen(fdLISTEN,5); \
  }

main(){int fd1;  MAKE_LISTEN_PORT( fd1, "55555" ); ... }
```

22.7.6 serverseitiges Warten auf 'IPv4' Anfragen

ist mit folgendem 'C'-Code Macro zu erwirken:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

#define WAIT_FOR_REQUEST( fdCOMM, fdLISTEN, IpPort ) {\
    char        hostName[256];                  \
    gethostname(hostName, 255);                 \
    printf("server: %s@%s waiting at port:%d\n", argv[0], hostName, atoi(IpPort));      \
    fdCOMM  = accept(fdLISTEN, NULL, NULL);     \
  }

main(){
  int  fdc,fd1;
  char rxByteArray[4096], outstr[1000];

  //  ... (Vorbereitungen) ...

  WAIT_FOR_REQUEST( fdc, fd1, "55555" );
  r1=read(fdCOMM, rxByteArray, sizeof(rxByteArray)-2 ));

  //  ... (Verarbeitung) ...

RueckAntwort:
  write(fdCOMM, outstr, strlen(outstr));
  close(fdCOMM);
}
```

22.7.7 Bsp. f. einen HTTP Server

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define IpPort  (argv[1])
#include "wsock.h"              //Macros MAKE_LISTEN_PORT, WAIT_FOR_REQUEST,

    /*MAIN:*/
int main(int argc,char **argv){
  int           fdPORT, fdCOMM, r1=0;
  char          rxdata[4096], outstr[4096];
  if(argc<2){ printf("usage: %s <PortNumber>",*argv); exit(0);}
  MAKE_LISTEN_PORT( fdPORT, IpPort );
  for(;;){
    WAIT_FOR_REQUEST( fdCOMM, fdPORT, IpPort );
    if(1 > (r1=read(fdCOMM, rxdata, sizeof(rxdata)-2 ))) break;
    MACH_ABSCHLUSS_NULL: rxdata[r1]=0;
    inhaltAnzeigeFunctionSieheHeaderfile(rxdata);
      /*antworten:*/
    snprintf( outstr, sizeof(outstr)-1,
        "HTTP/1.0 200 OK\r\nServer: RWS\r\nContent-Type: text/html\r\n\r\n"
        "<html><body bgcolor=tan><h1>RWS - kleiner Webserver</h1>\n"
        "<br>Nur zum Testen, ein kleiner text...\n"
        "<hr><small>(c)2005 richard weinberger</small></body></html>\r\n\r\n"
        );
    write(fdCOMM, outstr, strlen(outstr));
    close(fdCOMM);
    if(strstr(rxdata,"serverstop")) break; /*"geheim"-shutdown URLstring*/
  }
```

```
  shutdown(fdPORT,SHUT_RDWR);
  close(fdPORT);
  printf("\nserver \"%s\" terminated.\n",*argv);
}/*end of main*/
```

Listing 48: wsock.h

```
#ifndef WSOCK_H
#define WSOCK_H
/* File wsock.h  cfs@PdG,Pe2
 */
#include <sys/socket.h>
//#include <poll.h>
//#include <fcntl.h>
//#include <sys/types.h>
#define SO_REUSEPORT 15

#define MAKE_LISTEN_PORT( fdLISTEN, IpPort ) {\
    struct sockaddr_in  s_address;              \
    char        hostName[256];                  \
    gethostname(hostName, 255);                 \
    fdLISTEN = socket(AF_INET,SOCK_STREAM,0);           \
    s_address.sin_family      = AF_INET;                \
    s_address.sin_addr.s_addr = htonl(INADDR_ANY);      \
    s_address.sin_port        = htons(atoi(IpPort));    \
    int one=1;  \
    setsockopt(fdLISTEN, SOL_SOCKET, SO_REUSEADDR, &one, sizeof one); /*+SO_REUSEPORT*/ \
    if(bind(fdLISTEN,(struct sockaddr *) &s_address, sizeof(s_address))){       \
       perror("bind failed"); exit(EXIT_FAILURE);       \
    }   \
    listen(fdLISTEN,5); \
  }

#define WAIT_FOR_REQUEST( fdCOMM, fdLISTEN, IpPort ) {\
    char        hostName[256];                  \
    gethostname(hostName, 255);                 \
    printf("server: %s@%s waiting at port:%d\n", argv[0], hostName, atoi(IpPort));      \
    fdCOMM  = accept(fdLISTEN, NULL, NULL);     \
  }

int inhaltAnzeigeFunctionSieheHeaderfile(char *s){
  printf("\n*--------new-message:------------*\n");
  for(;*s;s++){
    if(isprint(*s)) putchar(*s);
    else printf("(0x%02X)",*s);
    if('\n'==*s) putchar('\n');
  }
  printf("\n*--------message-end-------------*\n");
}
#endif
```

22.7.8 Bsp. f. eine HTTP client Anfrage

```
/* File wclnt.c
 * Socket-Programmierung, cfs@GcF
 */
 #include <sys/types.h>
 #include <sys/socket.h>
 #include <stdio.h>
 #include <netinet/in.h>
 #include <arpa/inet.h>
 #include <unistd.h>
 #include <netdb.h>
 #define IpPort 80

int main(int argc,char **argv){
  int           fd1, result, l, n;
  char          ch='A', s1[4096], *s, hname[256];
  struct sockaddr_in    address;

/*** socket configurieren: ***/
```

```
  address.sin_family=AF_INET;
  address.sin_addr.s_addr = inet_addr("10.10.10.30");
  address.sin_port = htons(IpPort);     //(NetworkByteOrder)

  s=(char*)malloc(16384);
  *s=0;

  for(;;){
    printf("client \"%s\" @%s\n",*argv,(gethostname(hname,255),hname));
    fd1=socket(AF_INET,SOCK_STREAM,0);  /* == socket erzeugen und verbinden **/
    if((result=connect(fd1,(struct sockaddr *) &address,sizeof(address))) < 0){
      perror("cant connect\n");
      break;
    }else{
      sprintf(s,
        "GET /SMUE/srcvw.php?fn=weliza/weliza1.c&nr=123 HTTP1.0/\n"
        "Host: localhost:80\n"
        "\r\n\r\n"
      );
      write(fd1,s,l=strlen(s));
      if(*s == 'C') break;
      while( read(fd1,s,16383) > 0 ){
        printf("Reply from server: /%s/\n",s);
        usleep(100000L);
      }
      close(fd1);
    }
    break;
  }
  free(s);
  exit(0);
}
```

22.7.9 Bsp. f. serverseitige openSSL encryption

```
/* File srv03ssl.c , cfs@Pe2
 * HTTPS-Webserver
 *  ::#define- macros in separate header file
 * shows message from Web-Browser
 * +sends very simple reply
 * compile: gcc srv03ssl.c -o srvs03
 * run    : ./srvs03 <portNr>  (zB "./srvs03 4433")
 * stop   : Strg+C   oder "serverstop" im URLstring
 */
/* Certificate Generation:
#gen ca private key
 mkdir ca
 openssl genrsa -out ca/privkey.pem

#create server cert request
 mkdir ca/private
 openssl req -x509 -days 7200 -newkey rsa:1024 -keyout ca/private/ca.key -out ca/ca.crt

#create server cert request
 mkdir -p server/private
 openssl genrsa -out server/private/server.key 1024
 openssl req -new -key server/private/server.key -out server/server.csr

#create client cert request
 mkdir -p client/private
 openssl genrsa -out client/private/client.key 1024
 openssl req -new -key client/private/client.key -out client/client.csr

#sign certs:
 openssl x509 -req -days 7200 -in server/server.csr -CA ca/ca.crt -CAkey ca/private/ca.key -
     CAcreateserial -out server/server.crt
 openssl x509 -req -days 7200 -in client/client.csr -CA ca/ca.crt -CAkey ca/private/ca.key -CAserial
     ca/ca.srl -out client/client.crt

#check, using openssl a Server-Fake:
 openssl s_server -CAfile ca/ca.crt -cert server/server.crt -key server/private/server.key -Verify 1
```

```
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <err.h>
//#include <stdarg.h>

#include <openssl/bio.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <openssl/pem.h>
#include <openssl/x509.h>
#include <openssl/x509_vfy.h>

#define SSL_CA_CRT      "./cert/ca/ca.crt"
#define SSL_SERVER_CRT  "./cert/server/server.crt"
#define SSL_SERVER_KEY  "./cert/server/private/server.pem"
#define fatalx(a)       {perror(a);exit(-1);}

#define IpPort  (argv[1])
//statt #include "wsock.h": -> hierherkopiert:
  /* File wsock.h  cfs@PdG,Pe2
   */
//#include <sys/socket.h>
#define MAKE_LISTEN_PORT( fdLISTEN, IpPort ) {\
    struct sockaddr_in  s_address;              \
    char        hostName[256];                  \
    gethostname(hostName, 255);                 \
    fdLISTEN = socket(AF_INET,SOCK_STREAM,0);           \
    s_address.sin_family      = AF_INET;                \
    s_address.sin_addr.s_addr = htonl(INADDR_ANY);      \
    s_address.sin_port        = htons(atoi(IpPort));    \
    int one=1;  \
    setsockopt(fdLISTEN, SOL_SOCKET, SO_REUSEADDR, &one, sizeof one); /*+SO_REUSEPORT*/ \
    if(bind(fdLISTEN,(struct sockaddr *) &s_address, sizeof(s_address))){       \
       perror("bind failed"); exit(EXIT_FAILURE);       \
    }   \
    listen(fdLISTEN,5); \
  }

#define WAIT_FOR_REQUEST( fdCOMM, fdLISTEN, IpPort ) {\
    char        hostName[256];                  \
    gethostname(hostName, 255);                 \
    printf("server: %s@%s waiting at port:%d\n", argv[0], hostName, atoi(IpPort));      \
    fdCOMM  = accept(fdLISTEN, NULL, NULL);     \
  }

int inhaltAnzeigeFunctionSieheHeaderfile(char *s){
  printf("\n*--------new-message:------------* Len=%d\n", strlen(s));
  for(;*s;s++){
    if(isprint(*s)) putchar(*s);
    else printf("(0x%02X)",*s);
    if('\n'==*s) putchar('\n');
  }
  printf("\n*--------message-end-------------*\n");
}
/*---End-Of-wsock.h---*/




  /**********/
   /**MAIN**/
    /*    */
     /*  */
      /**/
  /**********/
```

```
   /**MAIN**/
    /*    */
     /*  */
      /**/
int main(int argc,char **argv){ if(argc<2){printf("usage: %s <port>",*argv); exit(0);}
  int           fdLISTEN, fdCOMM, r1=0;
  char          rxdata[4096], outstr[4096];
  SSL_CTX       *ctx1;
  SSL           *ssl1;
  BIO           *sbio;


    /*SSL preparation:*/
  SSL_load_error_strings();
  OpenSSL_add_ssl_algorithms();
  if(NULL==(ctx1 = SSL_CTX_new(SSLv23_server_method())))                fatalx("ctx");
  if(!SSL_CTX_load_verify_locations(ctx1, SSL_CA_CRT, NULL))            fatalx("verify");
  SSL_CTX_set_client_CA_list(ctx1, SSL_load_client_CA_file(SSL_CA_CRT));
  if(!SSL_CTX_use_certificate_file(ctx1, SSL_SERVER_CRT, SSL_FILETYPE_PEM))  fatalx("cert");
  if(!SSL_CTX_use_PrivateKey_file(ctx1, SSL_SERVER_KEY, SSL_FILETYPE_PEM))   fatalx("key");
  if(!SSL_CTX_check_private_key(ctx1))  fatalx("cert/key");
  SSL_CTX_set_mode(ctx1, SSL_MODE_AUTO_RETRY);
  //SSL_CTX_set_verify(ctx1, SSL_VERIFY_PEER | SSL_VERIFY_FAIL_IF_NO_PEER_CERT, NULL);//
      willClientCert
  SSL_CTX_set_verify(ctx1, SSL_VERIFY_NONE, NULL/*callback*/);
  SSL_CTX_set_verify_depth(ctx1, 1);

    /*setup socket: socket()/bind()/listen() */
  MAKE_LISTEN_PORT( fdLISTEN, IpPort );

  for(;;){
    WAIT_FOR_REQUEST( fdCOMM, fdLISTEN, IpPort );
    //if((fdCOMM = accept(fdLISTEN, 0, 0)) == -1)  err(EX_OSERR, "accept");
    if(fdCOMM < 0){perror("TCP accept");exit(-2);}//err(EX_OSERR, "accept");
    //if(fdCOMM < 0) err(EX_OSERR, "accept");

    sbio = BIO_new_socket(fdCOMM, BIO_NOCLOSE);
    ssl1 = SSL_new(ctx1);
    SSL_set_bio(ssl1, sbio, sbio);

      //if(1 > (r1=read(fdCOMM, rxdata, sizeof(rxdata)-2 ))) break;
    if((r1 = SSL_accept(ssl1)) == -1)  warn("SSL_accept");
    printf("SSL_accept:%d(%s) err=%d\n",
        r1, ((0<r1)?"ok":((0>r1)?"fatal":"shut")), SSL_get_error(ssl1,r1) );
        //err(SSL_get_error(ssl1,r1),NULL);
        //if(1>r1)break;

    //do{
      if(0 > (r1=SSL_read(ssl1, rxdata, sizeof(rxdata)-2 ))){
        perror("**ERROR: SSL_read()<0!");
        break;
      }
    //}while(0==r1);
    MACH_ABSCHLUSS_NULL: rxdata[r1]=0;
    inhaltAnzeigeFunctionSieheHeaderfile(rxdata);
      /*antworten:*/
    snprintf( outstr, sizeof(outstr)-1,
        "HTTP/1.0 200 OK\r\nServer: RWS\r\nContent-Type: text/html\r\n\r\n"
        "<html><body bgcolor=tan><h1>RwXh's - kleiner https://Webserver</h1>\n"
        "<br>SSL/TLS SSLv2+SSLv3+TLSv1 w/o client cert request\n"
        "<hr><small>(c)2005/rw 2015/xh</small></body></html>\r\n\r\n"
        );
    r1=SSL_write(ssl1, outstr, strlen(outstr));
printf("done: SSL_write(): rc=%d.\n", r1);
    r1=SSL_shutdown(ssl1);
printf("done: SSL_shutdown(): rc=%d(%s).\n", r1, ((0<r1)?"ok":((0>r1)?"fatal":"pending")) );
    r1=close(fdCOMM);
printf("done: close(fdCOMM): rc=%d(%s)\n", r1, (0==r1)?"ok":"error");
    BIO_free_all(sbio);
printf("done: BIO_free_all().\n");
    SSL_CTX_free(ctx1);
printf("done: SSL_CTX_free().\n");
    if(strstr(rxdata,"serverstop")) break; /*"geheim"-shutdown URLstring*/
```

```
  }
  shutdown(fdLISTEN,SHUT_RDWR);
  close(fdLISTEN);
  printf("\nserver \"%s\" terminated.\n",*argv);

}/*end of main*/
```

22.7.10 Bsp. f. clientseitige openSSL encryption

```
/* ssl04.c      cfs@Pe1,Pe6
 * home made ssl client, for learning purpose; from 'openssl in C.c'
 *
 * compile : 'gcc ssl04.c -lssl -oss4'
 * run     : './ss4  10.10.63.61  443'
 *
 *
*
*from file:
* A little while ago, I was working on a client/server communication module,
* and I wanted it to be secure.
* Looking at the documentation I could find,
* I quickly figured out that it wouldn't necessarily be easy to do.
*
* Amongst the first issues are the validity of the server's certificate,
* which, as I didn't want to battle with this, I decided to skip
*
* The example below shows how to connect and send/receive data on an SSL-encrypted socket
*
* To build this example program using gcc:
*   gcc -Wall -lssl -lcrypto -o ssl-demo ssl-demo.c
* Have fun with SSL!
*
 *not needed:
 * //#include <sys/socket.h>
 * //#include <sys/types.h>
 * //#include <netinet/in.h>
 * //#include <unistd.h>
 * //#include <errno.h>
 * //#include <openssl/rand.h>
 * //#include <openssl/err.h>
 * //#include <string.h>
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>     //4 getopt(),optarg,
#include <netdb.h>
#include <openssl/ssl.h>

// Simple structure to keep track of the handle, and
// of what needs to be freed later.
typedef struct {
    int         sockfd;
    SSL         *SSLssl1;
    SSL_CTX     *SSLctx1;
} myConnType;

  //For this example, we'll be testing on openssl.org
//#define SERVER_URL  "www.openssl.org"
//#define SERVER_URL  "www.google.at"
#define SERVER_URL  "tr1"
#define SERVER_PORT "4433"
#define ERRprint(n) { ERR_print_errors_fp(stderr); return(n);}



    /* Macro Code for command line parameter processing */
#define COMMANDLINE_PARAMETER_PROCESSING()      { \
  int opt;                                        \
  while ((opt = getopt(argc, argv, "hu:p:")) != -1) {   \
    switch(opt){                                  \
```

```
    case 'u':   strncpy(urlstr, optarg, sizeof(urlstr)-1); break;       \
    case 'p':   strncpy(portstr, optarg, sizeof(portstr)-1); break;     \
    case 'h':   \
    default:    printf("** usage: %s [-h] [-u URL] [-p port]\n",*argv);\
                exit(0); break; \
    }   \
  }     \
}



int sslConnect( myConnType *cc, char *srvName, unsigned int portint){
  printf("ask '%s'\n",srvName);

  //1.Establish regular tcp connection
  int                   erv;
  struct hostent        *host;
  struct sockaddr_in    clntsock;
  host        = gethostbyname(srvName);
printf("connecting %s(=%8X):%d...\n", srvName, *host->h_addr, portint);
  cc->sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if(0>cc->sockfd){ perror ("Socket"); cc->sockfd=-1; }else{
    clntsock.sin_family = AF_INET;
    clntsock.sin_port   = htons(portint);
    clntsock.sin_addr   = *((struct in_addr *) host->h_addr);
    bzero( &(clntsock.sin_zero), 8);
    if(-1== connect( cc->sockfd, (struct sockaddr*) &clntsock, sizeof(struct sockaddr))){
      perror ("**Error: TCP connect()."); return(cc->sockfd= -1);
    }
  }

  //2.Add SSL
  if(0<=cc->sockfd){
    SSL_load_error_strings ();  //Register errorstrings for libcrypto & libssl
    SSL_library_init ();        //Register available ciphers+digests
    if(NULL == (cc->SSLctx1 = SSL_CTX_new (SSLv23_client_method() )))  ERRprint(-10);
    if(NULL == (cc->SSLssl1 = SSL_new(cc->SSLctx1)))  ERRprint(-11);
    if(! SSL_set_fd(cc->SSLssl1, cc->sockfd))  ERRprint(-12);
    if(1 != SSL_connect (cc->SSLssl1))  ERRprint(-13); //Initiate SSL handshake
  }else{  printf("me got bad sockfd.\n"); return(-3);}

    /*Ueberpruefung des Server-Certifikats: fehlt! */
  CHECK_OF_SERVER_CERTIFICATE_VALIDITY_is_MISSING_here:
  return(0);
}


void sslDestroy(myConnType *cc){
  if(1>cc->sockfd)  close(cc->sockfd);
  if(cc->SSLssl1){
    SSL_shutdown(cc->SSLssl1);
    SSL_free(cc->SSLssl1);
  }
  if(cc->SSLctx1) SSL_CTX_free(cc->SSLctx1);
}



void sslWrite (myConnType *cc, char *text){  // Write text to the connection
  if(cc) SSL_write( cc->SSLssl1, text, strlen (text));
}

  //Read much available text from connection
char *sslRead(myConnType *cc){
  #define       BLKsz  1024
  #define       BLKnum 10
  static char   rxbuf[BLKnum*BLKsz];  //muss persistent!
  int           rxcnt, count;
  if(cc) for( count=0; count<BLKnum; count++ ){
    rxcnt = SSL_read(cc->SSLssl1, rxbuf+count*BLKsz, BLKsz);
    if(rxcnt < BLKsz){
```

```
      rxbuf[count*BLKsz + rxcnt] = '\0';
      break;
    }
  }
  return(rxbuf);
}



   /***********************/
    /*Very basic M.A.I.N.*/
     /*we send "GET /" */
      /*  and print  */
        /*response.*/
          /*     */
            /**/
int main (int argc, char **argv){ if(argc<1){printf("usage: %s <url>\n");exit(0);}
  char          *re1, urlstr[1023]=SERVER_URL, portstr[31]=SERVER_PORT;
  int           rv1;
  unsigned int  portint;
  myConnType    cc1={ 0, NULL, NULL};

  COMMANDLINE_PARAMETER_PROCESSING();
  portint=atoi(portstr);
  if(0>(rv1=sslConnect(&cc1, urlstr, portint)) ){
    printf("Connection is hin:%d/[%s:%d]\n",rv1, urlstr,portint );
  }else{
    sslWrite(  &cc1, "GET /\r\n\r\n");
    re1= sslRead(&cc1); printf("%s\n", re1);
  }
  sslDestroy(&cc1);
  return(0);
}
```

22.7.11 Benchmark Timing Example

Listing 49: Benchmark Timing Example

```
/* File httpRqTmg01.c, xh@QlF
 * Socket-Programmierung +GetTime => Benchmark
 * compile: 'gcc httpRqTmg01.c -o hRT1 -lrt'
 * run:     './hRT1'
 * stop:    (self halting)
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netdb.h>

#define IpPort  ((argc>1)?(argv[1]):"127.0.0.1")

int main(int argc,char *argv[]){
  int   fd1;
  char  anfrage[1024],
        antwort[4096],
        *errmsg;
  struct sockaddr_in    address;
  struct timespec       wtime = {0L, 50000000L}, //100ms
                        t1, t2;

  if((argc>1)?argv[1][1]=='h':0){
    printf("usage: '%s [IpAdresse] [p]'\n", argv[0] );
    exit(EXIT_SUCCESS);
  }
```

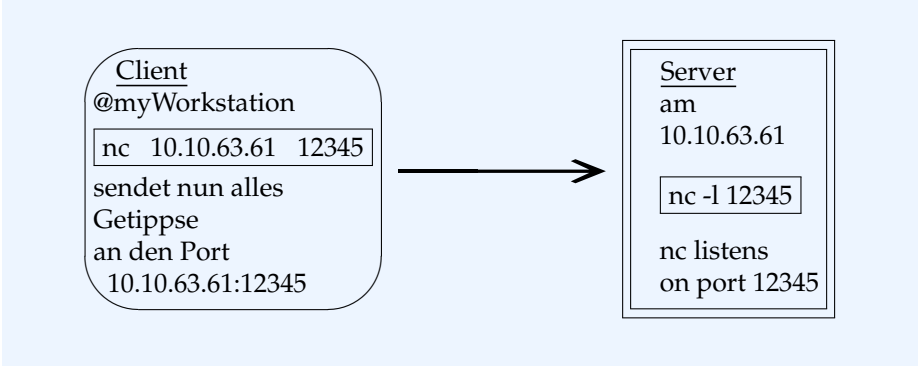
```
  address.sin_family=AF_INET;
  address.sin_addr.s_addr = inet_addr(IpPort);
  address.sin_port = htons(80); //(NetworkByteOrder)
  fd1 = socket(AF_INET, SOCK_STREAM, 0 );

  if(0 > connect(fd1,(struct sockaddr *) &address,sizeof(address))){
    asprintf(&errmsg,"cant connect '%s'", IpPort);
    perror(errmsg);
    free(errmsg);
    exit(EXIT_FAILURE);
  }else{
    sprintf(anfrage,
        "GET /wahuschi.html HTTP1.0/\nHost: localhost:80\n"
        "\r\n\r\n");
    clock_gettime(CLOCK_MONOTONIC, &t1 );
    write(fd1, anfrage, strlen(anfrage) );
    while( read(fd1, antwort, 4095) > 0 ){
      clock_gettime(CLOCK_MONOTONIC, &t2 );
      printf("reply time from t1=%ld.%09ld to t2=%ld.%09ld diff=%f[ms] \n",
        t1.tv_sec, t1.tv_nsec, t2.tv_sec, t2.tv_nsec,
        (t2.tv_sec-t1.tv_sec)*1000.0 + (t2.tv_nsec-t1.tv_nsec)/1000000.0 );
      if((argc>2)?(argv[2][0]=='p'):0){  //should print this?
        printf("Reply from server: '%s'\n", antwort);
      }
      nanosleep( &wtime, NULL);
    }
    close(fd1);
  }
}
```

22.7.12 IoT Internet-of-Things Device

→ s. Kap.22.9, Seite 391

22.8 supersimpelServerMit'netcat'

Das Utility *'netcat'* bzw. *'nc'* kopiert zwischen LAN-Port und Konsole



22.8.1 im Detail

Mit einem Shellscript der Art

Listing 50: service.sh

```
#!/bin/sh
while((1)); do
  echo -n -e "give command [adhilnrt^D]:";
  read x;
  if [ $x == "d" ]; then
    echo "---> date=$(cat /sys/class/rtc/rtc0/date)";
  elif [ $x == "t" ]; then
    echo "---> time=$(cat /sys/class/rtc/rtc0/time)";
  elif [ $x == "a" ]; then
```

```
    echo "---> MACaddress=$(cat /sys/class/net/eth0/address)";
  elif [ $x == "h" ]; then
    echo "---> HOSTname: $(cat /etc/hostname)";
  elif [ $x == "i" ]; then
    echo "---> IP: $(ip a)";
  elif [ $x == "l" ]; then
    echo "---> IP: $(ip link)";
  elif [ $x == "n" ]; then
    echo "---> IP: $(ip neigh)";
  elif [ $x == "r" ]; then
    echo "---> IP: $(ip rou)";
  else
  echo "------------------------------>>> ur command was: /$x/";
  fi
done
```

und dem Haupt-Script

Listing 51: sv1.sh

```
#!/bin/sh
mkfifo /tmp/p50100
while((1)); do
cat /tmp/p50100 |    \
  ./service.sh  2>&1  |      \
  nc -k -l 50100 >    \
  /tmp/p50100
done
```

entsteht ein stinksimples IoT Device, das mehrere Abfragekommandos versteht und entsprechend antwortet:

Auf Hostrechner xy startest Du

```
./sv1.sh
```

Auf der Clientworkstation kannst nun mit

```
nc myServer 50100
```

die Kommandos ausprobieren.

(den Client beendet man mit Strg+D und den Server mit Strg+C)

Mit zusätzlicher Absicherung durch einen ssh Tunnel ist das gar nicht mehr so schlecht. Sicherheitsbedenken gibt es aber gegenüber allen Shellscripten. Die Shells sind teilweise unglaublich uralt und enthalten noch schauderhafte security holes. Dies würde trotz ssh einem *'trusted client'* die unerwünschte Übernahme (hitch hiking) des Rechners ermöglichen.

22.8.2 file contents reporting IoT Server (risky!)

Listing 52: one file reading server

```
/* File IoTreadfile1.c , cfs@RcJ
 * +sends very simple reply
 * compile: gcc IoTreadfile1.c -o IoTrf1
 * run    : ./IoTrf1
 * stop   : Strg+C
 */
#include <stdlib.h>     /* exit() */
#include <stdio.h>      /* printf() */
#include <string.h>     /* strlen() */
#include <fcntl.h>      /* O_RDONLY */
#include <arpa/inet.h>  /* SOCK_STREAM ...*/
#define IpPort  (argv[1])
#define INFILE  (argc>2 ? argv[2]:"antwort1")
//#define SO_REUSEPORT 15

    /*MAIN:*/
int main(int argc,char **argv){
  int           fdLISTEN, fdCOMM, r1=0, fdFILE;
  char          anfr[4096], antw[4096];
  struct sockaddr_in    s_address;

  if(argc<2){
    printf("usage: %s <PortNumber> <InputFileName>",*argv);
    exit(EXIT_FAILURE);
  }

    /*MAKE_LISTEN_PORT( fdPORT, IpPort );*/
  fdLISTEN = socket(AF_INET,SOCK_STREAM,0);
  s_address.sin_family      = AF_INET;
  s_address.sin_addr.s_addr = htonl(INADDR_ANY);
  s_address.sin_port        = htons(atoi(IpPort));
  int one=1;
  setsockopt(fdLISTEN, SOL_SOCKET, SO_REUSEADDR, &one, sizeof one); /*+SO_REUSEPORT*/

  if(bind(fdLISTEN,(struct sockaddr *) &s_address, sizeof(s_address))){
     perror("bind failed"); exit(EXIT_FAILURE);
  }
  listen(fdLISTEN,5);

  for(;;){
      /*WAIT_FOR_REQUEST( fdCOMM, fdPORT, IpPort ): */
    fdCOMM  = accept(fdLISTEN, NULL, NULL);

      /*antworten:*/
    if(0> (fdFILE = open(INFILE, O_RDONLY))){
      perror(INFILE);
    }
    if(0> (r1=read(fdFILE, antw, sizeof(antw)-2))){
      perror(INFILE);
    }
    antw[r1]=0;
    close(fdFILE);
    write(fdCOMM, antw, strlen(antw));
    close(fdCOMM);
  }
  shutdown(fdLISTEN, SHUT_RDWR);
  close(fdLISTEN);
}/*end of main*/
```

22.8.3 any file reporting IoT Server (risky!)

Listing 53: any file reading server

```
/* File IoTreadfile2.c , cfs@RcJ
 * +sends file contents as reply
 * compile: gcc IoTreadfile2.c -o IoTrf2
 * run    : ./IoTrf2
 * stop   : Strg+C
 */
#include <stdlib.h>     /* exit() */
#include <stdio.h>      /* printf() */
#include <string.h>     /* strlen() */
#include <fcntl.h>      /* O_RDONLY */
#include <arpa/inet.h>  /* SOCK_STREAM ...*/
#define IpPort  (argv[1])
//#define SO_REUSEPORT 15

    /*MAIN:*/
int main(int argc,char **argv){
  int           fdLISTEN, fdCOMM, r1=0, fdFILE;
  char          anfr[4096], antw[4096];
  struct sockaddr_in    s_address;

  if(argc<2){
    printf("usage: %s <PortNumber>",*argv);
    exit(EXIT_FAILURE);
  }

    /*MAKE_LISTEN_PORT( fdPORT, IpPort );*/
  fdLISTEN = socket(AF_INET,SOCK_STREAM,0);
  s_address.sin_family      = AF_INET;
  s_address.sin_addr.s_addr = htonl(INADDR_ANY);
  s_address.sin_port        = htons(atoi(IpPort));
  int one=1;
  setsockopt(fdLISTEN, SOL_SOCKET, SO_REUSEADDR, &one, sizeof one); /*+SO_REUSEPORT*/

  if(bind(fdLISTEN,(struct sockaddr *) &s_address, sizeof(s_address))){
     perror("bind failed"); exit(EXIT_FAILURE);
  }
  listen(fdLISTEN,5);

  for(;;){
      /*WAIT_FOR_REQUEST( fdCOMM, fdPORT, IpPort ): */
    fdCOMM  = accept(fdLISTEN, NULL, NULL);
    if(-1< (r1=read(fdCOMM, anfr, sizeof(anfr)-2))){
      anfr[r1]=0;
      anfr[strcspn(anfr,"\n\r")]=0; /*eliminate CR LF...*/
        /*antworten:*/
      if(0> (fdFILE = open(anfr, O_RDONLY))){
        perror(anfr);
      }
      if(0> (r1=read(fdFILE, antw, sizeof(antw)-2))){
        perror(anfr);
      }
      antw[r1]=0;
      close(fdFILE);
      write(fdCOMM, antw, strlen(antw));
      close(fdCOMM);
    }
  }
  shutdown(fdLISTEN, SHUT_RDWR);
  close(fdLISTEN);
}/*end of main*/
```

22.8.4 command executing IoT Server (extreme risky!)

Listing 54: command executing server

```
/* File IoTreadfile3.c , cfs@RcJ
 * +sends command as reply
 * compile: gcc IoTreadfile3.c -o IoTrf3
 * run    : ./IoTrf3
 * stop   : Strg+C
 */
#include <stdlib.h>     /* exit() */
#include <stdio.h>      /* printf() */
#include <string.h>     /* strlen() */
#include <fcntl.h>      /* O_RDONLY */
#include <arpa/inet.h>  /* SOCK_STREAM ...*/
#define IpPort  (argv[1])
//#define SO_REUSEPORT 15

    /*MAIN:*/
int main(int argc,char **argv){
  int           fdLISTEN, fdCOMM, r1=0, fdFILE;
  char          anfr[4096], antw[4096];
  struct sockaddr_in    s_address;

  if(argc<2){
    printf("usage: %s <PortNumber>",*argv);
    exit(EXIT_FAILURE);
  }

    /*MAKE_LISTEN_PORT( fdPORT, IpPort );*/
  fdLISTEN = socket(AF_INET,SOCK_STREAM,0);
  s_address.sin_family      = AF_INET;
  s_address.sin_addr.s_addr = htonl(INADDR_ANY);
  s_address.sin_port        = htons(atoi(IpPort));
  int one=1;
  setsockopt(fdLISTEN, SOL_SOCKET, SO_REUSEADDR, &one, sizeof one); /*+SO_REUSEPORT*/

  if(bind(fdLISTEN,(struct sockaddr *) &s_address, sizeof(s_address))){
     perror("bind failed"); exit(EXIT_FAILURE);
  }
  listen(fdLISTEN,5);

  for(;;){
      /*WAIT_FOR_REQUEST( fdCOMM, fdPORT, IpPort ): */
    fdCOMM  = accept(fdLISTEN, NULL, NULL);
    if(-1< (r1=read(fdCOMM, anfr, sizeof(anfr)-2))){
      anfr[r1]=0;
      anfr[strcspn(anfr,"\n\r")]=0; /*eliminate CR LF...*/

        /*antworten:*/
      system(strncat(anfr,"> aw3", sizeof(anfr)));
      if(0> (fdFILE = open("aw3", O_RDONLY))){
        perror("aw3");
      }
      if(0> (r1=read(fdFILE, antw, sizeof(antw)-2))){
        perror(anfr);
      }
      antw[r1]=0;
      close(fdFILE);
      write(fdCOMM, antw, strlen(antw));
      close(fdCOMM);
    }
  }
  shutdown(fdLISTEN, SHUT_RDWR);
  close(fdLISTEN);
}/*end of main*/
```

22.8.5 mit ssh - secure shell

22.9 web based IoT sensor

22.9.1 Messwert Server C Code

Ganz einfaches Beispiel für einen http Server,
der (hier fingierte zufzi) Messwerte antwortet:

```
/* File tmpSens1.c, XH, 18Mar15 Fsst3a
 * beantwortet Anfragen mit (simuliertem) Temperaturwert
 * compile: gcc tmpSens1.c -o ts1
 * run:     ./ts1  10101
 * stop:    Strg+C
 * Bedienung: im Firefox mit "http://10.10.63.61:10101/" abrufen */

 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 #include <sys/types.h>
 #include <sys/socket.h>
 #include <netinet/in.h>
 #include <arpa/inet.h>
 #include <unistd.h>
 #include <fcntl.h>

#define IpPort  (argv[1])
int peep(char*);

int main(int argc,char **argv){
  int           fds,fdc,lenc,r1=0;
  char          buf[4096], buf2[4096],
                hname[256],
                ADCval[10];
  double        TempKorrFaktor=10.72;
  struct
  sockaddr_in   s_address, c_address;

  if(argc<2){ printf("usage: %s <PortNumber>",*argv); exit(0); }

  gethostname(hname,255);
  fds=socket(AF_INET,SOCK_STREAM,0);
  s_address.sin_family=AF_INET;
  s_address.sin_addr.s_addr = htonl(INADDR_ANY);
  s_address.sin_port = htons(atoi(IpPort));
  int one=1; setsockopt(fds, SOL_SOCKET, SO_REUSEADDR, &one, sizeof one);
  if(bind(fds,(struct sockaddr *) &s_address, sizeof(s_address))){
   perror("bind failed"); abort();
  }
  listen(fds,5);
  for(;;){
    //printf("server \"%s\" @ \"%s\" waiting\n",*argv,hname);
    lenc=sizeof(c_address);
    fdc=accept(fds,(struct sockaddr *) &c_address, &lenc);
    ADCfd=open("adc0", O_RDONLY);
    read(ADCfd, ADCval, 5);
    close(ADCfd);
    sprintf(buf2,
      "HTTP/1.0 200 OK\r\nServer: RWS\r\nContent-Type: text/html\r\n\r\n"
      "<html><body>%dK</body></html>\r\n",
      atoi(ADCval)/TempKorrFaktor );
    write(fdc, buf2, strlen(buf2));
    close(fdc);
  }
  shutdown(fds,SHUT_RDWR);
  close(fds);
}
```

22.9.2 Messwert 'Prozessvisualisierung' HTML Code

Ein sehr stark simplifiziertes Anfangsbeispiel für die Abfrage mehrerer 'IoT' IP Adressen/ Ports mit http- Rückmeldungen:

```
<html><head>
<script>
  window.setInterval( "reloader()", 3000 );
  function reloader(){
    window.location.reload();
  }
</script></head>
<frameset  cols="100,100,100,100,100,100,*" rows="100,100,100,*">
 <frame src=http://10.10.63.61:11999/></frame>
 <frame src=http://10.10.63.61:11998/></frame>
 <frame src=http://10.10.63.61:11997/></frame>
 <frame src=http://10.10.63.245:11999/></frame>
 <frame src=http://10.10.63.61:10201/></frame>
 <frame src=http://10.10.63.61:10202/></frame>
 <noframes>
   <body>
     reloading every 3000ms
   </body>
 </noframes>
</frameset>
</html>
```

22.10 ssh Fernstart mehrerer Messwert Server

SSH kann auf der Zielmaschine (Host) einen Befehl (zB. Programmstart) ausführen. [4] Hier sind mehrere solcher Fernstarts in ein shellscript zusammengefasst:

```
#!/bin/bash
ssh -f -n dings@10.10.63.245 /home/dings/c/IoT/ts1 11999
ssh -f -n dings@10.10.63.61 "/home/dings/public_html/Fst3a14/IoT/ts1 11999"
ssh -f -n dings@10.10.63.61 "/home/dings/public_html/Fst3a14/IoT/ts2 11998"
ssh -f -n dings@10.10.63.61 "/home/dings/public_html/Fst3a14/IoT/ts3 11997"
```

(Passwort- Eingabe- Problem muasch ggf. lösen)

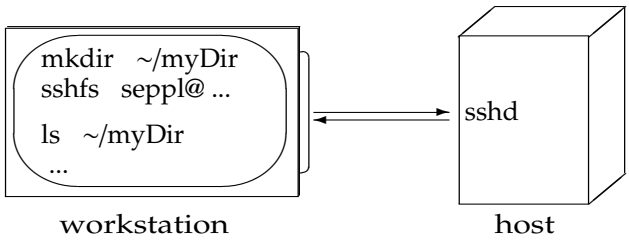
22.11 ssh Fernstart plus encrypted port forwarding

```
       /* ssh macht (auch) 'port forwarding'
          d.h. der lokale Port (hier 1234)
          wird zum Host-Port (hier 6666) durch'getunnelt',
          (ein User ist wegen Passwort anzugeben), mit: */

  ssh -f -L 1234:localhost:6666 ich@meinhost.mydomain

       /* zudem kannma am Host gleich ein Programm starten lassen
          (sinnvollerweise eines, das auf dem geforwardeten Port -hier 6666-
          kommuniziert): */

  ssh -f -L 1234:localhost:6666 ich@meinhost.mydomain "/meinPfad/meinProgramm"

       /* ausprobiertes Beispiel:
          (dann kann man 'http://localhost:1234' anbrausen) */

  ssh -f -L 1234:localhost:6666 cfs@tr3 "echo \"hallo\" | netcat -l 6666"


       /* encrypted 'Datums-Server';  local:1234 <---> tr3:6666
          -->
          Kommando  'netcat localhost 1234'
          liefert:  'Sun Feb 21 14:11:54 UTC 2016'
       */
  ssh -f -L 1234:localhost:6666 cfs@tr3 "while((1)); do date | netcat -l 6666; done"
```

[4] wenn man den Befehl — wie üblich — weglasst, startez a Kommando- *'shell'*

22.12 'sshfs' secure shell (remote) file system

'sshfs' 'holt' (mounts) ein entferntes (remote) Verzeichnis (directory) in den lokalen Dateibaum (etwa wie eine 'Netzlaufwerk', aber hier muss nix vorher 'freigegeben' werden — die Existenz des 'user' reicht).



Die Datenübertragung erfolgt *chiffriert (encrypted)* mittels *'ssh'* Protokoll. Über die Angabe von *username* und *password* erfolgt eine Authorisierungsprüfung (sodass nicht jede 'Pfeife' Zugriff hat)
Man schreibt

```
sshfs  user@machine:DirectoryPath  localMountPoint
```

zB.

```
mkdir  ~/myDir
ssh  seppl@10.10.63.245:public_html/glump  ~/myDir
```

→ Password: ('seppl's Passwort am '10.10.63.245')

sodann ist unter *'~/myDir'* der Inhalt des Verzeichnisses '/home/seppl/public_html/glump' des users 'seppl' am Rechner 10.10.63.245 verfügbar:

```
ls -l  ~/myDir
```

Wo 'seppl' Schreibrechte hat, hat er sie auch unter ~/myDir.

Aufgehoben wird dieser 'mount' mit

```
fusermount  -u  ~/myDir
```

aus der manpage 'man 1 sshfs':

Listing 55: man 1 sshfs

```
SSHFS(1)                        User Commands                SSHFS(1)
SSHFS version 2.0 April 2008

NAME    SSHFS - filesystem client based on ssh
SYNOPSIS
   mounting:
       sshfs [user@]host:[dir] mountpoint [options]

   unmounting:
       fusermount -u mountpoint

DESCRIPTION
       SSHFS  (Secure  SHell  FileSystem)  is a file system for Linux (and other operating
       systems with a FUSE implementation, such as Mac OS X or FreeBSD) capable of operat-
       ing  on  files  on  a remote computer using just a secure shell login on the remote
       computer. On the local computer where the  SSHFS  is  mounted,  the  implementation
       makes use of the FUSE (Filesystem in Userspace) kernel module. The practical effect
       of this is that the end user  can  seamlessly  interact  with  remote  files  being
       securely  served  over SSH just as if they were local files on his/her computer. On
       the remote computer the SFTP subsystem of SSH is used.
        [...]
```

22.12.1 IP-SocketProg in Python3

```
#!/usr/bin/python3
import os,socket,sys,time
from socket import AF_INET,SOCK_STREAM

IP=   "10.10.63.61"
PORT= 50100

if(1):
  s=socket.socket( AF_INET, SOCK_STREAM, 0)
  s.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
  s.bind( (IP,PORT) )
  s.listen(3)
  while(1):
    os.write(1,bytearray("wait...".encode('utf-8')))
    c,clnt= s.accept()
    os.write(1,bytearray(('connect from '+str(clnt)).encode('utf-8')))
    print(':',c.recv(1000))
    #c.send(bytearray("Hi!\r\n".encode('utf-8')))
    if(1):
      \textcolor{purple}{c.send(}bytearray(
        ("HTTP/1.0 200 OK\r\nServer: RWS\r\nContent-Type: text/html\r\n\r\n"
        +'<html><body bgcolor=red>reply from <h2>'+IP+':'+str(PORT)+'</h2>'
        +'<h3>(c)2005 richard weinberger <richard@nod.at></h3></body></html>\r\n\r\n'
        #+'\r\n'*99
        ).encode('utf-8'))
      )
    os.write(1,bytearray(" reply done\r\n".encode('utf-8')))

    #time.sleep(1)
    c.shutdown(socket.SHUT_RDWR)
    #time.sleep(1)
    c.close

  print('done se reply.')
#end
```

```
#!/usr/bin/python3
#       IPsocket-client, xh@TcO
import os,socket,sys
from socket import AF_INET,SOCK_STREAM

PORT= 50100

#       s.connect()
#       s.recv()        #This method receives TCP message
#       s.send()        #This method transmits TCP message
#       s.recvfrom()    #This method receives UDP message
#       s.sendto()      #This method transmits UDP message
#       s.close()       #This method closes socket
#socket.gethostname()   #Returns the hostname.


if(1):
  fd1=          socket.socket( AF_INET, SOCK_STREAM, 0)
  hostname='10.10.63.61'
  print( 'server hostname=['+hostname+']' )
  fd1.connect((hostname,PORT))
  if(1 < len(sys.argv)):
    rq= sys.argv[1]
  else:
    rq= 'Hi!'
  print( 'Request: [', rq, ']' )
  fd1.send( bytearray(rq.encode('utf-8')) )
  print( 'Reply:', fd1.recv(1024) )
  fd1.close()

print('Client exits')
```

Teil VIII
Signalverarbeitung

23 Rausch=Zufall=unvorhersehbar

⊗ **Rauschen = Zufall**

⊗ Ein Rausch-Signal ist unvorhersehbar
⊗ ein Bildungsgesetz ist **unbekannt** und **nicht erkennbar**
⊗ Signalabschnitte **wiederholen sich nicht**
⊗ manche gehorchen einer statistischen Verteilung wie zB. Normal-, Poisson- oder Gleichverteilung
⊗ sie gehorchen üblicherweise einer Gleichverteilung. Andere Verteilungen werden mit der *Monte-Carlo-Methode* errechnet.
⊗ pseudozufällige Bit-Folgen erzeugt man mit "rückgekoppelten" Schieberegistern

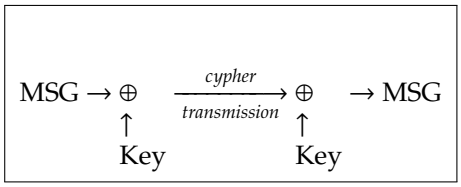
23.1 Pseudo-Zufall

⊗ 'PRN' ≡ 'PRN'
PseudoRandomNumbers ≡ PseudoRandomNoise
⊗ betrifft diskrete Signale
⊗ obwohl PRN errechnet sind, entstehen perfekt gleichverteilte Zufallszahlenfolgen, die nicht von wirklich zufälligen Zahlenfolgen zu unterscheiden sind.
⊗ ein Bildungsgesetz ist nur "eingeweihten" bekannt und *(lange - bis zur ersten Wiederholung der Folge)* nicht erkennbar
⊗ Signalabschnitte wiederholen sich **lange nicht**; Maß: 'Periodizität'

23.2 zur Kryptografie

:

Pseudozufallszahlen werden zur *Dithering*- Steigerung von ADC-Auflösung und intensiv in der Kryptografie als Chiffrier-Key verwendet:



Die einzelnen Symbole der Nachricht *MSG* werden als Zahlen codiert.

Die Pseudozufallsfolge *Key* wird zur Nachricht *MSG* addiert, um die *MSG* zu chaotisieren.

Beim Empfänger (receiver) wird dieselbe Pseudozufallsfolge *Key* — und dazu muss dort das *Bildungsgesetz bekannt* sein — wieder subtrahiert. (in der booleschen Arithmetik sind *addieren* ⊕ und *subtrahieren* dasselbe)

23.3 Zufzi-Algorithmus

```
Primzahl p1, p2;
int      zufzi;

zufzi = Anfangswert not null;
for i=1 to N {
  zufzi = ((zufzi*p1) % p2);
  print (zufzi);
}
```

Programmieren Sie das bitte in 'C'!

23.4 Monte-Carlo Simulation

gleichverteilte pseudo-Zufallszahlen	⟶	normal-, poisson-, χ^2-, ... verteilte pseudo-Zufallszahlen

Die *Verteilungsfunktion* (das Integral der Wahrscheinlichkeitsdichte-Funktion) wird *'in umgekehrter Richtung' (Umkehrfunktion)* verwendet, um aus gleichverteilten Pseudozufallszahlen die gewünschte Verteilung zu erzielen:

normal-verteilte ZufZi	=	$(\int Normalverteilung(x)dx)^{-1}\ (y : gleichverteilt)$

24 Signal und Spektrum

Ein 'Signal' ist eine
Funktion der Zeit $f(t, \ldots);$

Ein 'Spektrum' ist eine
Funktion der Frequenz $f(\omega, \ldots);$

24.0.1 Aliasing JScript Simulation

link → http://10.10.63.61/SMUE/shr/js/DSP/KSN15/Aliasing/jsAliasXhQbR.html
http://www.qsl.net/oe7csj/zuig/jsAliasXhQbR.html

24.0.2 Sin-Summen = Fourier^{-}1

link → http://10.10.63.61/SMUE/shr/js/DSP/KSN15/SinSum/jsSinSum2XhPeP.html
http://www.qsl.net/oe7csj/zuig/jsSinSum2XhPeP.html

24.0.3 FIR Filter Simulation

link → http://10.10.63.61/SMUE/shr/js/DSP/KSN15/FIR/jsFirXhQbR1.html
http://www.qsl.net/oe7csj/zuig/jsFirXhUeA1.html

24.1 Signal-Korrelation

Zwei normierte*, periodische* Signale f, g korrelieren bei einem *Korrelationskoeffizienten* von

+1	maximal identisch, d.h. f = g
±0	gar nicht (fremd, unabhängig, orthogonal)
-1	sind gegenteilig: f = -g
+5	f,g nicht normiert
usw.	...

Man errechnet eine
→*Ähnlichkeit*,
das ist dasselbe wie die Frage:
"Wie stark ist f in g enthalten?" (und umgekehrt), *"was haben f und g gemeinsam?"*

Korrelation	Wechselbeziehung, Entsprechung, Zusammenhang, Gemeinsamkeit aus lat. Relation (Beziehung), relativ(im Vergleich) Vorsilbe 'co-': zusammen, miteinander wie in Kooperation, Kommunion, koaxial
normiert	$\int_{t=0}^{T} f(t)^2 dt = 1$
periodisch in $[0, T]$	$f(t + n \cdot T) = f(t + (n + 1) \cdot T)$
Signal	f(t,...) zeitabhängige Funktion
Spektrum	f(ω,...) frequenzabhängige Funktion

24.2 Kreuz- und Autokorrelation

24.2.1 Begriff

aus Wikipedia, der freien Enzyklopädie

Die Autokorrelation (auch Kreuzautokorrelation) ist ein Begriff aus der Statistik und der Signalverarbeitung und beschreibt

die Korrelation einer Funktion oder eines Signals mit sich selbst zu einem früheren Zeitpunkt.

Korrelationsfunktionen werden für Folgen von Zufallsvariablen berechnet, die von der Zeit abhängen. Diese Funktionen geben an, wie viel Ähnlichkeit die um die Zeit verschobene Folge mit der ursprünglichen Folge hat. Da die unverschobene Folge mit sich selbst am ähnlichsten ist, hat die Autokorrelation für die unverschobene Folge den höchsten Wert. Wenn zwischen den Gliedern der Folge eine Beziehung besteht, die mehr als zufällig ist, hat auch die Korrelation der ursprünglichen Folge mit der verschobenen Folge in der Regel einen Wert, der signifikant von Null abweicht. Man sagt dann, die Glieder der Folge sind autokorreliert.
Das dient der Qualitätsbeurteilung von Zufall *bzw.* pseudo- Zufall *zB. f.d.* 'Knackbarkeit' *(richtig: Schwierigkeitsgrad der Kryptoanalyse) von Chiffren (*'Verschlüsselungen'*)*

24.2.2 Signal 'ausgraben'

Übung:
Satellitensignale (zB. GPS) sind so schwach, dass sie beim Empfang hoffnungslos im Rauschen 'versinken'. Als Regenerationsmethode wird abgewandelte *Autokorrelation* angewandt: Ein Empfangssignal-Ausschnitt mit der Dauer exakt gleich der Signalperiodendauer wird immer wieder exakt im Takt der Signalperiode empfangen und digitalisiert. Die erste Samplesequenz wir der Reihe nach mit den neuen gefaltet, sodass sich mit der Zeit das 'richtige' Signal verstärkt und die zufälligen Rauschspannungen gegenseitig abschwächen.

Der 'Aha!' Effekt stelle sich anhand folgender Programmierübung (C, Java, Javascript oder Python) ein:

Geg.: ein Sinus- Signal, überdeckt von starkem Rauschen.

→ Stelle dieses Signalgemisch als Array von 1024 Samplewerten dar
und erzeuge dieses als Summe aus
a) der sin() Funktion mit der Periodenlänge 1024
b) einer selbsterzeugten, gleichverteilten Zufallszahlenfolge, die 20 mal so *'laut'* ist, wie der sin() (achte darauf, dass der Zufallszahlenmittelwert=0 ist!)

Simuliere nun 500 solche Sample-Arrays und addiere jedes zum allerersten.

Ges.: Gib zum Schluss das erste und letzte Samplearray als 'ASCII Art' (-90 Grad gedreht) zum Vergleich am Bildschirm aus.

Lege alle 30 Min den Programmcode auf Deinem 'obelix' Account ins (vorher zu erzeugende) Unterverzeichnis `~/public_html/gibXH/Dic5/Autokorr/` zwecks Mitarbeitsbewertung ab.

24.3 Fourier -Reihe, -Transformation

`DFT` . . . discrete fourier transform (='digitalisiert')
`FFT` . . . fast fourier transform
(dasselbe, nur schneller durch Einsparung identischer Multiplikationen)

24.3.1 Experiment: Hören

Studier mal ganz vertieft, was Du grade alles hörst;

- Strassenlärm?
- Leute?
- Radio?
- Lüfter?
- Wind?
- Vogelgezwitscher?
- ein Summen von Netzteilen?
- eine Fliege?
- Uhrenticken?
- den eigenen Atem?
- den eigenen Herzschlag?
- Sausen im Ohr (Tinnitus)?
- Glucksen im Körper?
- Knirschen der Gelenke?

Man hört erstaunlich viele Sachen zugleich!
Wir zerlegen den Schall in bekannte Komponen-

ten, um sie getrennt zu *verarbeiten*, indem wir uns darauf konzentrieren: Eine Art Filterung.

Das Schallsignal ist dann die **Summe** aller *unterscheidbaren* Einzelgeräusche:

$$Schall(t) = \sum_{i=1}^{N} Geräusch_i(t) \qquad (1)$$

(Linearität vorausgesetzt)

24.3.2 Zusammensetzen und zerlegen

Dieses Zusammensetzten und zerlegen tun Menschen oft; zB. setzen wir einen Klang aus mehreren Tönen, Instrumenten oder Stimmen zu einem Akkord, Chor oder Klangkörper zusammen, und dann fragt man wiederum, wer da was spielt oder singt.

Von einem Essen wollen wir die Zutaten herausschmecken, oder wir erkennen, dass es nach Brot und Kaffee, Zigaretten, Klebstoff, Haarspray, Kuhmist usw. riecht

Das geht immer dann, wenn die Grundkomponenten *unterscheidbar* sind wie ein Schlagzeug und eine Gitarre;

Man kann jedoch kaum heraushören, ob 3 oder 5 Ventilatoren laufen, 4 oder 6 Schlagzeuge rasseln, 12 oder 15 Leute schwätzen.

Gewöhnlich muss man für so eine Erkennung längere Zeit, zB. einige Sekunden, zuhören - in 10ms erkennt man eine Stimme noch nicht. Man prüft also eine Zeit t lang, ob das gesuchte Signal im Klangverlauf enthalten ist.

Wie kann man dasselbe *rechnerisch* erreichen?

24.3.3 Alain Fouriers Partnersuche

Zusammensetzen klingt einfach: Summe rechnen.

Wie aber rechne ich eine

enthalten-sein – Stärke ('Enthaltsamkeit') aus?

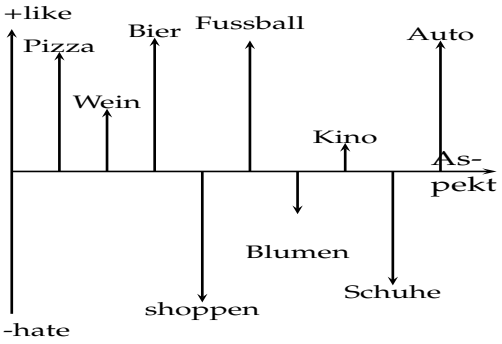
> *"Berechnen Sie die Enthaltsamkeit der Flöte in der kleinen Nachtmusik!"*

Wie ist wohl dieser *Alain Fourier*[5] auf seine Formel gekommen? Er war so Alain und sicherlich war ihm fad - youtube und facebook gabs ja nicht, kein SMS, kein chatroom; einsam lonely Boy; Enthaltsamkeit eben; könnte auf Partnersuche und als Mathematiker an der Berechnung der Enthaltsamkeit von Funktionen interessiert gewesen sein. Er fragt sich, welche Furie[6] passt am besten zu mir — in wessen Wesen ist mein Wesen am stärksten enthalten. Erste Idee

$$Enthaltsamkeit(a\ in\ b) = Enthaltsamkeit(b\ in\ a)$$

Kommutativ - es müsste also wurscht sein, wer wen sucht.

Jetzt kam der Typ wohl auf die Idee, ein 'mag / mag_nicht' Diagramm von sich anzufertigen:



Und jemand, der zu ihm passt, müsste ein *ähnliches* Diagramm haben. Aber wie rechnet man *ähnlich* aus? Wie stark ist mein Diagramm im anderen *enthalten*?

[5] Mathematiker 'Jean Baptiste Joseph Fourier', heisst der richtig! (21. März 1768 bis 16. Mai 1830)
[6] Rachegöttin in der römischen Mythologie

Die rettende Idee:

Gleichheit	→ Pluspunkte
Ungleichheit	→ Minuspunkte
unentschieden	→ keine Punkte

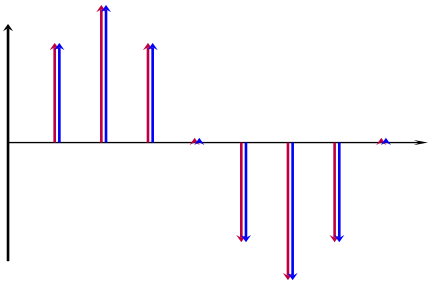
Welche mathematische Operation liefert

⊕ mit ⊕	→ +
⊖ mit ⊖	→ +
⊕ mit ⊖	→ -
⊖ mit ⊕	→ -
⊕ mit 0	→ 0
⊖ mit 0	→ 0
0 mit 0	→ 0

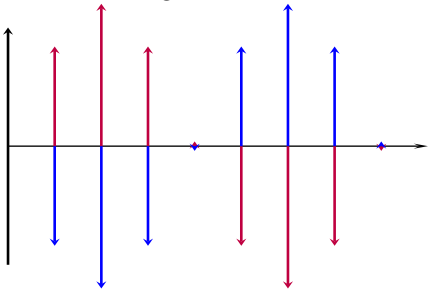
Es ist die Multiplikation.

Und was machtma mit den Punkten? - Ma zählt sie zamm!
→ Punktesumme

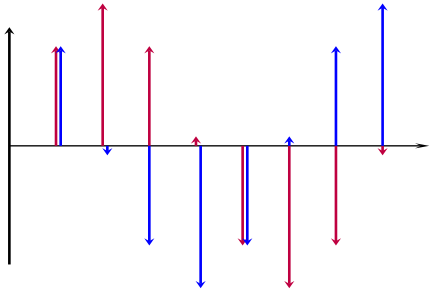
Die zwei passen maximal zamm: +100%



Aber die zwei gar nicht: -100%



Und ba de zwoa isch indifferent: ±0.0%



Hasch checkt, wasider hergmalen hun?

a) 2 mal sin(x)
b) sin(x) und -sin(x)
c) sin(x) und cos(x)

Wemmr also vu 2 Signale f(t) und g(t) viele paarweise zeitgleiche Samples nemmen und jeweils multiplizieren, dann isch die **Gesamtsumme**

$$f(0)g(0) + f(1)g(1) + f(2)g(2) + \ldots$$

(ma schreib kürzer:)

$$\frac{1}{N}\sum_{i=0}^{N} f(i)g(i)$$

(bei N *Samples*)
ein Maß für die "*Zusammenpassung*",
eine Art *"Durchschnittsmenge"*.

Bei kontinuierlichen Analogsignalen hatma ka einzelne Samples, sondern 'unendlich viele'; so wird aus der Summe → a Integral:

$$\frac{1}{T}\int_{\tau=0}^{T} f(\tau)g(\tau)dt$$

Des nenntma

Faltungs-Integral[7]
(convolution)

(Wahrheit: Die Faltung "*" ist eine Funktion:

$$f(t) * g(t) = \int_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Diese **Faltungsoperation** berechnet,[8]
wie gut zwei allgemeine Funktionen
(zB. f() und g()) zammpassen.

Wenn man *'gesampelte'* Funktionen f und g als vieldimensionale Vektoren auffasst, entspricht f*g dem `Skalarprodukt`, welches ja feststellt, wie sehr sich f und g gegenseitig *'enthalten'* - s.u.

Somit hat der französische Sack[9] aussagfunden, wie er das enthalten-Sein vu an beliebigen

[7] das ist Lüge, iXH weiss . . . . . richtig: $f(\tau) * g(t-\tau)$
[8] auch gelogen . . .
[9] und wieder gelogen: Der EULER war der Heuler; Laplaz haz nur als seins verkauft

f(t)=sin(ωt) oder f(t)=cos(ωt) in seine versauten Signale ausrechnen kann (Fourier is only f. periodic 0..2π Signals):

$$sin.Anteil(\omega) = \int_{t=0}^{2\pi} f(t)sin(\omega t)dt$$

$$cos.Anteil(\omega) = \int_{t=0}^{2\pi} f(t)cos(\omega t)dt$$

Das ist die
Sinus-/ Cosinustransformation
(Sonderfall der Fouriertransformation)

→

Die Fourieranalyse berechnet,
welche **sin(t)** und **cos(t)**
wie gut/stark
zu einem Signal f(t) passen (enthalten sind).

Klarerweise dürfen die Komponenten, in welche man das Signal zerlegt, sich nicht gegenseitig enthalten. (Wenn ich zB die 'Zutaten' Nüsse, Salz und Salznüsse zum Kochen verwende, ist nachher nicht mehr erkennbar, ob extra Salz zugegeben wurde, oder das Salz von den Nüssen stammt).

Man prüft das, indem man die Sinus-Cosinus-Zerlegung mit den Komponenten untereinander versucht; der Anteil muss dann jeweils '0' sein, zB:

$$0 = Anteil(\omega) = \int_{t=0}^{2\pi} sin(2\omega t)sin(\omega t)dt$$

$$0 = Anteil(\omega) = \int_{t=0}^{2\pi} sin(\omega t)cos(\omega t)dt$$

Beim sin()/cos() System ist das der Fall.

Wennma des mit digitalisierten Samples macht, nennt es sich DFT ... *Discrete Fourier Transform*. Die laufzeitoptimierte Version davon ist die FFT *Fast Fourier Transform*

24.3.4 Vektoren Skalarprodukt

Wenn wir zwei Funktionen a(t) und b(t) Zeit-digitalisieren (*'sampeln'*), können wir sie auch als vieldimensionale Vektoren auffassen:

$a = (a_1, a_2, a_3, a_4, \ldots)$
$b = (b_1, b_2, b_3, b_4, \ldots)$

$a \cdot b$ ist dann das `Skalarprodukt` und bedeutet als Wert die Länge der Projektion von a auf b bzw. b auf a, eine Art *'Ähnlichkeitsmaß'*.

Was ist, wenn ich digitalisierte Filterung machen will

das folgende JavaScript-Programmele macht s'genaue Gegenteil - zammsetzen:

es setzt ein Signal aus einzelnen
sin() und cos() Anteilen
zamm

Die Programmiersprache *Javascript*

- steht ina HTML Webseite
- wird vom Browser ausgeführt
  (muass 'Javascript' aktiviert sein)
- lafft also auf alle Plattformen
- braucht kan Compiler
- kann nit auf Rechnerhardware zugreifen (auch ka Files)
- leider mehrere verbreitete Versionen
- HTML5 kann Pixelgrafik (dieses PGM nimmtse nid)

Listing 56: jsSinSum2

```
<html>
<!--README:
        folgende HTMLfile is mitna Javascript (s.u.);
        es zeigt an Funktionsverlauf vu
        Summe[i=1..7] (sin(N.w.t)+cos(N.w.t))
        mit
        Buttons fuer Squarewave, Triangle und Sawtooth

        also pse "javascript" einschalten.

        LGX@PeP8k
 -->
<body bgcolor=wheat ondblclick="malen('graf')">
  Fourier-Reihen: Summe von sin()+cos() Funktionen
  <div id=graf style="position:relative; width:629px; height:200px; background-color:#
      F0F0FF;
        border:1px black solid; ">
        <div style="height:95px"></div><hr>
  </div>
  <form name="F" action="" style="font:500 12pt monospace;">
   f(t) =
```

```
   <input type=text   value=1.000       name=A1sin      size=5></input>*
   <input type=button value="sin(1&omega;t)" edisabled onClick=""></input>+
   <input type=text   value=1.000       name=A1cos      size=5></input>*
   <input type=button value="cos(1&omega;t)" disabled onClick=""></input>
   <input type=button value="sinus" onClick="sine()"></input>
   <br>       
   <input type=text   value=1.000       name=A2sin      size=5></input>*
   <input type=button value="sin(2wt)" disabled onClick=""></input>+
   <input type=text   value=1.000       name=A2cos      size=5></input>*
   <input type=button value="cos(2wt)" disabled onClick=""></input>
   <input type=button value="cosinus" onClick="cosine()"></input>
   <br>       
   <input type=text   value=1.000       name=A3sin      size=5></input>*
   <input type=button value="sin(3wt)" disabled onClick=""></input>+
   <input type=text   value=1.000       name=A3cos      size=5></input>*
   <input type=button value="cos(3wt)" disabled onClick=""></input>
   <input type=button value="sinus+cosinus" onClick="sincos()"></input>
   <br>       
   <input type=text   value=1.000       name=A4sin      size=5></input>*
   <input type=button value="sin(4wt)" disabled onClick=""></input>+
   <input type=text   value=1.000       name=A4cos      size=5></input>*
   <input type=button value="cos(4wt)" disabled onClick=""></input>
   <input type=button value="square" onClick="sqwv()"></input>
   <br>       
   <input type=text   value=1.000       name=A5sin      size=5></input>*
   <input type=button value="sin(5wt)" disabled onClick=""></input>+
   <input type=text   value=1.000       name=A5cos      size=5></input>*
   <input type=button value="cos(5wt)" disabled onClick=""></input>
   <input type=button value="triangle" onClick="tri()"></input>
   <br>       
   <input type=text   value=1.000       name=A6sin      size=5></input>*
   <input type=button value="sin(6wt)" disabled onClick=""></input>+
   <input type=text   value=1.000       name=A6cos      size=5></input>*
   <input type=button value="cos(6wt)" disabled onClick=""></input>
   <input type=button value="sawtooth" onClick="sawth()"></input>
   <br>       
   <input type=text   value=1.000       name=A7sin      size=5></input>*
   <input type=button value="sin(7wt)" disabled onClick=""></input>+
   <input type=text   value=1.000       name=A7cos      size=5></input>*
   <input type=button value="cos(7wt)" disabled onClick=""></input>
   <input type=button value="custom draw" onClick="drawgraphs()"></input>
   <hr>
  </form>
</body>

<head>
<script language=javascript>
var     divId   ='graf';        //name of se Anzeige-Rechteck
var     PointSzX=2,             //Breite van Kurven-Pixel
        PointSzY=5;             //Hoehe van Kurven-Pixel (damiz besser siegsch)
var     Farb = new Array(  "#FF8080", "#80FF80", "#8080FF", "#2200FF",
                "#FF2200", "#00FF22", "#600000", "#006000", "#000060" );

function sine(){
  allNull();
  document.F.A1sin.value=1.0;
  drawgraphs();
}
function cosine(){
  allNull();
  document.F.A1cos.value=1.0;
  drawgraphs();
}
function sincos(){
  allNull();
  document.F.A1sin.value=1.0;   document.F.A1cos.value=1.0;
  drawgraphs();
}
function sqwv(){
  allNull();
  document.F.A1sin.value=1.0;   document.F.A3sin.value=0.333;
  document.F.A5sin.value=0.200; document.F.A7sin.value=0.143;
  drawgraphs();
```

```
}
function tri(){
  allNull();
  document.F.A1sin.value=1.0;   document.F.A3sin.value=-0.111;
  document.F.A5sin.value=0.040; document.F.A7sin.value=-0.020;
  drawgraphs();
}
function sawth(){
  allNull();
  document.F.A1sin.value=-1.0;  document.F.A2sin.value=-0.5;
  document.F.A3sin.value=-0.33; document.F.A4sin.value=-0.25;
  document.F.A5sin.value=-0.20; document.F.A6sin.value=-0.167;
  document.F.A7sin.value=-0.143;
  drawgraphs();
}

function drawgraphs(){
  var   i, y, w, e, elId,
        divRef=document.getElementById(divId),
        PointSzPx=PointSzX+'px',
        nSteps;
  w  = 3.0 * 0.01*PointSzX;     // w... omega/2Pi; 3 Wellen
  y0 = parseInt(divRef.style.height)/2;
  yf = -y0/2;
  nSteps = 628.0/PointSzX;
  for(i=0;i<nSteps;i++){
    y = y0 + yf*(               //oissi zamzojn:
       parseFloat(document.F.A1sin.value)*Math.sin(w*i)
          + parseFloat(document.F.A1cos.value)*Math.cos(w*i)
      +parseFloat(document.F.A2sin.value)*Math.sin(2.0*w*i)
          + parseFloat(document.F.A2cos.value)*Math.cos(2.0*w*i)
      +parseFloat(document.F.A3sin.value)*Math.sin(3.0*w*i)
          + parseFloat(document.F.A3cos.value)*Math.cos(3.0*w*i)
      +parseFloat(document.F.A4sin.value)*Math.sin(4.0*w*i)
          + parseFloat(document.F.A4cos.value)*Math.cos(4.0*w*i)
      +parseFloat(document.F.A5sin.value)*Math.sin(5.0*w*i)
          + parseFloat(document.F.A5cos.value)*Math.cos(5.0*w*i)
      +parseFloat(document.F.A6sin.value)*Math.sin(6.0*w*i)
          + parseFloat(document.F.A6cos.value)*Math.cos(6.0*w*i)
      +parseFloat(document.F.A7sin.value)*Math.sin(7.0*w*i)
          + parseFloat(document.F.A7cos.value)*Math.cos(7.0*w*i)
      );
    y = Math.max(0.0, Math.min(y, parseInt(divRef.style.height))); //Bild.Begrenzg.
    elId = "Pix:" + i;
    if(! (e=document.getElementById(elId))){ //if nit existiert
        e = document.createElement('div');
        divRef.appendChild(e);          //em div unhaengen, synsch zoaxes nid
        e.id            = elId;         //damizes wiederfindt
    }
    e.style.position    = 'absolute';
    e.style.top         = y + 'px';
    e.style.left        = (i*PointSzX) + 'px';
    e.style.width       = PointSzPx;
    e.style.height      = PointSzY+'px';
    e.style.background  = Farb[7];
    e.style.bgcolor     = Farb[2];
    e.style.color       = Farb[3];
  }
}


function allNull(){
  document.F.A1sin.value=0.0;   document.F.A1cos.value=0.0;
  document.F.A2sin.value=0.0;   document.F.A2cos.value=0.0;
  document.F.A3sin.value=0.0;   document.F.A3cos.value=0.0;
  document.F.A4sin.value=0.0;   document.F.A4cos.value=0.0;
  document.F.A5sin.value=0.0;   document.F.A5cos.value=0.0;
  document.F.A6sin.value=0.0;   document.F.A6cos.value=0.0;
  document.F.A7sin.value=0.0;   document.F.A7cos.value=0.0;
}

</script>
</head>
```

```
</html>
```

24.4 Windowing Fensterfunktion

Da DFT/FFT die Länge des "Schieberegisters" zugleich als PERIODENDAUER verwendet, – auch wenn ein Signal mit anderer Periode drinsteht, entstehen in so einem Falle starke FALSCHE Oberwellen.

→ was tut man dagegen:

Man schwächt den Anfang und das Ende des Schieberegister-Inhaltes 'sanft' ab.

Eine Fensterfunktion legt in der digitalen Signalverarbeitung fest, mit welcher Gewichtung die bei der Abtastung eines Signals gewonnenen Abtastwerte innerhalb eines Ausschnittes (Fenster) in nachfolgende Berechnungen eingehen. Fensterfunktionen kommen bei der Frequenzanalyse (z. B. mittels diskreter Fouriertransformation), beim Filterdesign, beim Beamforming und anderen Signalverarbeitungsanwendungen zum Einsatz.

Ein andauerndes Signal wird in der Regel in Blöcken verarbeitet. Da Blocklängen in der Praxis endlich sind, kommt es zum sogenannten Leck-Effekt (englisch leakage effect), wenn die Blocklänge nicht gerade ein natürlichzahliges Vielfaches der Periode des Signals ist. Das errechnete Frequenzspektrum wird zu breit, es ist bildlich gesprochen "verschmiert". Dieser Effekt resultiert aus den Eigenschaften der Fourier-Transformation.

Durch die Verwendung einer geeigneten Fensterfunktion lässt sich der Effekt vermindern, aber nicht ganz vermeiden. Das Signal wird hierbei meistens am Fensterbeginn "eingeblendet" und am Fensterende "ausgeblendet", was zu einer künstlichen Periodisierung des Signals innerhalb der Zeitfensterlänge führt.

Die Fensterfunktion beeinflusst neben der spektralen Verbreiterung außerdem die Frequenzselektivität und den maximal möglichen spektralen Fehler. Es gibt verschiedene Fensterfunktionen unterschiedlicher Komplexität. Die Auswahl einer passenden Fensterfunktion ist daher stets ein Kompromiss, der den speziellen Anforderungen des jeweiligen Anwendungsfalls Rechnung trägt.

Filterdesign
Eine häufig angewandte Methode für das Design von digitalen Filtern mit endlicher Impulsantwort (FIR-Filter) ist die Fenstermethode (engl. window method). Dabei wird der gewünschte Frequenzgang des Filters definiert und mit der inversen Fouriertransformation die ideale Impulsantwort ermittelt.

Im Filterdesign führen breite (selektive) Fensterfunktionen zu steilen Übergängen zwischen Durchlass- und Sperrbereich, aber zu geringer Sperrdämpfung. Schmale (nicht selektive) Fensterfunktionen führen zu flachen Übergängen zwischen Durchlass- und Sperrbereich, dafür aber zu großer Sperrdämpfung.

Beispiele von Fensterfunktionen:

- **Rechteck-Fenster** (Rechteck-Fensterfunktion): wirkungslos
  Die Rechteck-Fensterfunktion, auch bezeichnet als Dirichlet-Fenster (nach Peter Gustav Lejeune Dirichlet), ist im gesamten Fensterbereich 1 und außerhalb 0. Die einfache Verarbeitung des Eingangssignals in Blöcken entspricht der Anwendung dieser Fensterfunktion. Das Betragsspektrum entspricht dem Betragsverlauf der si-Funktion. Nur im Sonderfall, wenn die Fensterbreite exakt ein ganzzahliges Vielfaches der Periodendauer der harmonischen Signalschwingung umfasst, tritt bei zeitdiskreten Signalen zufolge der Fensterung mit dem Rechteck-Fenster kein Leck-Effekt auf.
- Trapez-Window: lineare Abschwaechung am Rand des Schieberegisters

  verschieden ägerundete"Gewichtungen:

- **Von-Hann-Fenster** (Hann-Fensterfunktion, 'Hanning-Window')
  Das Von-Hann-Fenster basiert auf einer Überlagerung von drei spektral gegeneinander verschobenen si-Funktionen um gegenüber dem Rechteck-Fenster mit nur einer si-Funktion im Spektrum eine stärkere Unterdrückung der Nebenkeulen zu erreichen. Der Nachteil ist eine Reduktion in der Frequenzauflösung. Das Von-Hann-Fenster mit der Haversine-Funktion wird auch als Raised-Cosinus-Fenster bezeichnet. [...] Die Bezeichnung Hann-Fenster stammt aus der Publikation "Particular Pairs of Windows" von R. B. Blackman und John W. Tukey, die dieses nach Julius von Hann benannt haben. Aus diesem Artikel stammt auch die weit verbreitete Bezeichnung Hanning-Fenster, wobei dort jedoch lediglich die Anwendung des Hann-Fensters als "hanning" (abgeleitet

von "to hann") bezeichnet wird.

- Hamming-Fenster
  Diese Fensterfunktion ist benannt nach Richard Hamming und stellt eine Abwandlung des Von-Hann-Fensters dar. Bei dem Hamming-Fenster werden die Koeffizienten so gewählt, dass die Nullstellen der beiden benachbarten und größten Nebenkeulen maximal unterdrückt werden. Dies entspricht einer unterschiedlichen Gewichtung der einzelnen si-Funktionen im Spektrum der Fensterfunktion.
- Blackman-Fenster (3-Term)
- Blackman-Harris-Fenster (Abwandlung der Blackman-Fensterfunktion)
- Blackman-Nuttall-Fenster
  Das Blackman-Nuttall-Fenster ist bis auf die vier fast identischen Koeffizienten identisch mit dem Blackman-Harris-Fenster, was den Einfluss der notwendigen Genauigkeit bei der Implementierung der Koeffizienten bei dieser Klasse von Fensterfunktionen verdeutlicht.
- Flat-Top-Fenster
  Das Flat-Top-Fenster ist eine teilweise negativ bewertende Fensterfunktion, welche unter anderem in Spektrumanalysatoren für die Messung und Bewertung des Betrags von einzelnen Amplituden eingesetzt wird. Das Flat-Top-Fenster weist einen vergleichsweise kleinen Amplitudenfehler auf, nachteilig ist die schlechte Frequenzauflösung.
- Bartlett-Fenster
- Dreieck-Fensterfunktion
  Eine eng verwandte Variation der Bartlett-Fensterfunktion basiert auf der Dreiecksfunktion und weist als Unterschied an den Anfangs- bzw. Endwerten keine Nullwerte auf.
- Bartlett-Hann-Fenster
  Dies ist eine Kombination der Dreiecksfunktion des Bartlett-Funktion mit der Hann-Fensterfunktion
- Kosinus-Fenster
  Die Kosinus-Fensterfunktion ist auch als Sinus-Fensterfunktion bekannt
- Tukey-Fenster
- Lanczos-Fenster
- Kaiser-Fenster
- Gauß-Fenster
  Das Gauß-Fenster basiert auf der Gaußschen Glockenkurve
- Ultraspherical-Fenster
- Dolph-Chebyshev
- Saramäki
- Slepian (DPSS)
- Poisson

(aus https://de.wikipedia.org/wiki/Fensterfunktion 09Sep'22)

24.4.1 Fourier transform

Fourier...
TBD.

24.4.2 'Filtern' ganz allgemein

Ein *Filter* ist ein input/output- System, eine *Blackbox*.

$u_e(t)$ *in* → **Filter f** → $u_a(t)$ *out*

Als Beschreibung *im Zeitbereich* gibt man

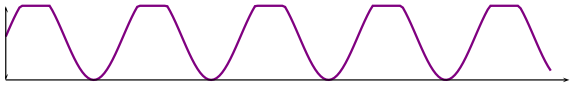
$u_a(t) = f(u_e(t))$

an. Bei *Verstärkern* geht das gut:

$u_a(t) = A_V * u_e(t)$

A_V ... Amplification of Voltage

zumindest, solange sie 'linear' sind. Ein `'Filter'` sollte aber etwas vom Input weg-`*filtern*`, ein `'Filter'` sollte `*entfernen*`. Ein *'Begrenzer'* (ob gewollt oder nicht) setzt dem Output ein *'Limit'* (engl. 'limiter'), er *'filtert'* Spitzen weg.



Das wäre als Zeit-Funktion etwa

$$u_a(t) = \begin{cases} u_e(t) & if < 1.7 \\ 1.7 & else \end{cases}$$

Klingen tuts schrecklich — alles verzerrt! Allerdings gibt es Leute, die dafür zahlen, wenn mans ihnen an die Elektrogeige klemmt, so Carlos, Erics, Jimmies, Jeffs udgl.

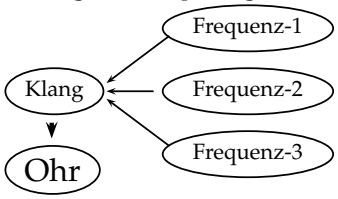
Ein RC-Tiefpass wird schon lästiger

$$u_a(t) = \frac{1}{C}\int_{\tau=0}^{t} \frac{u_e(\tau)-u_C(\tau)}{R}\,\frac{1}{C}\,d\tau$$

24.4.3 Beschreibung im Frequenzbereich

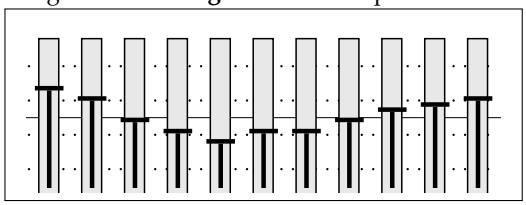
Nun ist es ja so, dass das menschliche (Ohr) die Signale als `Tonhöhen` wahrnimmt. Wir beschreiben einen Schall als *'hell'* oder *'tief'* und hören *'einen Dreiklang'* oder *'einen Bass und eine Pfeife'*.

Seit der Entdeckung der `Frequenz` wissen wir, dass *'Tonhöhe'* dasselbe ist, und wir beschreiben einen `Klang` als `Frequenzgemisch`.



Die Zusammensetzung aus Frequenzen sind wir gut gewöhnt

- Besonders Tonstudios und Großkonzerte verwenden *'Equalizer*: Ein Satz Schieberegler, deren Anordnung ein **Bode-Diagramm** nachempfindet.



- Auf besseren / älteren *'Elektronenorgeln* (heute *Keyboard*) findet man Klangeinstellregister mit *'Zugriegeln'*, an denen man den Klang aus *Natur-Obertonreihen* zusammenstellen kann. (Hammond) Bei richtigen Pfeifenorgeln erreicht man Vergleichbares mit Regulierung der Luftströme zu den Pfeifengruppen (*'Register'*; davon kommt der Spruch *'alle Register ziehen'*).

24.4.4 Filter-Grenzfrequenz

- Audiotechnik, RC-Filter: **-3dB**
  Frequenz, bei der die -3dB *'Grenze'* gegenüber dem Maximum erstmals (bleibend?) unterschritten wird.
- (schmalbandige) HF-Filter: **-6dB**
  Frequenz, bei der die **-6dB** *'Grenze'* gegenüber dem Maximum erstmals (bleibend?) unterschritten wird. (Grund: die Welligkeit (passband ripple) schmalbandiger HF-Filter ist oft > 3dB)
- Bandfilter haben eine -3dB (bzw. -6dB) `Bandbreite`. Bei Schmalbandfiltern wird oft auch die -60dB Bandbreite angegeben, sowie ein *'shape factor'*::= $\frac{-60dBBandbreite}{-6dBBandbreite}$.

24.4.5 Filter-Ordnung

Definition-1
beschreibt den Anstieg/Abfall im Übergangsbereich (transition region/band)

$$\text{Ordnung} = \frac{\frac{Anstieg}{Dekade}[dB]}{20[dB]} + \frac{\frac{Abfall}{Dekade}[dB]}{20[dB]}$$

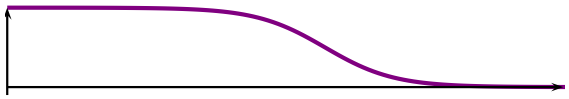
(aus https://de.wikipedia.org/wiki/Filter_(Elektrotechnik) 17.Mar16)

Ordnung
*Die Ordnung eines Filters beschreibt die Verstärkungsabnahme (Dämpfung und Flankensteilheit) von Frequenzen (weit) oberhalb oder unterhalb der jeweiligen Grenzfrequenz des Filters. Sie ist bei Tiefpass- oder Hochpassfilter über der Frequenz etwa n*6 dB pro Oktave (n*20 dB pro Dekade), wobei n die Ordnung des Filters darstellt. Für Bandpässe bzw. Bandsperren, welche Kombinationen aus Tiefpass- und Hochpassfiltern darstellen und somit zwei Filterflanken aufweisen, ist die Filterordnung als Funktion der Steilheit der Filterflanke doppelt so hoch: Ein Bandpass 4. Ordnung weist mitunter 40 dB pro Dekade auf.*
Filter höherer Ordnung können entweder wirklich erstellt oder durch Hintereinanderschaltung von Filtern niedriger Ordnung (1. und 2. Ordnung) realisiert werden.
(aus https://de.wikipedia.org/wiki/Filter_(Elektrotechnik) 17Mar16)

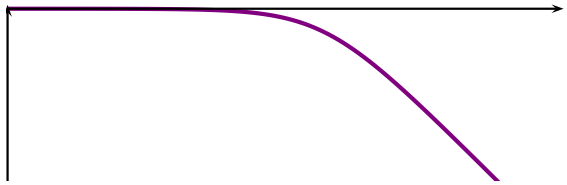
Definition-2
Die Filterordnung ist der Grad des *'Charakteristischen Polynoms'* (=Nennerpolynom) der Übertragungsfunktion H(s)

einfacher RC-Tiefpass

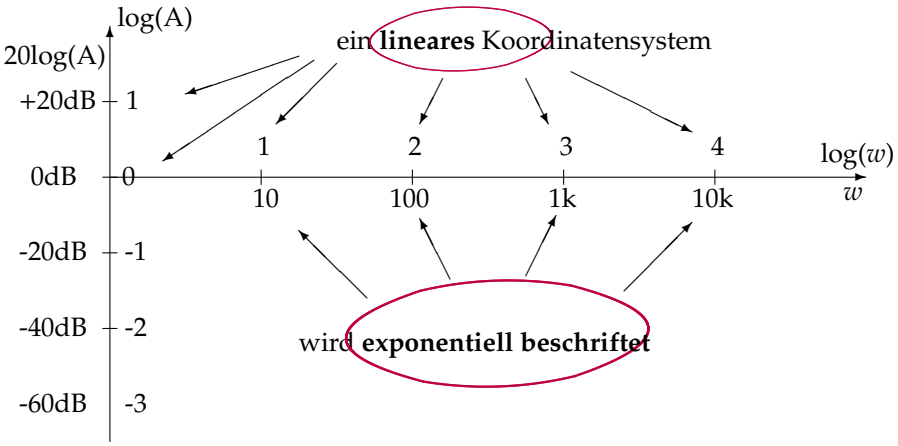
R=1k, C=100nF (RC=1E-4, w=1E4, f=1591.549Hz),
linear:

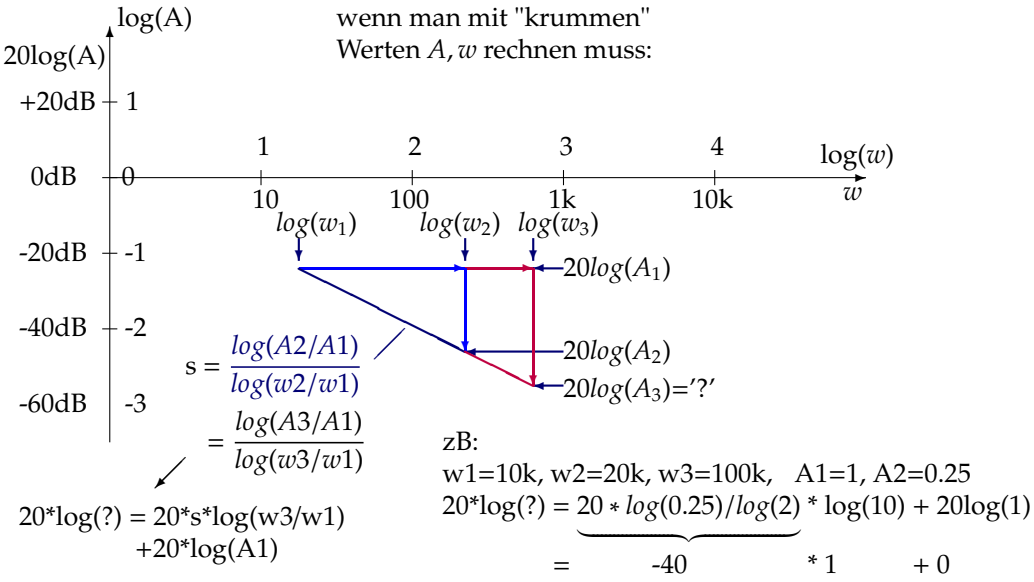


logarithmisch:



24.4.6 rechnen im Bode-Diagramm





24.4.7 Filtercharakteristiken

- Filter mit kritischer Dämpfung
  - rückwirkungsfreie(!) Reihenschaltung passiver RC-TP/HP
  - kein Überschwingen der Sprungantwort.
  - Char.Polynom besitzt nur reele Nullstellen
  - "schlechter Amplitudenfrequenzgang" ::= wenig Flankensteilheit
  - steile Filter brauchen hohe Filterordnung

- Bessel-Filter:
  - keine Welligkeit
  - bester Phasenfrequenzgang (konstante Gruppenlaufzeit) im Passband, → geringste Verzerrungen / bestes Impulsübertragungsverhalten.
  - Übergangsbereich nicht so 'scharf' wie Butterworth
  - $\tau_{gr} = const. \iff \varphi(\omega) \propto \omega$

    φ ... Phasenverschiebung, phase lag
    ω ... Frequenz (Kreisfrequenz)
    τ_{gr} ... Gruppenlaufzeit
    $\propto$ ... proportional
  - Flankensteilheit nur mit hoher Filterordnung → nur Spezialfälle.

- Gauß Filter:
  - [...] Sprungantwort keine Überschwingung
  - [...] maximale Flankensteilheit im Übergangsbereich aufweisen. Als Besonderheit besitzt bei diesem Filter sowohl
  - die Übertragungsfunktion als auch
  - die Impulsantwort den Verlauf einer gaußschen Glockenkurve [...] wovon sich auch der Name dieses Filtertyps ableitet. Anwendungsbereiche [...]
  - digitale Modulationsverfahren und [...] (Gauß-Filter besitzen eine konstante Gruppenlaufzeit im Sperr- und Durchlassbereich und kein Überschwingen in der Sprungantwort. Einsatzbereich dieses Filters liegt primär zur Impulsformung mit Anwendungsbereichen in der digitalen Signalverarbeitung. Die Impulsformung findet [Anwendung] bei digitalen Modulationsverfahren wie dem Gaussian Minimum Shift Keying (GMSK), da damit die einzelnen, meist rechteckförmigen Sendesymbole in Impulse der gaußschen Glockenkurve mit geringerem Bandbreitenbedarf als die ursprünglichen rechteckförmigen Sendesymbole umgewandelt werden können. Damit ist eine höhere spektrale Effizienz des Modulationsverfahrens verbunden.)
  - Bildverarbeitung (In der Bildverarbeitung werden Gauß-Filter zur Glättung oder Weichzeichnen des Bildinhaltes verwendet. Es kann damit das Bildrauschen vermindert werden: Kleinere Strukturen gehen verloren, gröbere Strukturen bleiben)

  ('*aus Wikipedia, der freien Enzyklopädie*' de.wikipedia.org/wiki/Gauss-Filter 21Mar16)

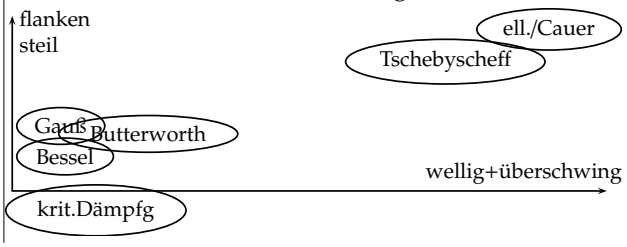
- Butterworth Charakteristik
  - best-horizontaler Amplitudenfrequenzgang
  - keine Welligkeit im Passband,
  - schärfster Knick vor der Grenzfrequenz + maximal steil
  - nicht konstante Gruppenlaufzeit
  - leichtes Überschwingen
  - gut dimensionier- u. realisierbar
    → verbreitet

- Tschebyscheff:
  - Amplituden-Welligkeit nur im Passband (passband ripple)
  - steilster Übergang (steepness)
  - Welligkeit $\overset{\propto}{\leftrightarrow}$ Flankensteilheit
  - mit Welligkeiten 0.1, 0.2, 0.5, 1, 2, 3dB tabelliert
  - gut dimensionier- u. realisierbar
    → verbreitet

- Tschebyscheff invers:
  - Welligkeit nur im Sperrbereich
  - Gruppenlaufzeit/Phasenverschiebung besser als Tschebyscheff
  - gut dimensionierbar
  - weniger gut realisierbar wegen Nullstellen

- Elliptische Filter (Cauer):
  - größte Flankensteilheit
  - Welligkeit im Durchlass- und Sperrbereich (spezifizierbar)
  - aufwendige Dimensionierung u. Realisierung → Synthesesoftware

- nicht-impedanz-angepasste Eingangs- oder Ausgangsbeschaltungen erhöhen iA. die Welligkeit (Eingangs- u. Ausgangs-'R' sind Bestandteil der Dimensionierung)



24.5 Impuls + Rechteck

24.5.1 Dirac - Impuls

Paul Adrien Maurice DIRAC *"[...] (* 8. August 1902 in Bristol; + 20. Oktober 1984 in Tallahassee) war ein britischer Physiker, Nobelpreisträger und Mitbegründer der Quantenphysik. Eine seiner wichtigsten Entdeckungen ist in der Dirac-Gleichung von 1928 beschrieben, in der Einsteins Spezielle Relativitätstheorie und die Quantenphysik erstmals zusammengebracht werden konnten. Ferner legte er die Grundlagen für den späteren Nachweis von Antimaterie. [...]"*
(Quelle https://de.wikipedia.org/wiki/Paul_Dirac 06Jan17)

"[...] Die Delta-Distribution (auch δ-Funktion; Dirac-Funktion, -Impuls, -Puls, -Stoß (nach Paul Dirac), Stoßfunktion, Nadelimpuls, Impulsfunktion oder Einheitsimpulsfunktion genannt) als mathematischer Begriff ist eine spezielle irreguläre Distribution mit kompaktem Träger. Sie hat in der Mathematik und Physik grundlegende Bedeutung. Ihr übliches Formelsymbol ist δ (kleines Delta)[...]"
(Quelle https://de.wikipedia.org/wiki/Delta-Distribution 06Jan17)

Der *'Dirac-Impuls'* ist ein theoretisches, praxisfremdes, aber nützliches Konstrukt für Berechnungen. Er hat die Eigenschaften

- auf der Y-Achse
- unendlich schmal
- unendlich hoch
- Fläche = 1

- das Integral $\int \delta(t) * dt$ ist die Sprungfunktion (engl. 'Heaviside step function'), die 'links der y-Achse' 0 und 'rechts' 1 ist.

- es gibt bereits viele Erkenntnisse rund um den Diracimpuls, wie etwa:

- die Laplacetransformierte $\mathcal{L}(s) = 1$ (enthält jede Frequenz gleich stark)
  (dass das einer unendlichen Leistung entspräche, ist kein Problem, da der Dirac-Impuls ja auch unendlich *hoch* sei)
- Digitalisierung/Sampling entspricht der Modulation
  mit einer Folge n von Dirac-Impulsen
  $\Delta_T(t) = \sum_{n=0}^{\infty} \delta(t - nT)$
  T ... sampling interval, n ... sampleNr.

Die (Dirac-)Impuls-Antwort h(t) eines Übertragungssystems ist die invers Laplace-Transformierte der Übertragungsfunktion H(s)
Wenn Übertragungssystem == digitalFilter
⇒ Impulsantwort == Filterkoeffizienten

Die Antwort eines Übertragungssystems auf die Sprungfunktion als Input heisst natürlich *"Sprung-Antwort"* und $\mathcal{L}(Sprung) = \frac{1}{s}$

Sei nun
f(t) ..... periodisches Signal mit Periode T
SR ..... SamplingRate, vielfaches von 1/T
$\mathcal{F}(j\omega)$ ..... Fouriertransformierte von f(t)
$\delta(t)$ ..... Diracimpuls (immer 0 ausser bei t=0)
n ..... $\epsilon\ \mathbb{N}_0$
und man weiss von J.Fourier, dass ein periodisches Signal f(t) ausschliesslich *harmonische* Oberwellen mit Frequenzen
n * 1/T
besitzt, d.h das Spektrum $|\mathcal{F}(j\omega)|$ ist nur
an Stellen n * $2\pi/T \neq 0$

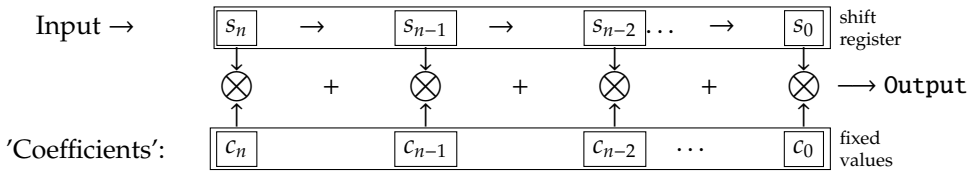
Was bedeutet das für die Samplerei?

25 FIR und IIR Filter

25.1 FIR - Finite Impulse Response

(1) Wir studieren Aliasing anhand der Javascript- programmierten Webseiten
http://www.qsl.net/oe7csj/zuig/jsAliasXhQbR.html
http://10.10.63.61/SMUE/shr/js/DSP/KSN15/Aliasing/jsAliasXhQbR.html

(2) Wir studieren noch einmal FIR Prinzip u. Verhalten anhand der Javascript- programmierten Webseiten
http://www.qsl.net/oe7csj/zuig/jsFirXhUeA1.html
http://10.10.63.61/SMUE/shr/js/DSP/KSN15/FIR/jsFirXhQbR1.html



Als Pseudo-C-Programmcode:

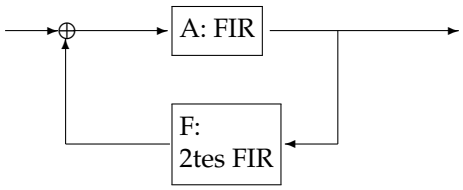
```
wiederhole:
  outputsum = 0;
  for(i=0 ; i<N ; i++ ){
    outputsum+=  sample[i] * k[i];
  }
  samples_rechtsshiften();
  goto wiederhole;
```

kleine Untersuchung:

```
Filterlaenge:2
Koef
fizienten: +1 -1
Input                           Output
 1  1  1  1  1  1  1  1         0  0  0  0  0  0  0  0
 1  0  1  0  1  0  1  0        +1 -1 +1 -1 +1 -1 +1
+1 -1 +1 -1 +1 -1 +1 -1        +2 -2 +2 -2 +2 -2 +2
 1  1  1  0  0  0  1  1         0  0 +1  0  0 -1  0
-->es ist ein 'Flankendetector' (Hochpassfilter)

Filterlaenge: 4
Koeffizienten: +1 -1 +1 -1
Input                           Output
 1  1  1  1  1  1  1  1  1  1    0   0   0   0   0   0   0
 1  0  1  0  1  0  1  0  1  0   +2  -2  +2  -2  +2  -2  +2
-->die Koeffizienten sind nicht normiert -- das Filter verstaerkt!
-->
Koeffizienten: +.5 -.5 +.5 -.5
Input                           Output
 1  1  1  1  1  1  1  1  1  1    0   0   0   0   0   0   0
 1  0  1  0  1  0  1  0  1  0   +1  -1  +1  -1  +1  -1  +1
 1  1  1  1  0  0  0  0  1  1    0 +.5   0 +.5   0 -.5   0
-->der Flankendetector erkennt 'Oberwellen'
```

25.2 IIR - Infinite Impulse Response

Ein IIR entsteht aus der Erweiterung eines FIR um eine Rückkopplung (feedback loop). Damit entsteht ein System nach dem Muster einer Regelung (control loop) und damit potentiell auch endlos abklingende sowie die Gefahr ansteigender Eigenschwingung (Stabilitätsproblematik):



Vor-/Nachteile: Weil der Input die Filterkette vielfach durchläuft, wirkt es wie ein viieeeel längeres FIR. So

- ist das Filter steilflankiger + bandbreitenschmäler
- zudem stabilitätsgefährdet und
- eigenschwingungsgefährdet,
- anfälliger für numerischen Überlauf
- sumieren sich Rundungsfehler
- ist genauere (FP-) Arithmetik erforderlich
- viel schwerer zu dimensionieren

25.3 akademisch:

Kapitel 19:
Digitale Filter

Im Kapitel 13 auf S. 815 haben wir eine Reihe von Möglichkeiten zur Realisierung verschiedener Übertragungsfunktionen mit Hilfe von aktiven Filtern kennen gelernt. Die verarbeiteten Signale waren Spannungen in Form kontinuierlicher Zeitfunktionen. Die verwendeten Bauelemente waren Widerstände, Kondensatoren und Verstärker.

In neuerer Zeit geht man mehr und mehr dazu über, die Signalverarbeitung nicht analog sondern digital durchzuführen. Die Vorteile liegen in der höheren Genauigkeit und Reproduzierbarkeit sowie in der geringen Störempfindlichkeit. Nachteilig ist der höhere Schaltungsaufwand, der jedoch angesichts des zunehmenden Integrationsgrades digitaler Schaltungen immer weniger ins Gewicht fällt.

Statt kontinuierlicher Größen werden diskrete Zahlenfolgen verarbeitet. Die Bauelemente sind Speicher und Rechenwerke. – Beim Übergang vom Analog- zum Digitalfilter stellen sich drei Fragen:

1) Wie lässt sich aus der kontinuierlichen Eingangsspannung eine Folge von diskreten Zahlenwerten gewinnen, ohne dabei Information zu verlieren?
2) Wie muss man diese Zahlenfolge verarbeiten, um die gewünschte Übertragungsfunktion zu erhalten?
3) Wie lassen sich die Ausgangswerte wieder in eine kontinuierliche Spannung zurückverwandeln?

Die Einbettung eines digitalen Filters in eine analoge Umgebung ist in Abb. 19.1 schematisch dargestellt. Das Abtast-Halte-Glied entnimmt aus dem Eingangssignal $U_e(t)$ in den Abtastaugenblicken t_μ die Spannungen $U_e(t_\mu)$ und hält sie jeweils für ein Abtastintervall konstant. Damit bei der Abtastung keine irreparablen Fehler entstehen, muss das Eingangssignal gemäß dem Abtasttheorem auf die halbe Abtastfrequenz bandbegrenzt sein. Daher ist meist ein Tiefpass am Eingang erforderlich.

Der Analog-Digital-Umsetzer wandelt die zeitdiskrete Spannungsfolge $U_e(t_\mu)$ in eine zeit- und wertdiskrete Zahlenfolge $x(t_\mu)$ um. Bei den Werten x handelt es sich üblicherweise um N-stellige Dualzahlen. Die Stellenzahl N bestimmt dabei die Größe des Quantisierungsrauschens (s. Gl. (18.12) von S. 1000). Abbildung 19.2 zeigt einige Beispiele für gebräuchliche Abtastfrequenzen und Auflösungen.

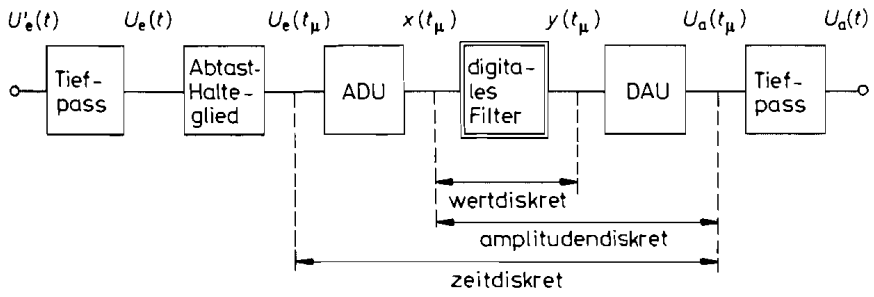


Abb. 19.1. Einsatz eines digitalen Filters in einer analogen Umgebung

(Tietze-Schenk)

Signal	Abtastfrequenz	Auflösung
Telefon-Sprache	8 kHz	12 bit
CD-Musik	44,1kHz	16 bit
Digitales Fernsehen	13,3MHz	8 bit

Abb. 19.2. Übliche Abtastfrequenzen und Wortbreiten für die digitale Signalverarbeitung

Das digitale Filter in Abb. 19.1 erzeugt die gefilterte Zahlenfolge $y(t_\mu)$. Um sie wieder in eine Spannung zu verwandeln, verwendet man einen Digital-Analog-Umsetzer. Er liefert an seinem Ausgang eine wert- und zeitdiskrete treppenförmige Spannung. Um sie in eine kontinuierliche Spannung umzuwandeln, muss man einen Tiefpass zur Glättung nachschalten.

19.1 Abtasttheorem

Ein kontinuierliches Eingangsignal lässt sich in eine Folge von diskreten Werten umwandeln, indem man mit Hilfe eines Abtast-Halte-Gliedes in äquidistanten Zeitpunkten $t_\mu = \mu T_a$ Proben aus dem Eingangssignal entnimmt. Dabei ist $f_a = 1/T_a$ die Abtastfrequenz. Man erkennt in Abb. 19.3, dass sich die entstehende Treppenfunktion umso weniger von dem kontinuierlichen Eingangssignal unterscheidet, je höher die Abtastfrequenz ist. Da aber der schaltungstechnische Aufwand stark mit der Abtastfrequenz wächst, ist man bemüht, sie so niedrig wie möglich zu halten. Die Frage ist nun: welches ist die niedrigste Abtastfrequenz, bei der sich das Originalsignal noch *fehlerfrei*, d.h. ohne Informationsverlust rekonstruieren lässt. Diese theoretische Grenze gibt das Abtasttheorem an, das wir im Folgenden erläutern wollen.

Zur mathematischen Beschreibung ist die Treppenfunktion in Abb. 19.3 nicht gut geeignet. Man ersetzt sie deshalb wie in Abb. 19.4 durch eine Folge von Dirac-Impulsen:

$$\tilde{U}_e(t) = \sum_{\mu=0}^{\infty} U_e(t_\mu) T_a \delta(t - t_\mu) \tag{19.1}$$

Ihre Impulsstärke $U_e(t_\mu)T_a$ ist dabei symbolisch durch einen Pfeil charakterisiert. Man darf sie nicht mit der Impulshöhe verwechseln; denn der Dirac-Impuls ist nach der Definition ein Impuls mit unendlicher Höhe und verschwindender Dauer, dessen Fläche jedoch einen endlichen Wert besitzt, den man als Impulsstärke bezeichnet. Diese Eigenschaft

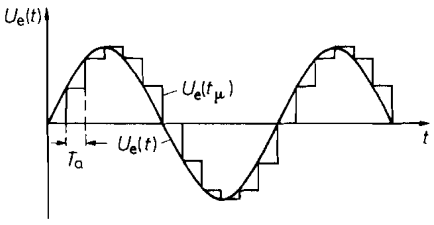


Abb. 19.3. Beispiel für das Eingangssignal $U_e(t)$ und die Abtastwerte $U_e(t_\mu)$

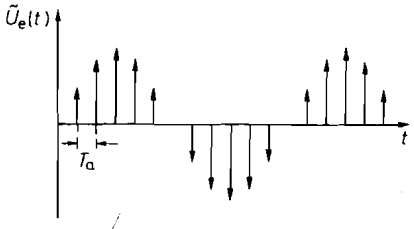


Abb. 19.4. Darstellung des Eingangssignals durch eine Impulsfolge

(Tietze-Schenk)

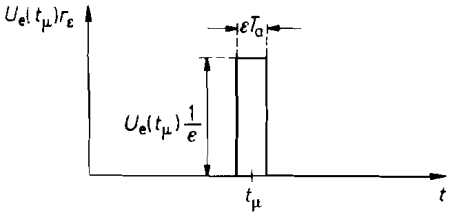


Abb. 19.5. Näherungsweise Darstellung eines Dirac-Impulses durch einen endlichen Spannungsimpuls

wird durch Abb. 19.5 verdeutlicht, in der der Dirac-Impuls näherungsweise durch einen Rechteckimpuls r_ε dargestellt ist. Dabei gilt der Grenzübergang:

$$U_e(t_\mu)T_a\delta(t-t_\mu) \;=\; \lim_{\varepsilon\to 0} U_e(t_\mu)r_\varepsilon(t-t_\mu) \tag{19.2}$$

Um zu untersuchen, welche Informationen die in Gl. (19.1) dargestellte Impulsfolge enthält, betrachten wir ihr Spektrum. Durch Anwendung der Fourier-Transformation auf Gl. (19.1) erhalten wir:

$$\tilde{X}(jf) \;=\; T_a \sum_{\mu=0}^{\infty} U_e(\mu T_a)e^{-2\pi j\mu f/f_a} \tag{19.3}$$

Man erkennt, dass dieses Spektrum eine periodische Funktion ist. Ihre Periode ist gleich der Abtastfrequenz f_a. Durch Fourier-Reihenentwicklung dieser periodischen Funktion lässt sich nun weiter zeigen, dass das Spektrum $|X(jf)|$ im Bereich $-\frac{1}{2}f_a \le f \le \frac{1}{2}f_a$ identisch ist mit dem Spektrum $|X(j,f)|$ der Originalfunktion [19.1]. Es enthält also noch die volle Information, obwohl nur wenige Werte aus der Eingangsfunktion entnommen werden.

Dabei ist lediglich eine Einschränkung zu machen, die wir anhand der Abb. 19.6 erläutern wollen: Das Originalspektrum erscheint nur dann unverändert, wenn die Abtastfrequenz mindestens so hoch gewählt wird, dass sich die periodisch wiederkehrende Spektren nicht überlappen. Das ist nach Abb. 19.6 für

$$\boxed{f_a > 2f_{\max}} \tag{19.4}$$

der Fall. Diese Bedingung wird als Abtasttheorem bezeichnet.

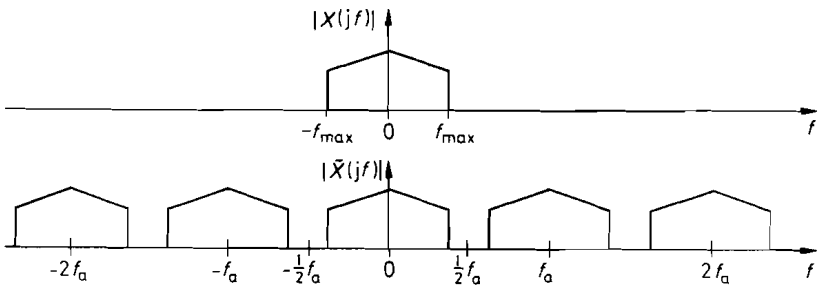


Abb. 19.6. Spektrum der Eingangsspannung vor dem Abtasten (oben), und nach dem Abtasten (unten)

(Tietze-Schenk)

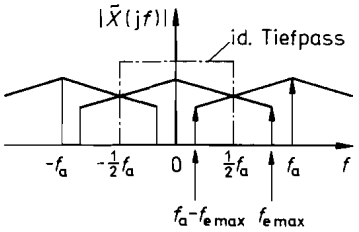


Abb. 19.7. Überlappung der Spektren bei zu niedriger Abtastfrequenz

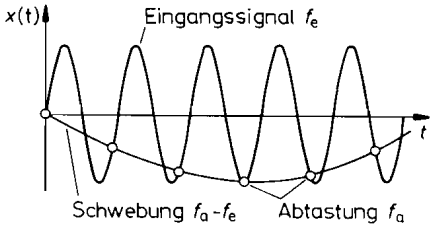


Abb. 19.8. Zustandekommen der Schwebung bei zu niedriger Abtastfrequenz für $f_e \lessapprox f_a$

Rückgewinnung des Analogsignals

Aus Abb. 19.6 lässt sich unmittelbar die Vorschrift für die Rückgewinnung des Analogsignals ablesen: Man braucht lediglich mit Hilfe eines Tiefpassfilters die Spektralanteile oberhalb $\frac{1}{2} f_a$ abzuschneiden. Dabei muss der Tiefpass so dimensioniert werden, dass die Dämpfung bei $f_{\max}$ noch Null ist und bei $\frac{1}{2} f_a$ bereits unendlich.

Zusammenfassend ergibt sich die Aussage, dass man aus den Abtastwerten einer kontinuierlichen, bandbegrenzten Zeitfunktion die ursprüngliche Funktion wieder vollständig rekonstruieren kann, wenn die Voraussetzung $f_a \geq 2 f_{\max}$ erfüllt ist. Dazu muss man aus den Abtastwerten eine Folge von Dirac-Impulsen erzeugen und diese in ein ideales Tiefpassfilter mit $f_g = f_{\max}$ geben.

Wählt man die Abtastfrequenz niedriger als nach dem Abtasttheorem vorgeschrieben, entstehen Spektralanteile mit der Differenzfrequenz $f_a - f < f_{\max}$, die vom Tiefpassfilter nicht unterdrückt werden und sich am Ausgang als Schwebung äußern (Aliasing). Abbildung 19.7 zeigt diese Verhältnisse. Man erkennt, dass die Spektralanteile des Eingangssignals oberhalb von $\frac{1}{2} f_a$ nicht einfach verloren gehen, sondern invers in das Nutzband gespiegelt werden. Die höchste Signalfrequenz $f_{e\,\max}$ findet sich dann als die niedrigste Spiegelfrequenz $f_a - f_{e\,\max} < \frac{1}{2} f_a$ im Basisband des Ausgangsspektrums wieder. In Abb. 19.8 sind diese Verhältnisse für ein Eingangssignal dargestellt, dessen Spektrum nur eine einzige Spektrallinie bei $f_{e\,\max} \lessapprox f_a$ besitzt. Man erkennt, wie hier eine Schwingung mit der Schwebungsfrequenz $f_a - f_{e\,\max}$ zustande kommt.

19.1.1 Praktische Gesichtspunkte

Bei der praktischen Realisierung tritt das Problem auf, dass man mit einem realen System keine Dirac-Impulse erzeugen kann. Man muss die Impulse also gemäß Abb. 19.5 näherungsweise mit endlicher Amplitude und endlicher Dauer erzeugen, d.h. auf den Grenzübergang in Gl. (19.2) verzichten. Durch Einsetzen von Gl. (19.2) in Gl. (19.1) erhalten wir mit endlichem ε die angenäherte Impulsfolge:

$$\tilde{U}'_e(t) = \sum_{\mu=0}^{\infty} U_e(t_\mu) r_\varepsilon(t - t_\mu) \tag{19.5}$$

Durch Fourier-Transformation erhalten wir das Spektrum:

$$\tilde{X}'(jf) = \frac{\sin \pi \varepsilon T_a f}{\pi \varepsilon T_a f} \cdot \tilde{X}(jf) \tag{19.6}$$

(Tietze-Schenk)

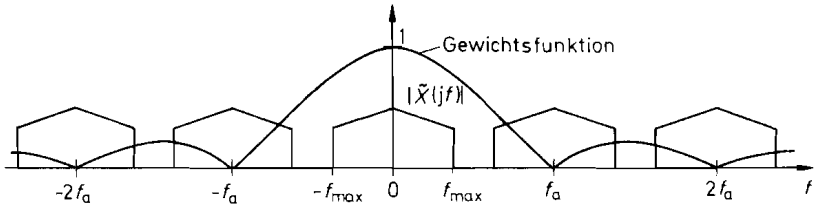


Abb. 19.9. Übergang vom Spektrum der Dirac-Folge zum Spektrum der Treppenfunktion durch die Gewichtsfunktion $|(\sin \pi f/f_a)/(\pi f/f_a)|$

Das ist dasselbe Spektrum wie bei Dirac-Impulsen, jedoch mit einer überlagerten Gewichtsfunktion, die dazu führt, dass höhere Frequenzen abgeschwächt werden. Besonders interessant ist der Fall der Treppenfunktion. Bei ihr ist die Impulsbreite εT_a gleich der Abtastdauer T_a. Dafür ergibt sich das Spektrum:

$$\tilde{X}'(jf) = \frac{\sin(\pi f/f_a)}{\pi f/f_a} \cdot \tilde{X}(jf) \tag{19.7}$$

Der Betrag der Gewichtsfunktion ist in Abb. 19.9 über dem symbolischen Spektrum der Dirac-Impulse aufgezeichnet. Bei der halben Abtastfrequenz tritt eine Abschwächung mit dem Faktor 0,64 auf.

Wie man bei der Wahl der Abtastfrequenz und der Eingangs- bzw. Ausgangsfilter vorgehen kann, soll an dem Beispiel in Abb. 19.10 erklärt werden. Angenommen sei ein Eingangsspektrum eines Musiksignals im Bereich $0 \le f \le f_{max} = 16\,\text{kHz}$, das abgetastet und unverfälscht rekonstruiert werden soll. Dabei ist es unerheblich, ob 16 kHz-Komponenten auch tatsächlich mit voller Amplitude auftreten; der lineare Frequenzgang soll vielmehr andeuten, dass in diesem Bereich eine konstante Verstärkung gefordert wird.

Selbst wenn man sicher ist, dass keine Töne über 16 kHz auftreten, so bedeutet dies nicht automatisch, dass das Spektrum am Eingang des Abtasters auf 16 kHz beschränkt ist. Eine breitbandige Störquelle ist z.B. das Verstärkerrauschen. Aus diesem Grund ist es immer angebracht, den in Abb. 19.1 eingezeichneten Eingangstiefpass vorzusehen. Er soll das Eingangsspektrum auf die halbe Abtastfrequenz begrenzen, um Aliasing zu verhindern. Seine Grenzfrequenz muss mindestens f_{max} betragen, um das Eingangssignal nicht zu beschneiden. Andererseits ist es wünschenswert, dass er bei einer nur wenig höheren Frequenz vollständig sperrt, um einen möglichst niedrigen Wert für die Abtastfrequenz verwenden zu können. Mit der Abtastfrequenz steigt nämlich der Aufwand in den AD- bzw. DA-Umsetzern und im digitalen Filter. Andererseits steigt der Aufwand für den Tiefpass mit zunehmender Filtersteilheit und Sperrdämpfung. Deshalb ist immer ein Kompromiss zwischen dem Aufwand in den Tiefpassfiltern einerseits und den Umsetzern und dem digitalen Filter andererseits zu finden. In dem Beispiel mit $f_{max} = 16\,\text{kHz}$ kann man beispielsweise $\frac{1}{2} f_a = 22\,\text{kHz}$ wählen, also eine Abtastfrequenz von $f_a = 44\,\text{kHz}$ verwenden.

Das bandbegrenzte Eingangssignal wird durch die Abtastung, wie man in Abb. 19.10 erkennt, zu f_a periodisch fortgesetzt. Deshalb muss nach der DA-Umsetzung das Basisband $0 \le f \le \frac{1}{2} f_a$ wieder herausgefiltert werden. Da man am Ausgang des DA-Umsetzers eine Treppenfunktion erhält, muss man noch zusätzlich die $\sin x/x$-Bewertung nach Gl. (19.7) berücksichtigen.

(Tietze-Schenk)

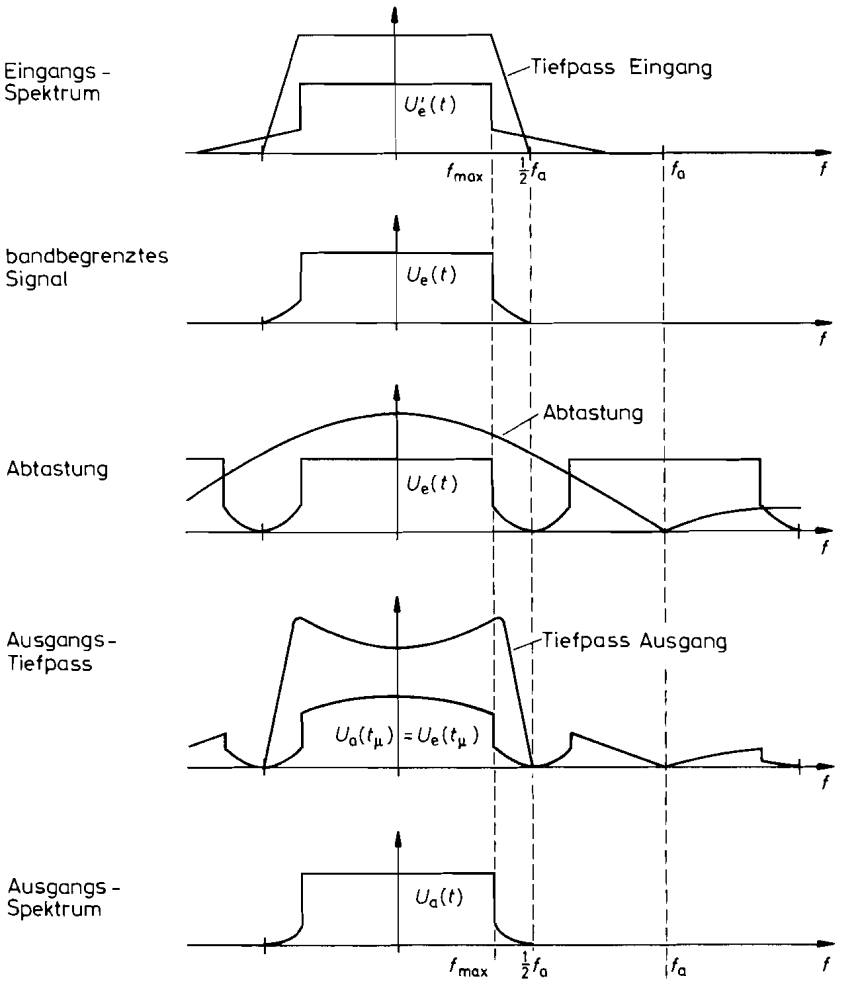


Abb. 19.10. Rekonstruktion des Eingangsspektrums in einem digitalen System gemäß Abb. 19.1 für $y(t_\mu) = x(t_\mu)$

Man kann die dafür erforderliche Entzerrung entweder im Frequenzgang des digitalen Filters berücksichtigen oder im Ausgangs-Tiefpass durchführen. Die letztere Möglichkeit ist in Abb. 19.10 eingezeichnet. Die Hauptaufgabe des Ausgangsfilters besteht aber darin, das Basisband $0 \leq f \leq \frac{1}{2} f_a$ aus dem Spektrum herausfiltern: Bei der Frequenz $f_{\max}$ muss es noch voll durchlässig sein, während es bei der unter Umständen nur knapp darüber liegenden Frequenz $\frac{1}{2} f_a$ schon vollständig sperren soll. Man sieht, dass es hier bezüglich der Filtersteilheit dieselbe Problematik gibt wie beim Eingangsfilter. Um das Filter realisieren zu können, muss also auch hier ein ausreichender Abstand zwischen $f_{\max}$ und $\frac{1}{2} f_a$ bestehen.

Die Problematik, das Eingangs- bzw. Ausgangsfilter zu realisieren, lässt sich entschärfen, wenn man eine deutlich höhere Abtastfrequenz verwendet, also z.B. den doppelten oder vierfachen Wert. Durch diese *Überabtastung* (oversampling) steigt natürlich der Aufwand für die AD- und DA-Umsetzer. Man kann jedoch die Abtastfrequenz mit einem

(Tietze-Schenk)

digitalen Tiefpass hinter dem AD-Umsetzer wieder auf den nach dem Abtasttheorem erforderlichen Wert reduzieren. Dadurch vermeidet man eine Erhöhung der Datenrate bei der Übertragung bzw. Speicherung der Daten. Vor der DA-Umsetzung berechnet man mit einem Interpolator wieder Zwischenwerte, um auch dort durch Überabtastung mit einem einfachen Ausgangstiefpass auskommen zu können [19.2].

19.2 Digitale Übertragungsfunktion

Im Kapitel 13 haben wir gesehen, dass man Analogfilter mit Hilfe von Integratoren, Addierern und Koeffizientengliedern realisieren kann. Der Übergang zum Digitalfilter geschieht dadurch, dass man die Integratoren durch Verzögerungsglieder ersetzt. Solche Verzögerungsglieder kann man z.B. mit Hilfe von Schieberegistern realisieren, durch die man die Abtastwerte der Eingangsfunktion mit der Abtastfrequenz f_a hindurchschiebt. Der einfachste Fall ist die Verzögerung um ein Zeitintervall T_a; ein derartiges Verzögerungsglied ist in Abb. 19.11 schematisch dargestellt.

19.2.1 Beschreibung im Zeitbereich

Gegeben ist die Zahlenfolge $\{x(t_\mu)\} = \{x_\mu\}$, die man sich als Abtastwerte mit einer Wortbreite von 8-, 16- oder 32 bit vorstellen kann. Sie werden in ein Register mit entsprechend vielen parallel-getakteten Flip-Flops geschoben. Die Ausgangsfolge $\{y(t_\mu)\} = \{y_\mu\}$ stellt die um einen Takt T_a verzögerte Eingangsfolge dar. Es gilt also

$$y(t_\mu) = x(t_{\mu-1}) \tag{19.8}$$

19.2.2 Beschreibung im Frequenzbereich

Zur Untersuchung des Frequenzganges gibt man eine Sinusfolge $x(t_\mu) = k_0 \sin \omega t_\mu$ auf den Eingang. Ist das System linear, entsteht auch eine Sinusfolge am Ausgang. Das Verhältnis der Amplituden ist wie bei den Analogfiltern gleich dem Betrag der Übertragungsfunktion für $p = j\omega$. Die Linearität eines Digitalfilters erkennt man an der Linearität der Differenzengleichung. Gemäß Gl. (19.8) ist das Filter in Abb. 19.11 also linear.

Die Übertragungsfunktion kann man wie bei den Analogfiltern direkt mit Hilfe der komplexen Rechnung der Schaltung entnehmen. Dazu benötigen wir den Frequenzgang eines Verzögerungsgliedes: Aus der harmonischen Eingangsfolge

$$x(t_\mu) = \hat{x} e^{j\omega t_\mu}$$

folgt die harmonische Ausgangsfolge

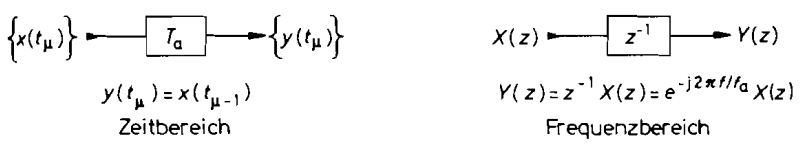


Abb. 19.11. Darstellung eines Verzögerungsgliedes

(Tietze-Schenk)

$$y(t_\mu) = \hat{x}e^{j\omega(t_\mu - T_a)} = \hat{x}e^{j\omega t_\mu} \cdot e^{-j\omega T_a} = x(t_\mu)e^{-j\omega T_a}$$

und mit $j\omega = s$ die Übertragungsfunktion:

$$A(p) = \frac{y(t_\mu)}{x(t_\mu)} = e^{-j\omega T_a} = e^{-sT_a} \tag{19.9}$$

Sie ist eine periodische Funktion mit der Periode $f = f_a = 1/T_a$. Darin ist f_a die Taktfrequenz. Man führt nun die Abkürzung

$$\boxed{z^{-1} = e^{-sT_a} = e^{-j2\pi f/f_a}} \tag{19.10}$$

ein und erhält damit aus Gl. (19.9) die Übertragungsfunktion:

$$\boxed{\tilde{A}(z) = z^{-1}} \tag{19.11}$$

Dies ist die in Abb. 19.11 dargestellte Beschreibung des Verzögerungsgliedes im Frequenzbereich.

Im Kapitel 13 haben wir bereits erwähnt, dass die Übertragungsfunktion $A(p)$ den Zusammenhang zwischen dem Ausgangssignal und einem beliebig von der Zeit abhängigen Eingangssignal über die Laplace-Transformation herstellt gemäß:

$$L\{y(t)\} = A(s) \cdot L\{x(t)\} \tag{19.12}$$

Diese Beziehung gilt auch für ein digitales System. Mit Hilfe der umgeformten Übertragungsfunktion Gl. (19.11) kann man die Beziehung für Zahlenfolgen vereinfachen. Es gilt nämlich:

$$Z\{y(t_\mu)\} = \tilde{A}(z) \cdot Z\{x(t_\mu)\} \tag{19.13}$$

Darin ist

$$Z\{x(t_\mu)\} = X(z) = \sum_{\mu=0}^{\infty} x(t_\mu)z^{-\mu} \tag{19.14}$$

die Z-Transformierte der Eingangsfolge. Die Ausgangsfolge erhält man durch die entsprechende Rücktransformation [19.3, 19.4]. Aufgrund dieser Eigenschaft bezeichnet man $\tilde{A}(z)$ als *digitale Übertragungsfunktion*.

Daraus lassen sich die analoge Übertragungsfunktion bzw. die daraus abgeleiteten Größen wie Betrag, Phase und Gruppenlaufzeit berechnen. Für das Verzögerungsglied folgt aus:

$$\tilde{A}(z) = \frac{Y(z)}{X(z)} = z^{-1} \quad \text{mit} \quad z^{-1} = e^{-j\omega T_a},$$

$$\underline{A}(j\omega) = z^{-1} = e^{-j\omega T_a} = \cos\omega T_a - j\sin\omega T_a$$

Damit ergibt sich der Betrag:

$$|\underline{A}(j\omega)| = \sqrt{\cos^2\omega T_a + \sin^2\omega T_a} = 1$$

die Phase

$$\varphi = \arctan\frac{-\sin\omega T_a}{\cos\omega T_a} = \arctan(-\tan\omega T_a) = -\omega T_a = -2\pi\frac{f}{f_a}$$

(Tietze-Schenk)

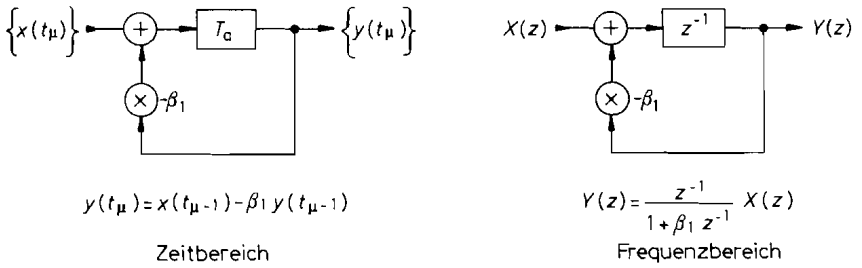


Abb. 19.12. Beispiel für ein rekursives Digitalfilter 1. Ordnung

und die Gruppenlaufzeit:

$$T_{gr} = -\frac{d\varphi}{d\omega} = T_a$$

Beispiel-Tiefpass

Mit den Beziehungen für ein Verzögerungsglied ist die Beschreibung von digitalen Filtern einfach. Am Eingang des Speichers in Abb. 19.12 liegt im Zeitpunkt t_μ der Zahlenwert $x(t_\mu) - \beta_1 y(t_\mu)$. Dieser Wert erscheint eine Taktzeit später am Ausgang des Speichers. Damit erhalten wir für die Werte der Ausgangsfolge die Beziehung:

$$y(t_{\mu+1}) = x(t_\mu) - \beta_1 y(t_\mu)$$

Diese *Differenzengleichung* stellt das Analogon zur Differentialgleichung eines kontinuierlichen Systems dar. Man kann sie als Rekursionsformel zur Berechnung der Ausgangsfolge benutzen, indem man einen Startwert $y(t_0)$ vorgibt. Als Beispiel wählen wir $y(t_0) = 0$ und berechnen die Sprungantwort für $\beta_1 = -0{,}75$. Sie ist in Abb. 19.13 aufgezeichnet. Man erkennt, dass die Schaltung ein Tiefpassverhalten aufweist.

Der Frequenzgang des Beispiel-Tiefpasses lässt sich wie beim Verzögerungsglied berechnen. Aus der rechten Darstellung entnehmen wir:

$$Y(z) = [X(z) - \beta_1 Y(z)]z^{-1}$$

Daraus folgt die digitale Übertragungsfunktion:

$$\tilde{A}(z) = \frac{Y(z)}{X(z)} = \frac{z^{-1}}{1 + \beta_1 z^{-1}}$$

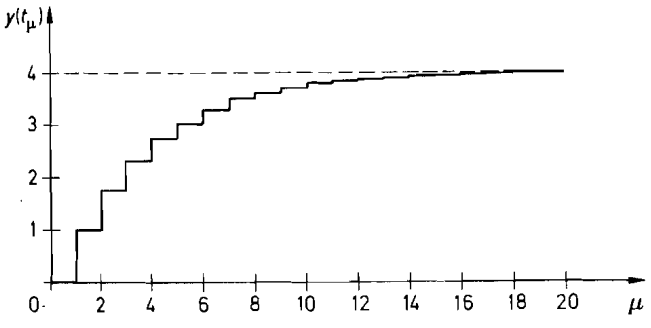


Abb. 19.13. Sprungantwort des Digitalfilters in Abb. 19.12 für $\beta_1 = -0{,}75$ bei einem Eingangssprung von 0 auf 1

(Tietze-Schenk)

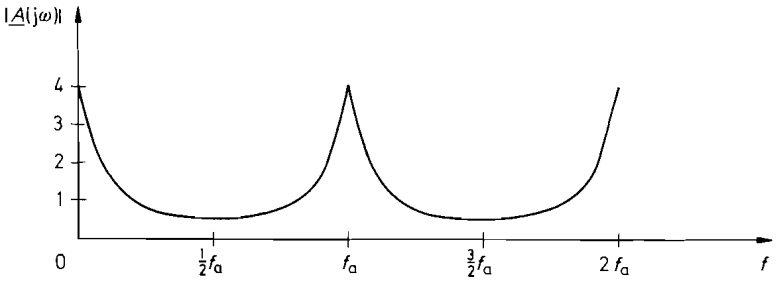


Abb. 19.14. Amplitudenfrequenzgang des Digitalfilters in Abb. 19.12 für $\beta_1 = -0{,}75$

Zur Berechnung des Frequenzganges setzen wir

$$z^{-1} = e^{-j\omega T_a} = \cos\omega T_a - j\sin\omega T_a$$

und erhalten:

$$\underline{A}(j\omega) = \frac{1}{\beta_1 + e^{j\omega T_a}} = \frac{1}{\beta_1 + \cos\omega T_a + j\sin\omega T_a}$$

Mit $\omega T_a = 2\pi f/f_a$ ergibt sich daraus der Betrag:

$$|\underline{A}(j\omega)| = \frac{1}{\sqrt{(\beta_1 + \cos 2\pi f/f_a)^2 + (\sin 2\pi f/f_a)^2}}$$

Man erkennt in Abb. 19.14, dass er mit f_a periodisch ist und zu $\frac{1}{2}f_a$ spiegelbildlich. Diese Eigenschaft ist allen Digitalfiltern gemeinsam. Der Frequenzbereich oberhalb von $\frac{1}{2}f_a$ lässt sich allerdings nicht nutzen, da man sonst das Abtasttheorem verletzt.

Ein interessanter Sonderfall ergibt sich für $\beta_1 = -1$. Dann lässt sich der Betrag der Übertragungsfunktion vereinfachen gemäß $\cos^2 x + \sin^2 x = 1$:

$$|\underline{A}(j\omega)| = \frac{1}{\sqrt{2 - 2\cos 2\pi f/f_a}} = \frac{1}{\sqrt{4(\sin \pi f/f_a)^2}} = \frac{1}{2\sin \pi f/f_a}$$

Für niedrige Frequenzen $f \ll f_a$ erhalten wir daraus mit $\sin x \approx x$:

$$|\underline{A}(j\omega)| = \frac{f_a}{2\pi f} \sim \frac{1}{f}$$

also den Frequenzgang eines Integrators. Die resultierende Schaltung ist die übliche Anordnung für einen Summierer bzw. Akkumulator.

19.3 Grundstrukturen

Zur Realisierung von digitalen Filtern gibt es – abgesehen von Abzweigfiltern – drei Anordnungen, die in den Abbildungen 19.15 bis 19.17 dargestellt sind. Sie besitzen alle drei dieselben Übertragungsfunktionen, wenn man die Filterkoeffizienten α_k und β_k jeweils an den eingetragenen Stellen einsetzt [19.5, 19.6, 19.7].

Man erkennt in Abb. 19.15 bis 19.17, dass die Filter neben den Verzögerungsgliedern Multiplizierer benötigen, die die Variablen mit den festen Filterkoeffizienten multiplizieren, und Summierer, die zwei bzw. drei Zahlen addieren. Die Struktur in Abb. 19.15 ist die

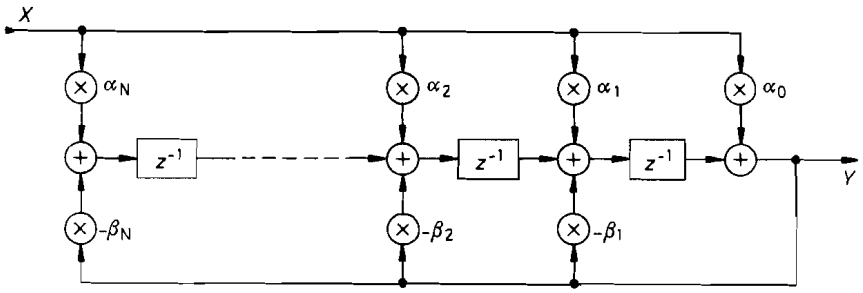


Abb. 19.15. Digitales Filter mit verteilten Summierern

gebräuchlichste, da hier jede Multiplizierer-Akkumulator-Stufe (MAC) von der nächsten durch ein Verzögerungsglied getrennt ist. Dadurch steht für diese Operationen eine ganze Taktdauer zur Verfügung. Die Verzögerungsglieder ergeben hier eine „Pipeline"-Struktur. Bei den beiden anderen Schaltungen müssen viele Variablen in einem einzigen Takt addiert werden. Das erfordert zwar nicht mehr Addierer, aber mehr Rechenzeit.

Bei der Schaltung in Abb. 19.16 erkennt man, dass sich das Eingangssignal für die Verzögerungskette aus dem Eingangssignal X und allen gewichteten Zwischenwerten ergibt. Entsprechend ist das Ausgangssignal die gewichtete Summe aller Zwischenwerte. Man kann daher die Addierer zu zwei globalen Addierern zusammenfassen: einen am Eingang und einen am Ausgang.

Bei der Schaltung in Abb. 19.17 gibt es nur einen einzigen globalen Addierer am Ausgang. Er summiert sowohl das verzögerte und gewichtete Eingangssignal als auch das verzögerte und gewichtete Ausgangssignal. Dazu ist eine zusätzliche zweite Verzögerungskette erforderlich, die am Ausgang angeschlossen ist. Der Mehraufwand dafür ist jedoch gering.

Die Zahl der Filterstufen gibt die Ordnung N des Filters an. Man benötigt je Stufe 1 Verzögerungsglied (2 bei Abb. 19.17), 2 Koeffizienten-Multiplizierer, und man muss 3 Summanden addieren. Lediglich die erste bzw. letzte Stufe ist etwas einfacher.

Die Analyse der Schaltungen soll am Beispiel von Abb. 19.15 gezeigt werden. Die Differenzengleichung lautet:

$$y(t_N) = \sum_{k=0}^{N} \alpha_k x_{N-k} - \sum_{k-1}^{N} \beta_k y_{N-k} \tag{19.15}$$

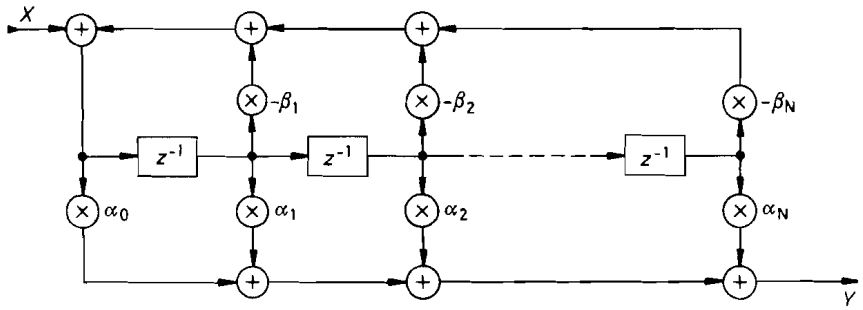


Abb. 19.16. Digitales Filter mit je einem globalen Summierer am Eingang und Ausgang

(Tietze-Schenk)

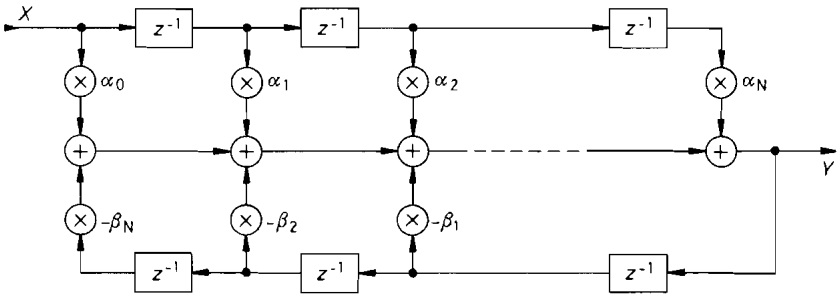


Abb. 19.17. Digitales Filter mit einem einzigen globalen Summierer am Ausgang

Für die Übertragungsfunktion erhält man aus der Schaltung die Beziehung:

$$Y(z) = \sum_{k=0}^{N} \alpha_k z^{-k} X(z) - \sum_{k=1}^{N} \beta_k z^{-k} Y(z)$$

Daraus ergibt sich die Übertragungsfunktion:

$$A(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} \alpha_k z^{-k}}{1 + \sum_{k=1}^{N} \beta_k z^{-k}} \tag{19.16}$$

$$A(z) = \frac{\alpha_0 + \alpha_1 z^{-1} + \alpha_2 z^{-2} + \ldots + \alpha_{N-1} z^{-(N-1)} + \alpha_N z^{-N}}{1 + \beta_1 z^{-1} + \beta_2 z^{-2} + \ldots + \beta_{N-1} z^{-(N-1)} + \beta_N z^{-N}}$$

Zur Berechnung des komplexen Frequenzganges setzt man wieder:

$$z^{-1} = e^{-j\omega T_a} = \cos \omega T_a - j \sin \omega T_a$$

Zusätzlich ist es zweckmäßig, alle Frequenzen auf die Abtastfrequenz $f_a = 1/T_a$ zu normieren. Damit ergibt sich die normierte Frequenzvariable F:

$$\boxed{F = \frac{f}{f_a} \quad \text{bzw.} \quad \omega T_a = 2\pi F} \tag{19.17}$$

Um das Abtasttheorem nicht zu verletzen, muss gelten:

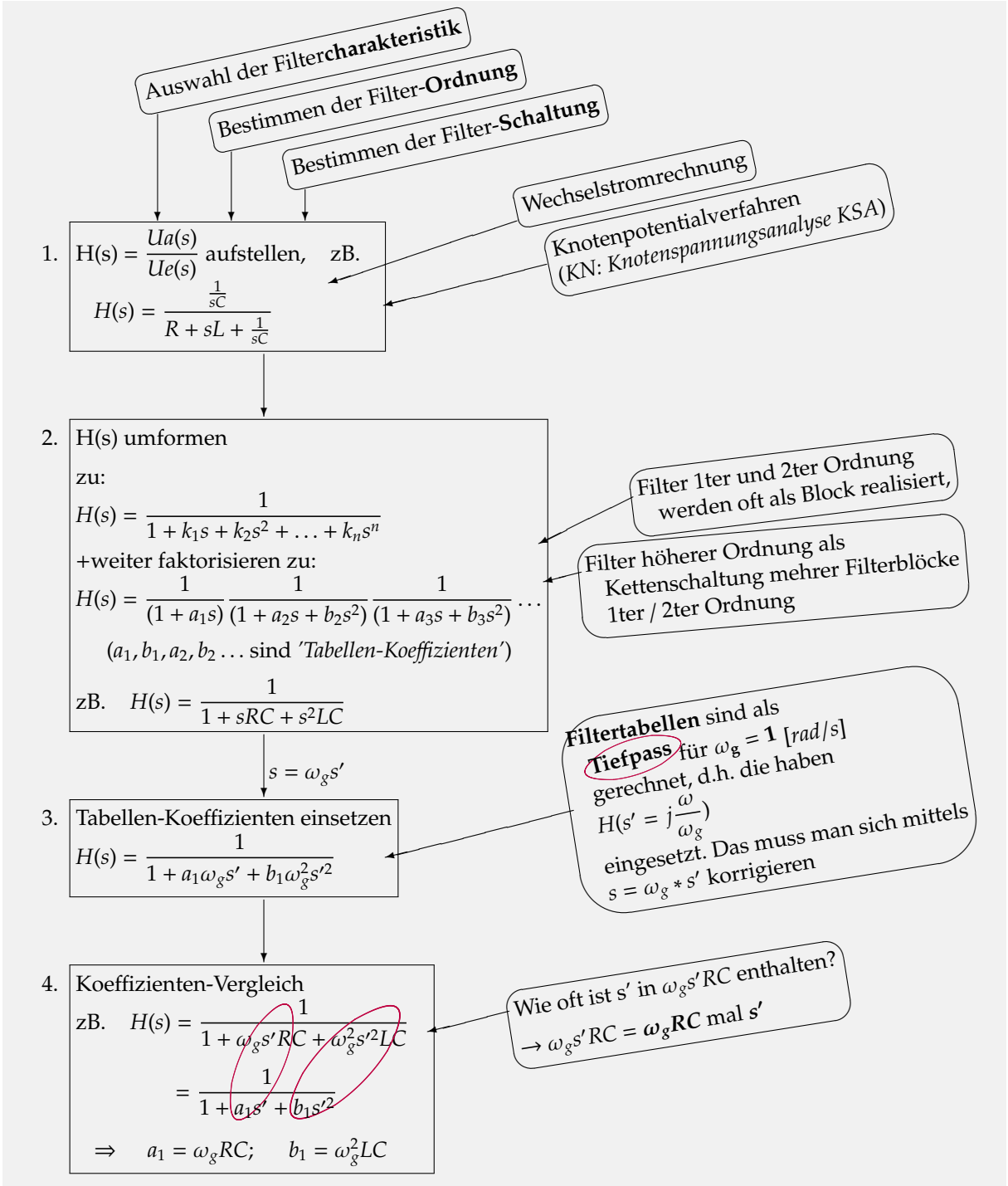
$$\boxed{0 \le f \le \frac{1}{2} f_a \quad \text{bzw.} \quad 0 \le F \le \frac{1}{2}}$$

Für den Betrag des komplexen Frequenzganges folgt damit aus Gl. (19.16):

$$|\underline{A}(j\omega)| = \sqrt{\frac{\left[\sum_{k=0}^{N} \alpha_k \cos 2\pi k F\right]^2 + \left[\sum_{k=0}^{N} \alpha_k \sin 2\pi k F\right]^2}{\left[\sum_{k=0}^{N} \beta_k \cos 2\pi k F\right]^2 + \left[\sum_{k=0}^{N} \beta_k \sin 2\pi k F\right]^2}} \tag{19.18}$$

(Tietze-Schenk)

25.4 kontinuierliche, analoge Filter - DesignMethode



25.5 Ordnung und Reihenfolge

Wenn die Filtercharakteristik nicht scharf genug ist, muss man Filter höherer Ordnung verwenden. Dazu schaltet man Filter erster und zweiter Ordnung in Reihe. Dabei multiplizieren sich die Frequenzgänge der einzelnen Filter. Die Koeffizienten, die für a1 und b1 zu wählen sind, um die Grenzfrequenz fg zu erzielen, sind für jede Stufe verschieden und in der Literatur tabelliert, z.B. in Tietze Schenk. Entsprechend können dann die Bauelemente der verschiedenen Stufen für die gemeinsame Grenzfrequenz fg bestimmt werden.

Im Prinzip ist es gleichgültig, in welcher Reihenfolge man die einzelnen Filterstufen anordnet, da der resultierende Frequenzgang immer derselbe bleibt. In der Praxis gibt es jedoch verschiedene Gesichtspunkte für die Reihenfolge der Filterstufen, z.B. die Aussteuerbarkeit. Nach diesem Gesichtspunkt ist es günstig, die Teilfilter der [*steigenden(xh)*] Grenzfrequenz nach zu ordnen und das mit der niedrigsten Grenzfrequenz

an den Eingang zu schalten. Sonst kann die erste Stufe bereits übersteuert werden, wenn am Ausgang der zweiten noch keine Vollaussteuerung auftritt. Das kommt daher, dass die Filterstufen mit der höheren Grenzfrequenz durchweg eine höhere Polgüte besitzen und damit auch einen Anstieg der Verstärkung in der Nähe ihrer Grenzfrequenz aufweisen.
(aus ZHAW School of Engineering, EK2, HS2009)

25.6 Tiefpass/Hochpass Transformation

In der logarithmischen Darstellung kommt man vom Tiefpass zum analogen Hochpass, indem man die **Frequenzgangkurve der Verstärkung** (Bodediagramm)

an der Grenzfrequenz spiegelt,

d.h. Ω durch 1/Ω bzw. P durch 1/P ersetzt.

(mit $\Omega = \frac{\omega}{\omega_g}$, $P = j\Omega$)

Die Grenzfrequenz bleibt dabei erhalten und A_0 geht in A_∞ über. $H(P = j\omega/\omega_g)$ lautet dann

$$H(P) = \frac{A_\infty}{\prod_i (1 + \frac{a_1}{P} + \frac{b_i}{P^2})}$$

Die Überlegungen können allerdings nicht auf das Verhalten im Zeitbereich übernommen werden, da die Sprungantwort ein prinzipiell anderes Verhalten aufweist. (Es ergibt sich selbst bei Hochpassfiltern mit kritischer Dämpfung eine Schwingung um den stationären Wert.) Die Übertragungsfunktion eines Tiefpasses erster Ordnung lautet allgemein:

$$H(P) = \frac{A_0}{1 + a_1 P}$$

Um den analogen Hochpass zu erhalten, muss man P durch 1/P ersetzen.

$$H(P) = \frac{A_0}{1 + a_1/P}$$

In der Schaltung lässt sich dies ganz einfach dadurch realisieren, dass man **R mit C vertauscht** (was natürlich eine andere Uebertragungsfunktion ergibt).

25.7 Knotenpotentialverfahren

Das Knotenpotentialverfahren (auch Knotenspannungsanalyse oder Knotenadmittanzverfahren) ist ein Verfahren zur Netzwerkanalyse in der Elektrotechnik. Mit dieser Methode lassen sich die Knotenpotentiale eines elektrischen Netzwerks aus linearen Bauelementen bestimmen.

Anwendung des Verfahrens
Das Verfahren wird gewöhnlich zur Bestimmung eines Stromes in einem Zweig verwendet. Gegenüber der Zweigstromanalyse werden bei diesem Verfahren so viele Gleichungen eingespart, wie das Netzwerk unabhängige Maschen besitzt. Im folgenden werden alle Schritte zum gesuchten Wert aufgezeigt. Dieses Verfahren gilt auch für komplexe und magnetische Netzwerke, sofern nur lineare Bauelemente vorkommen.

Knotenpotentiale und Bezugsknoten festlegen
Bei einem Netzwerk mit k Knoten gibt es k-1 unabhängige Knotengleichungen. Für einen Knoten muss keine Gleichung aufgestellt werden, da sich dessen Gleichung aus den Gleichungen der anderen Knoten aufstellen ließe und damit linear abhängig wäre. Dieser Knoten ist deshalb der Bezugsknoten mit Nullpotential (Masse) und kann beliebig gewählt werden. Zweckmäßigerweise sollte der Knoten an einem Zweig mit gesuchtem Spannungsabfall liegen, da so schon ein benötigtes Potential feststeht und das Gleichungssystem einmal weniger gelöst werden muss. Alle anderen Potentiale sind noch unbekannt und werden mit einem eindeutigen Variablennamen bezeichnet.

Umwandlung der Widerstände und Spannungsquellen
Die Zweigströme werden als Produkt aus Zweigleitwert und Knotenpotenzialdifferenz ausgedrückt. Deshalb werden die Zweigwiderstände durch deren Leitwerte ersetzt und die Spannungsquellen nach dem Norton-Theorem in Ersatzstromquellen umgeformt. Ideale Spannungsquellen ohne Widerstand im Zweig können nicht umgeformt werden. Weiteres dazu im Punkt Behandlung von idealen Spannungsquellen.

Matrix des linearen Gleichungssystems aufstellen
Die Leitwertmatrix wird wie folgt aufgestellt:

* Auf der Hauptdiagonalen $\forall i, j$ mit $i = j$ steht die Summe der Leitwerte aller Zweige, die mit Knoten i verbunden sind.

* An den anderen Stellen $\forall i, j$ mit $i \neq j$ steht die negative Summe der Leitwerte zwischen den benachbarten Knoten i und j (Koppelleitwerte). Besteht keine direkte Verbindung zwischen zwei Knoten, wird an dieser Stelle eine Null eingetragen.

Die Leitwertmatrix ist eine symmetrische Matrix. Folglich sind die gegenüberliegenden Koppelleitwertwerte (bezüglich der Hauptdiagonale) identisch ($G_{ij} = Gji \forall i, jmiti \neq j$). Das muss so sein, weil sich diese Koppelleitwerte in beiden Fällen zwischen denselben Knoten befinden. Im Gegensatz zu den positiven Summenleitwerten auf der Hauptdiagonalen sind alle Koppelleit-

werte negativ.
Im Vektor der Knotenpotentiale muss die gleiche Reihenfolge wie auf der Hauptdiagonalen der Leitwertmatrix eingehalten werden.
Im Vektor der Knotenströme auf der anderen Seite des Gleichungssystems steht die Summe der Ersatzstromquellen mit denen der jeweilige Knoten verbunden ist. Hinfließende Ströme gehen positiv, wegfließende Ströme gehen negativ in die Summe ein (es geht auch andersherum, es muss nur einheitlich für alle Knoten erfolgen). Sind keine Quellen mit dem Knoten verbunden, wird eine Null eingetragen.

Behandlung idealer Spannungsquellen
In sehr seltenen Fällen kann sich eine ideale Spannungsquelle (ohne Innen-/Zweigwiderstand) in einem Zweig zwischen zwei Knoten befinden. Dadurch ist die Spannungsdifferenz zwischen den beiden Knoten bekannt und ein Potential kann mit Hilfe des konstanten Wertes der Quellspannung aus dem anderen direkt berechnet werden. Dabei ist zu beachten, in welche Richtung die Spannung der Quelle abfällt:

$$\phi_{high} - \phi_{low} = U_{q_{ideal}}$$

mit Spannungsabfall von "high"-Knoten zu "low"-Knoten
Die Gleichung wird nach dem zu ersetzenden Potential umgeformt und in das Gleichungssystem eingesetzt. Falls einer der Knoten der Bezugsknoten ist, muss selbstverständlich das Potential des anderen ersetzt werden. Im Gleichungssystem wird der eingesetzte Term in jeder Zeile mit den Leitwerten in der zugehörigen Spalte multipliziert. Die Terme mit $U_{q_{ideal}}$ werden auf die Seite der Stromquellen verschoben.

Das weitere Vorgehen ist nun abhängig von der Position des Bezugsknotens. Alle Zeilen der ersetzten Potentiale, deren ideale Spannungsquelle mit dem Bezugsknoten direkt verbunden ist, müssen gestrichen werden. Dadurch reduziert sich der Grad des Gleichungssystems mit jeder idealen Spannungsquelle am Bezugsknoten um Eins. Für alle anderen idealen Spannungsquellen wird in ihren Zweig ein unbekannter Zweigstrom eingeführt. Diese werden zunächst auf die Seite der Stromquellen nach dem gleichen Schema wie die Stromquellen eingetragen. Hinfließende addiert, Wegfließende subtrahiert. Abschließend werden die unbekannten Zweigströme auf die linke Seite gebracht. Für ideale Spannungsquellen ohne direkte Verbindung zum Bezugsknoten reduziert sich der Grad des Gleichungssystems folglich nicht, da für jedes entfallene Potential ein unbekannter Strom hinzukommt.

Gesuchte Potentiale berechnen
Vor der Berechnung eines Zweigstromes müssen die Potenziale der beiden angrenzenden Knoten (ϕ_i und ϕ_j) bekannt sein. Dazu wird das Gleichungssystem für eines der i, j Potentiale gelöst. Dies geschieht entweder mit Hilfe der cramerschen Regel oder durch das Gaußsche Eliminationsverfahren. Sollte einer davon der Bezugsknoten sein, muss nur ein Potential berechnet werden. Die Zweigspannung wird durch die Differenz der Knotenpotentiale für gewöhnlich so berechnet, dass die resultierende Zweigspannung in die vermutete Richtung des gesuchten Stroms abfällt. Der Wert einer eventuell vorhandenen Spannungsquelle im Zweig muss nach dem Maschensatz von der Zweigspannung subtrahiert werden, wenn ihre Spannung in Richtung Zweigspannung abfällt, oder addiert werden, wenn sie in entgegengesetzte Richtung verläuft. Das Ergebnis wird anschließend durch den Zweigwiderstand geteilt bzw. mit dem Zweigleitwert multipliziert, um den gesuchten Strom zu erhalten. Ein positiver Zweigstrom fließt in Richtung des Spannungsabfalls der Knotenpotenzialdifferenz, ein negativer Zweigstrom in entgegengesetzte Richtung.

$$I_{ij} = \frac{\phi_i - \phi_j \pm U_{q_{ij}}}{R_{ij}} = \phi_i - \phi_j \pm U_{q_{ij}} \cdot G_{ij}$$

Anwendung
Das Knotenpotentialverfahren eignet sich hervorragend zur computerunterstützten Berechnung des Lösungsvektors, da sein Lineares Gleichungssystem durch einen einfacher zu programmierenden Algorithmus aufgestellt werden kann als beim Maschenstromverfahren, bei dem zunächst das Netzwerk graphentheoretisch nach einem vollständigen Baum abgesucht werden muss. Es bildet deshalb die Basis der meisten Rechnerprogramme zur Analyse Linearer Elektrischer Netzwerke. Allerdings ist die optimale Auswahl des zu verwendenden Netzwerkanalyseverfahrens abhängig von der Struktur des Netzwerks (Anzahl der Zweige verglichen mit der Anzahl der Knoten) und in der Praxis individuell für jedes Netzwerk.

Quelle: http://de.wikipedia.org/w/index.php?oldid=94636288 Bearbeiter: Aka, Backsideficker, Balumir, Biezl, BuSchu, Ephraim33, Gereon K., Heinte, Ilion, Jodo, Mathemaduenn, Mik81, Pavel007, Regi51, Reseka, Saehrimnir, Segelmaus, Steevie, Stefan Birkner, Tobydox, Tram fan, 35 anonyme Bearbeitungen (21.Apr2012)

25.8 Laplace Transform

Um die Konvergenz für einen größeren Umfang von Funktionen bei der Laplace-Transformation zu sichern, erweitert man die Integration des Fourier-Integrals um den Faktor $e^{-\sigma t}, \sigma \in \mathbb{R}$, *der für* $\sigma > 0$ *und* $t \to \infty$ *gegen 0 geht und so für hinreichend große t die Konvergenz sicherstellt. Damit geht der Frequenzparameter der Fourier-Transformation* $i\omega$ *in den komplexen Frequenzparameter* $s = \sigma + i\omega$ *über.* i *ist die imaginäre Einheit mit* $i^2 = -1$. *(In der Elektrotechnik ist* j *statt* i *gebräuchlich, da dort* i *für den elektrischen Wechselstrom steht.) Im Gegensatz zu der zweiseitigen Laplace-Transformation und auch der Fourier-Transformation ist die gewöhnliche, einseitige Laplace-Transformation nur für positive Werte von* $t \geq 0$ *definiert. Diese Einschränkung ist deswegen zulässig, da im Rahmen der Systemtheorie und Anwendung im Bereich der Physik und Technik nur real existierende kausale Systeme eine Rolle spielen. Bei der theoretischen Untersuchung nichtkausaler Systeme ist die zweiseitige Laplace-Transformation nötig.*

[. . .]

Aufgrund der besseren Konvergenz gegenüber der Fourier-Transformation können beispielsweise Übertragungsfunktionen auch dann noch analysiert werden, wenn sich ein lineares System instabil verhält.

Bei zeitdiskreten Systemen wird, um die Periodizität in der s-Ebene zu vermeiden (diese ist bedingt durch die zeitlich diskreten Abtastwerte im Rahmen der diskreten Laplace-Transformation) eine konforme, nichtlineare Abbildung auf die sogenannte z-Ebene durchgeführt, was zu der Z-Transformation führt.

[. . .]

Die ersten Hinweise auf die Idee der Laplace-Transformation finden sich bereits in den Arbeiten des Basler Mathematikers und Physikers ***Leonhard Euler*** *(1707–1783, Institutiones calculi integralis, vol. 2, 1768). Benannt wird die Laplace-Transformation nach dem französischen Mathematiker und Astronomen Pierre-Simon Laplace (1749–1827), der die Transformation 1782 im Rahmen von Wahrscheinlichkeitsstudien einführte.*

Tatsächlich war der ungarische Mathematiker Jozsef Miksa Petzval (1807–1891) der erste, der sie systematisch untersuchte, wohingegen Laplace sie nur zur Lösung seiner Probleme anwandte. [. . .]

Sei $f : [0, \infty[\to \mathbb{C}$ *eine Funktion. Die Laplace-Transformation von* $f(t)$ *ist durch*
$F(s) = \mathcal{L}\{f\}(s) = \int_0^\infty f(t)e^{-st}\,dt, \quad s \in \mathbb{C}$ *definiert, insofern das Integral existiert. Es handelt sich um ein (uneigentliches) Parameterintegral mit dem Parameter s.*

(aus https://de.wikipedia.org/wiki/Laplace-Transformation*)*

Wos Enk der XH über den Unterschied von Laplace und Fourier erzählt, is natürlich purer Blödsinn (Du weisch ja: *alls was der XH sog, ischa Bledsinn*)

→ **Fourier** analysiert **periodische** Funktionen als H(jω)

→ **Laplace** analysiert auch **Einschwingvorgänge** und **Instabilitäten** als H(s)

→ **z-** is jedenfalls immer für die **digitalisierte** (='diskretisierte') Variante

das Minus
$s = a + jx\ e^s = e^{a+jx} = e^a * e^{jx} = A * e^{jx}$
Ein $\int A * e^{jx}$ kovergiert aber ja nach A nicht (sicher). Deshalb hat man da ein Minus (-) eingesetzt, denn dann nimmt ja e^{-st} exponentiell ab und $\int e^{-st}$ konvergiert.

Das DGL (**D**ifferential-**GL**eichung) -Lösen gelingt, indem man die linke und die rechte Seite getrennt Laplacetransformiert. Die Laplacetransformation sei ja **linear**, dann ist sie auch eine Äquivalenz-Umformung, sodass die Gleichungsbedingung unverändert bleibt:

Links = Rechts
$\Longleftrightarrow$
$\mathcal{L}(Links) = \mathcal{L}(Rechts)$

Beispiel: geg: $a_1 * y'(t) + a_0 * y(t) = b_0 * u(t)$

1. durch a1 dividieren:
$$1 \cdot y'(t) + \frac{a_0}{a1} * y(t) = \frac{b_0}{a_1} u(t)$$
2. Laplacetransformieren:
$$\mathcal{L}\left(y'(t) + {}^{a_0}/_{a_1} \cdot y(t)\right) = \mathcal{L}\left({}^{b_0}/_{a_1} \cdot u(t)\right)$$
3. mit Anfangsbedingung:
$$sY(s) - y(0) + a_0/a_1 Y(s) = b_0/a_1 U(s)$$
4. Laplacetransformieren und nach Y(s) auflösen:
$$Y(s) = \frac{b_0}{s + a_0/a_1} \cdot U(s) + \frac{1}{s + a_0/a_1}$$

Es ist also schon korrekt, dass man DGL mit Laplacetransformation löst, aber nicht, indem man ganze Gleichungen transformiert!

Man transformiert immer linke und rechte Seiten getrennt als **Funktionen**

25.9 LC Filter

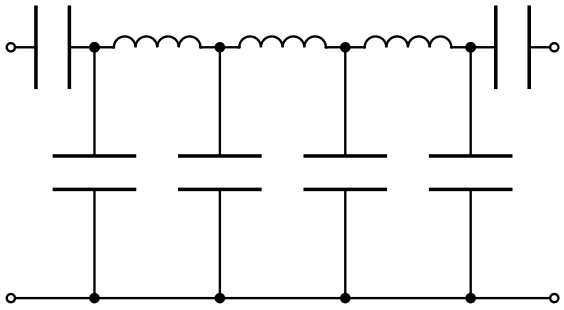
Da es sich um 2 reaktive Bauteile handelt, bilden sie Filter 2-ter Ordnung (ausser bei reiner Serien- oder Parallelschaltung).

- Tief-/Hochpass mit -40dB/Dekade
- Bandpass
- Bandsperre

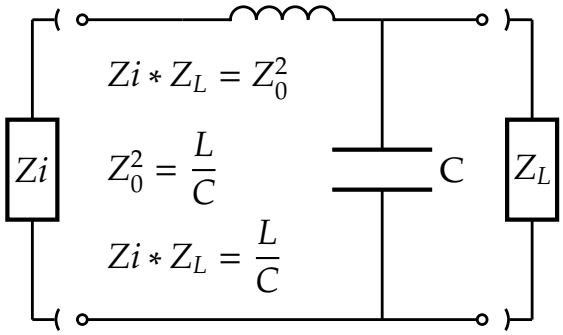
Oft werden sie (zugleich) wegen ihrer *'Nebenwirkungen'*

- Impedanztransformation
- U/I-Phasenverschiebung
- Laufzeitverzögerung

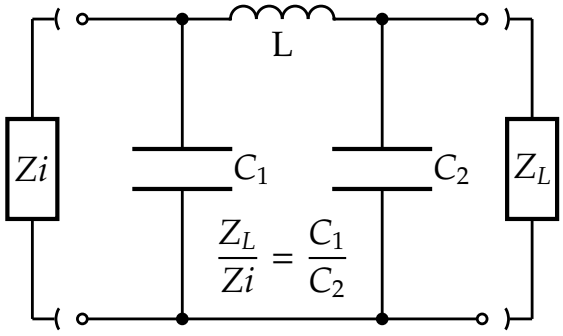
eingesetzt:



25.9.1 LC-, CL- Glied u. Anpassung

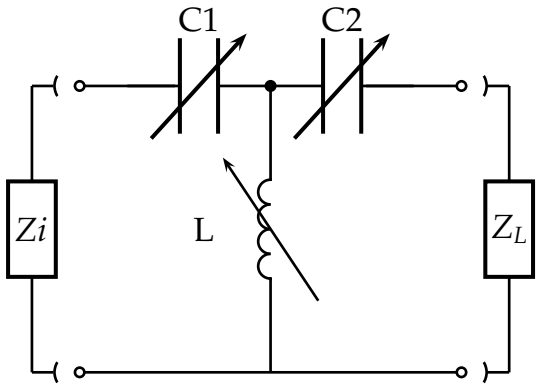


25.9.2 Pi-Glied



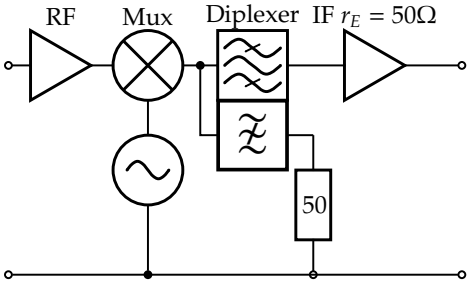
25.9.3 T-Glied

entsteht aus dem Pi-Glied durch Stern-Dreieck-Umformung. T-Glieder haben kleinere Cap-Werte, was bei Drehkondensatoren großer Strom- und Spannungsfestigkeit vorteilhaft ist.



25.9.4 Sonderform Diplexer

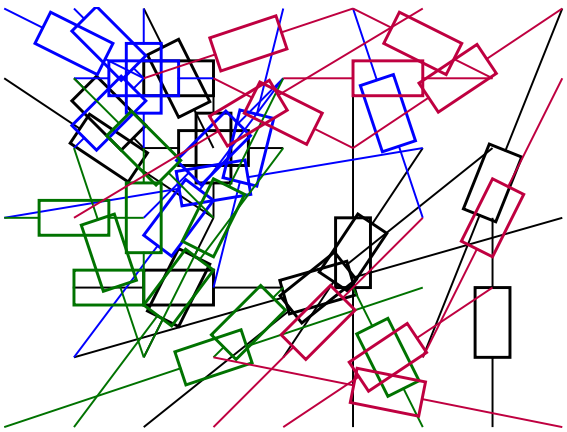
Ein *'Diplexer'* 'verteilt' ein Eingangssignal Ue **frequenzabhängig** auf zwei Ausgänge Ua1, Ua2



Ziel: $R_L(Mux) = 50\Omega$ bei allen Frequenzen:

- Der Bandpass terminiert den Mux bei der IF mit $Z_{in}(IF) = 50\Omega$
- Die Bandsperre terminiert den Mux ausserhalb mit $R = 50\Omega$

25.10 Sallen-Key RC Filter



Rocky P. Sallen und Elvis L. Key schlagen eine vereinfachte Dimensionierung mit gleichen Bauteilwerten R1=R2, C1=C2 vor:

[...] Ein Sallen-Key-Filter, im Englischen auch als Sallen and Key Filter oder als Voltage Controlled Voltage Source Filter (VCVS Filter) bezeichnet, ist ein aktives elektronisches Filter, das aus einem Operationsverstärker und mehreren elektrischen Widerständen und Kondensatoren besteht. Die Bezeichnung leitet sich aus den Namen der beiden Entwickler R. P. Sallen und E. L. Key ab, welche dieses Filter 1955 am Massachusetts Institute of Technology (MIT) entwickelten.

Der Vorteil des Sallen-Key-Filters besteht darin, dass mit minimalem schaltungstechnischem Aufwand ein analoges Filter [...] realisiert werden kann. Es lassen sich durch Anpassen der Filterstruktur sowohl Tiefpassfilter, Hochpassfilter als auch Bandpassfilter realisieren. Da die Übertragungsfunktion aufgrund der Abzweigstruktur keine komplexen Nullstellen aufweisen kann, können damit nur so genannte Allpolfilter, Filter ohne Übertragungsnullstellen, realisiert werden. Bandsperren und Filter mit elliptischen Grundgliedern wie Cauer-Filter können mit Sallen-Key-Filter nicht realisiert werden.

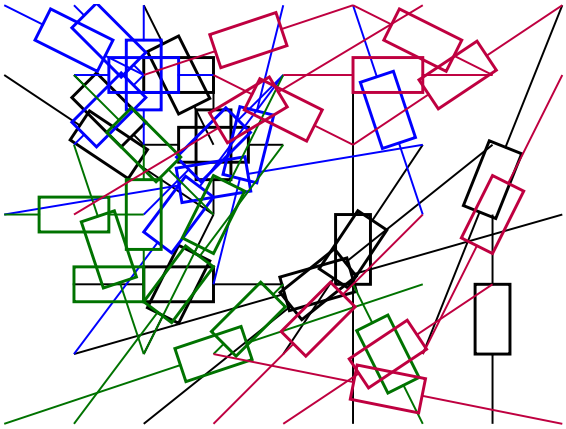
Sallen-Key-Filter weisen dabei immer eine Filterordnung von zwei auf. Dies bedeutet, dass bei einem Tiefpass- bzw. Hochpassfilter der Betragsfrequenzgang eine Änderung von 12 dB pro Oktave erfährt. Höhere Filterordnungen erfordern eine Reihenschaltung mehrerer Sallen-Key-Filter, womit ausschließlich gerade Filterordnungen erzielt werden können. Bei Sallen-Key-Filtern wird der Operationsverstärker üblicherweise mit positivem Verstärkungsfaktor betrieben, auch als Mitkopplungsstruktur bezeichnet. Da sich die

konjugiert-komplexen Pole der Übertragungsfunktion auch mit einem negativen Verstärkungswert realisieren lassen, können Sallen-Key-Filter auch in einer Gegenkopplungsstruktur, allerdings mit höherem Bauelementeaufwand, realisiert werden. Aufgrund des höheren Bauelementeaufwandes bei identischer Übertragungsfunktion spielen Sallen-Key-Filter in Gegenkopplungsstruktur keine wesentliche Rolle. Der Operationsverstärker wird in beiden Fällen in Gegenkopplung betrieben.

Die Filterstruktur ist zudem relativ stabil gegenüber Bauteiltoleranzen, was neben dem einfachen Aufbau die praktische Relevanz begründet. Erkauft werden diese Vorteile durch den Nachteil, dass zur Erzielung eines möglichst hohen Gütefaktors Q extrem hohe und nicht praktikable Bauelementewerte nötig sind. Sallen-Key-Filter werden daher in der analogen Schaltungstechnik bevorzugt dort eingesetzt, wo es auf minimale Bauelementeanzahl mit großem Toleranzschema ankommt und wo geringe Gütefaktoren akzeptierbar sind.

Anwendungsbeispiele in Form eines Tiefpasses finden sich auf der analogen Seite gegen Aliasing unmittelbar vor dem Analog-Digital-Umsetzer und als Filter gegen Spiegelspektren (Anti-Imaging-Filter) unmittelbar nach dem Digital-Analog-Umsetzer in nachrichtentechnischen Systemen. [...] (aus https://de.wikipedia.org/wiki/Sallen-Key-Filter 12May16)

25.11 Baxandall Tonblende



Peter J. Baxandall (August 11, 1921 – September 8, 1995) was an English audio engineer and electronics engineer and a pioneer of the use of analog electronics in audio. He is probably best known for what is now called the Baxandall tone control circuit, first published in a paper in Wireless World.

Baxandall's bass and treble circuit, when made public in Wireless World (1952), "swept all others before it". An early version of the design had already won him an award in 1950 (a $25 watch) at the British Sound Recording Association, a predecessor of the Audio Engineering Society. The design is now employed in millions of hi-fi systems (Baxandall received no royalties for his work).

It exists in several versions—Baxandall's original had two capacitors per potentiometer, but it is possible to use only one at either the treble or bass potentiometers, or both. It finds an application in hi-fi audio equipment and in amplifiers and effects for musical instruments. [...] (aus https://en.wikipedia.org/wiki/Peter_Baxandall 31Aug21)

s.http://makearadio.com/tech/tone.htm

s.https://www.petervis.com/record_players_and_turntables/baxandall/baxandall-tone-circuit.html

s.https://de.wikipedia.org/wiki/Klangregler

26 Anwendungen der MAC Operation

'MAC' ≡ **M**ultiply and **AC**cumulate ≡ multiplizieren und zammzählen

`Programmierung:` `Rechnung (Algorithmus):` `Formel:`

```
a[],b[] ... array
sum=0;
for i=0 to N:
    sum = sum + a[i] * b[i]
```

$=$

$$\begin{aligned} &a_1 \cdot b_1 \\ +\ &a_2 \cdot b_2 \\ +\ &a_3 \cdot b_3 \\ +\ &\ldots \\ \hline &= \text{Summe} \end{aligned}$$

$$\sum_i^N a_i \cdot b_i$$

⊠ Skalarprodukt von Vektoren: $a \cdot b = \sum_1^N (a[i] \cdot b[i])$

Man weiß Vektoren mit 2 oder 3 Dimensionen räumlich zu deuten — rechnen und nutzen kann man aber auch vieldimensionale Vektoren zB. den k-Vektor(Physik) oder Folgen digitaler Signale.

```
a[],b[] ... vector
sum=0;
for i=0 to N:
    sum = sum + a[i] * b[i]
```

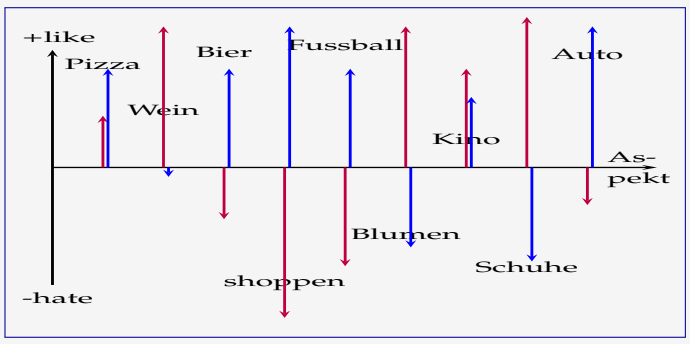
$=$

$$\begin{aligned} &a_1 \cdot b_1 \\ +\ &a_2 \cdot b_2 \\ +\ &a_3 \cdot b_3 \\ +\ &\ldots \\ \hline &= \text{Summe} \end{aligned}$$

$$\sum_i^N a_i \cdot b_i$$

⊠ Partnervermittlung "Pizza-Bier-Blumen"

Bewertet man die Vorlieben von Person_A und von Person_B mit ±Punkten
und multipliziert sie, so ist die Punkte-Summe ein Maß für die Ähnlichkeit dieser Personen



$$\begin{aligned} &A_{Pizza} \cdot B_{Pizza} \\ +\ &A_{Bier} \cdot B_{Bier} \\ +\ &A_{Blumen} \cdot B_{Blumen} \\ +\ &\ldots \\ \hline &= \text{Passung} \end{aligned}$$

$$\text{Affinität} = \sum_{i=1}^{Aspekte} (Punkte_A[i] \cdot Punkte_B[i])$$

⊠ DFT discrete fourier transform

(ohne 'Windowing')

```
CosMag[],SinMag[] ... array
s[] ... signal, digitalisierte samples
for i=1 to MaxFreq {
    CosMag[i] = 0;
    SinMag[i] = 0;
    for j=1 to s_Len{
        CosMag[i] += s[j] · cos(i 2πj / s_Len)
        SinMag[i] += s[j] · sin(i 2πj / s_Len)
    }
}
```

$=$

$$\begin{aligned} &s_1 \cdot cos_{11} \\ +\ &s_2 \cdot cos_{12} \\ +\ &s_3 \cdot cos_{13} \\ +\ &\ldots \\ \hline &= cos_1\text{Mag.} \end{aligned}$$

$$\begin{aligned} &s_1 \cdot cos_{21} \\ +\ &s_2 \cdot cos_{22} \\ +\ &s_3 \cdot cos_{23} \\ +\ &\ldots \\ \hline &= cos_2\text{Mag.} \end{aligned}$$

. . . usw. . . .

$$\begin{aligned} &s_1 \cdot sin_{11} \\ +\ &s_2 \cdot sin_{12} \\ +\ &s_3 \cdot sin_{13} \\ +\ &\ldots \\ \hline &= sin_1\text{Mag.} \end{aligned}$$

$$\begin{aligned} &s_1 \cdot sin_{21} \\ +\ &s_2 \cdot sin_{22} \\ +\ &s_3 \cdot sin_{23} \\ +\ &\ldots \\ \hline &= sin_2\text{Mag.} \end{aligned}$$

. . . usw. . . .

$$cos.Magnitude_i = \sum_j^N s_j \cdot cos(i\frac{2\pi j}{N})$$

$$sin.Magnitude_i = \sum_j^N s_j \cdot sin(i\frac{2\pi j}{N})$$

⊠ digitale FIR Filter

```
Shiftreg[] ... array:Schieberegister
Koeffizient[] ... array:Filterkoeffizienten
Output=0;
for i=1 to N {
    Output +=
    Shiftreg[i] * Koeffizient[i]
}
for i=1 to N-1 {
    Shiftreg[i+1] = Shiftreg[i]
}
Shiftreg[1] = Input
```

$$
\begin{aligned}
&Shiftreg_1 \cdot Koeffizient_1 \\
&+ Shiftreg_2 \cdot Koeffizient_2 \\
&+ Shiftreg_3 \cdot Koeffizient_3 \\
&+ \ldots \\
&\text{= Output} \\
&\text{Registerinhalte schieben}
\end{aligned}
$$

$$\text{Output[t]} = \sum_i Schieberegister[i] \cdot Koeffizient[i]$$

⊠ digitale IIR Filter

```
ShiftregF[] ... array:Schieberegister
ShiftregB[] ... array:Schieberegister
KoeffizientF[] ... array:Filterkoeffizienten
KoeffizientB[] ... array:Filterkoeffizienten
F ... Forward, B ... Back
Output=0
Feedback=0
for i=1 to N {
    Output +=
    ShiftregF[i] * KoeffizientF[i]
    Feedback +=
    ShiftregB[i] * KoeffizientB[i]
}
for i=1 to N-1 {
    ShiftregF[i+1] = ShiftregF[i]
    ShiftregB[i+1] = ShiftregB[i]
}
ShiftregF[1] = Input+Feedback
ShiftregB[1] = Output
```

$$
\begin{aligned}
&Shiftreg_1 \cdot Koeffizient_1 \\
&+ Shiftreg_2 \cdot Koeffizient_2 \\
&+ Shiftreg_3 \cdot Koeffizient_3 \\
&+ \ldots \\
&\text{= Output} \\
&\text{Registerinhalte schieben}
\end{aligned}
$$

$$
\begin{aligned}
&\text{Output[t]} = \sum_i Schieberegister[i] \cdot Koeffizient[i] \\
&\text{Input[t]} = \\
&\text{Input[t-1]} + \\
&\sum_i Schieberegister[i] \cdot Feedback[i]
\end{aligned}
$$

⊠ (gewichtete) Mittelwertbildung:

```
w[] ... Gewichte
v[] ... Werte
Mittel=0;
for i=1 to N {
    Mittel += w[i] * v[i]
}
```

$$
\begin{aligned}
&w_1 \cdot v_1 \\
&+ w_2 \cdot v_2 \\
&+ w_3 \cdot v_3 \\
&+ \ldots \\
&\text{= Mittelwert}
\end{aligned}
$$

$$
\begin{aligned}
&AVG = \textstyle\sum_i v_i \cdot w_i \\
&\text{mit } \textstyle\sum_i w_i = 1 \\
&\text{AVG} \ldots \text{"average"} \\
&v_i \ldots values \\
&w_i \ldots weights
\end{aligned}
$$

⊠ Faltungsoperation '*' (s. auch z-Transformation)

(s. auch z-Transformation)

```
f[] ... digitalisierte Funktion f
g[] ... digitalisierte Funktion g
faltprodukt=0;
for i=1 to N {
    faltprodukt += f[i] * g[N-i]
}
```

$$
\begin{aligned}
&f_1 \cdot g_N \\
&+ f_2 \cdot g_{N-1} \\
&+ f_3 \cdot g_{N-2} \\
&+ \ldots \\
&\text{= Faltprodukt}
\end{aligned}
$$

$$
\begin{aligned}
&f(t) * g(t)(\tau) ::= \int_{-\infty}^{+\infty} f(t) \cdot g(\tau - t) dt \\
&\Rightarrow \\
&f(t) * g(t)(0) ::= \sum_{i=1}^{Anz} f[i] \cdot g[N - i]
\end{aligned}
$$

⊠ Faktura, Buchhaltung, Steuererklärung udgl.

```
Menge[] ... array
Preis[] ... array
Gesamt=0;
for i=1 to N {
   Gesamt += Menge[i] * Preis[i]
}
```

=

$$\begin{aligned} &Menge_1 \cdot Preis_1 \\ &+ Menge_2 \cdot Preis_2 \\ &+ Menge_3 \cdot Preis_3 \\ &+ \ldots \\ &= \text{Gesamt} \end{aligned}$$

$$\sum Menge_i \cdot Einzelpreis_i$$

$$\sum Steuersatz_i \cdot Betrag_i$$

⊠ Discrete Cosinus-Transformation (DCT) (Audio-/Bild-Kompression

```
s[] ... Input, digitalisiert: samples
Mag ... 'Magnitude'
CosMag[] ... array
for i=1 to MaxFreq {
   CosMag[i] = 0;
   for j=1 to Nsamples {
      CosMag[i] += s[j] · cos(i 2πj / N_samples)
   }
}
```

=

$$\begin{aligned} &sample_1 \cdot cos_{11} \\ &+ sample_2 \cdot cos_{12} \\ &+ sample_3 \cdot cos_{13} \\ &+ \ldots \\ &= cos_1\text{Mag.} \\ &sample_1 \cdot cos_{21} \\ &+ sample_2 \cdot cos_{22} \\ &+ sample_3 \cdot cos_{23} \\ &+ \ldots \\ &= cos_2\text{Mag.} \\ &\ldots \text{usw.} \ldots \end{aligned}$$

$$\text{CosMag}[n] ::= \sum_{j=0}^{Anz} s[i] * cos(n * \frac{j * 2\pi}{Anz})$$

⊠ Elektroniksimulation
In allen Stromknoten rechnet man: *'Summe aller Ströme ist Null'*

```
Strom[] ... array
U[] ... array: Knotenspannungen
Y[][] ... array Leitwerte
for i=1 to Nk {
   Strom[i]=0
   for j=1 to Nk {
      Strom[i] += (U[j]-U[i])*Y[i,j]
   }
}
```

=

$$\begin{aligned} &U_1 \cdot Y_{11} \\ &+ U_2 \cdot Y_{12} \\ &+ U_3 \cdot Y_{13} \\ &+ \ldots \\ &= 0\ ? \\ &U_1 \cdot Y_{21} \\ &+ U_2 \cdot Y_{22} \\ &+ U_3 \cdot Y_{23} \\ &+ \ldots \\ &= 0\ ? \\ &\ldots \text{usw.} \ldots \end{aligned}$$

$$\sum_{n=1}^{N_k} I_n = \sum_{n=1}^{N_k} \mathbf{U_n} * \mathbf{Y_n}$$

I_n ... Stromstärke aus Zweig n
U_n ... Spannung am Bauteil im Zweig n
Y_n ... komplexer Leitwert des Bauteils n
zB. 1/R, 1/jwL, jwC
N_k ... Anzahl Zweige zum Knoten k

⊠ Ein ANN (Artificial Neural Net) rechnet

```
SM[][] ... SynapsenMatrix
I...Anz.Inputs
O...Anz.Outputs
for i=1 to N+I {
   SM[i,0]=0
   for j=1 to N+O {
      SM[i,0] += SM[0,j]*SM[i,j]
   }
}
```

=

$$\begin{aligned} &Inp_1 \cdot weight_{11} \\ &+ Inp_2 \cdot weight_{12} \\ &+ Inp_3 \cdot weight_{13} \\ &+ \ldots \\ &= Out_1 \\ &Inp_1 \cdot weight_{21} \\ &+ Inp_2 \cdot weight_{22} \\ &+ Inp_3 \cdot weight_{23} \\ &+ \ldots \\ &= Out_2 \\ &\ldots \text{usw.} \ldots \end{aligned}$$

$$Output_n = \sum_{i=1}^{m} Input_m * Gewicht_{m,n}$$

⊠ Die Defuzzifizierung in der **'Fuzzy Logic'**
errechnet einen *'Flächenschwerpunkt'*
der Fläche unter einer Funktion f(x)

```
cog=0 ... center of gravity
for i=1 to N {
    cog += x[i]*f[i]
}
```

$$\begin{aligned} &x_1 \cdot f_1 \\ &+ x_2 \cdot f_2 \\ &+ x_3 \cdot f_3 \\ &+ \dots \\ &= x_{Schwerpunkt} \end{aligned}$$

$$x\_Koordinate(Schwerpunkt) = \underbrace{\frac{\sum_{x=Xmin}^{Xmax} x \cdot f(x)}{\sum_{x=Xmin}^{Xmax} f(x)}}_{gewichtetesMittel}$$

⊠ Matrixmultiplikation

```
A[][] ... Matrix A: P × Q
B[][] ... Matrix B: Q × P
M[][] ... neue Matrix M: P × P
for i=1 to P {
    for j=1 to P {
        for k=1 to Q {
            M[i,j] = A[i,k] * B[k,j]
        }
    }
}
```

$$\begin{aligned} &A_{11} \cdot B_{11} \\ &+ A_{12} \cdot B_{21} \\ &+ A_{13} \cdot B_{31} \\ &+ \dots \\ &= M_{11} \\ &+ \dots \\ &A_{u1} \cdot B_{1v} \\ &+ A_{u2} \cdot B_{2v} \\ &+ A_{u3} \cdot B_{3v} \\ &+ \dots \\ &= M_{uv} \end{aligned}$$

$$M_{i,j} = \sum_{k=1}^{Q} A_{i,k} \cdot B_{k,j}$$

⊠ Bildverarbeitung - Bildschärfung/Bildglättung/Rauschminderung → s. 'Convolution'

⊠ jpeg Bildkompression → 'Cosinus Transformation'

⊠ mp3 Audiokompression → 'Cosinus Transformation'

27 zu 'Signalverarbeitung': Linear-zeitinvariante (LTI) zeitdiskrete Systeme und deren Beschreibung im Zeit- und Frequenzbereich

'LTI' == Linear + Time Invariant

27.1 'linear' ::= f(ax+by) = af(x)+bf(y)

ist eine Eigenschaft mathematischer Abbildungen, Verknüpfungen, Funktionen. In der Elektronik werden auch Funktionen mit einem *konstanten Y-'Offset'* as *'linear'* angesehen, indem wir die Abbildung (=Übertragung) in einen *'Arbeitspunkt'* und ein *'Kleinsignalverhalten'* zerlegen.

27.2 'zeitinvariant' ::= 'gestern gleich wie heute'

Mit *'zeit-invariant' (time invariant)* ist **nicht** etwa *'zeitunabhängig'*gemeint,
sondern:

das System wird sich (bei gleichem Input und gleichem Anfangszustand) morgen genauso verhalten wie heute und gestern

Es kommt nicht darauf an, wann ich das *'Experiment'* mache/ ≡ das System anwende.

Eine Abhängigkeit besteht nur von $t - t_0$, der Zeit-seit-Systemstart

27.3 Zeitbereich: Signal u(t)

Ein 'Signal' ist eine

Funktion der Zeit f(t, ...);

meistens einer physikalischen Größe wie Spannung, Schalldruck, Helligkeit, Feldstärke, Rauchdichte odgl.

27.4 Frequenzbereich: Spectrum a(ω)

Ein 'Spektrum' ist eine

Funktion der Frequenz f(ω, ...);

d.h. die X-Achse ist $\omega = 2\pi f$ oder $log\omega$; die Y-Achse ist die Verstärkung A (Amplification) oder Abschwächung/Dämpfung a (Attenuation), gewöhnlich auch logarithmisch *loga*, oft in der Maßeinheit [dB] *'deciBel'* mit zB.
20dB ≡ Faktor 10.0
10dB ≡ Faktor 3.16
0dB ≡ Faktor 1.0
-10dB ≡ Faktor 0.316
-20dB ≡ Faktor 0.1
usw.

27.5 Digitalisierung

'digital' heißt

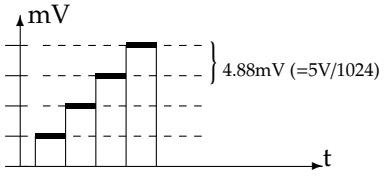
es hat Stufen (ohne Zwischenwerte)

und

'analog' bedeutet *'stufenlos'*

Auflösung, Resolution:

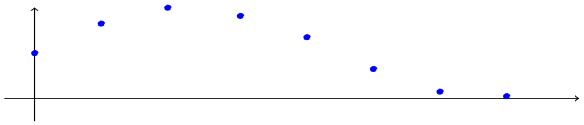
Die *'Feinstufigkeit'*, die Anzahl von Stufen in der Y-Richtung, nennen wir *'Auflösung' (resolution)*, indem wir zB. sagen *'10 Bit Auflösung'* oder *'4.88mV Auflösung'*.



Die Stufen müssen nicht zwingend alle gleich gross sein!
(haben oder möchten wir aber meistens)
(es gibt zB. auch logarithmische Stufung bei *'σ − Δ-Wandlung'*)

27.5.1 Abtastung, Sampling

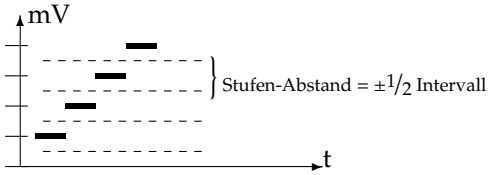
Die Stufen in x-Richtung nennen wir *'Abtastung' (sample)*:
→ Wir haben nicht mehr zu jedem Zeitpunkt einen y-Funktionswert, sondern nur mehr zu den Abtast-Zeitpunkten.



Auch hier ist ungleichmäßige Stufung denkbar (zumindest ungewollt, etwa durch instabilen Takt (jittern), oder zB. absichtlich in sog. 'subharmonischen' Abtastungen oder bei lokaler Steigerung der Abtastrate an Funktionsstellen starker Änderung (das machen mp3 und SPICE-Elektroniksimulatoren — das punktweise Errechnen einer math.Funktion ist ja auch eine Abtastung!).

27.5.2 Quantisierungsfehler

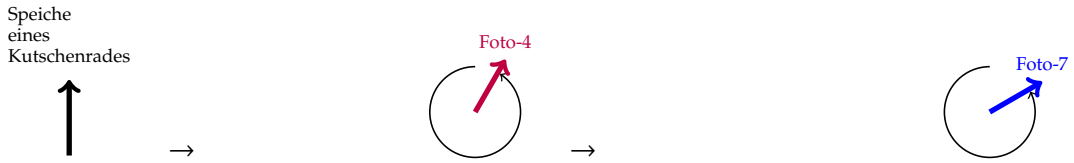
(quantization error) Da Y-Achsen-Werte auf die nächstliegende digital-Stufe abgebildet werden, entsprechen digitalisierte Werte *nicht exakt* den analogen. Das sind Digitalisierungs-Fehler (*quantisation error*, *Quantisierungsrauschen (quantisation noise)*, *Quantisierungseffekte - quantisation phenomena*).



X-Achsen-Timing-Fehler führen ebenso zu Abweichungen vom wirklichen analogen Verlauf, steigern also das Quantisierungsrauschen – sie multiplizieren sich mit der Kurvensteigung (s. Gauß'sche Fehlerfortpflanzung). Auch bei der digital→analog-Wandlung ('Analogisierung') entstehen Quantisierungsfehler aus den Linearitäts-Fehlern des DAC (digital to analog converter), die allerdings meist weniger ins Gewicht fallen, weil man einem DAC ein Tiefpass(TP)-Filter (mindestens ein RC-TP) nachschaltet (wirkt 'glättend').

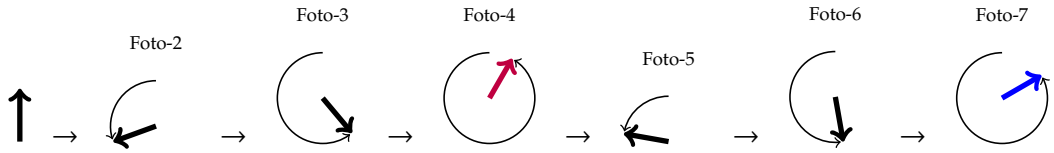
27.5.3 Aliasing (Stroboskopeffekt)

Sicherlich kennst Du diesen Stroboskop-Effekt,
wo sich in den Wildwest-Filmen die Kutschenräder
beim Stehenbleiben scheinbar rückwärts drehen:



Verursacht durch die "Abtastung" der realen Bewegung
mit **nur** 25 Bildern pro Sekunde (samples/s)
(hier mittels fehlender Bilder symbolisiert).

mit 75 Bildern pro Sekunde hingegen



sieht man, dass das Wagenrad nicht wirklich rückwärts dreht.

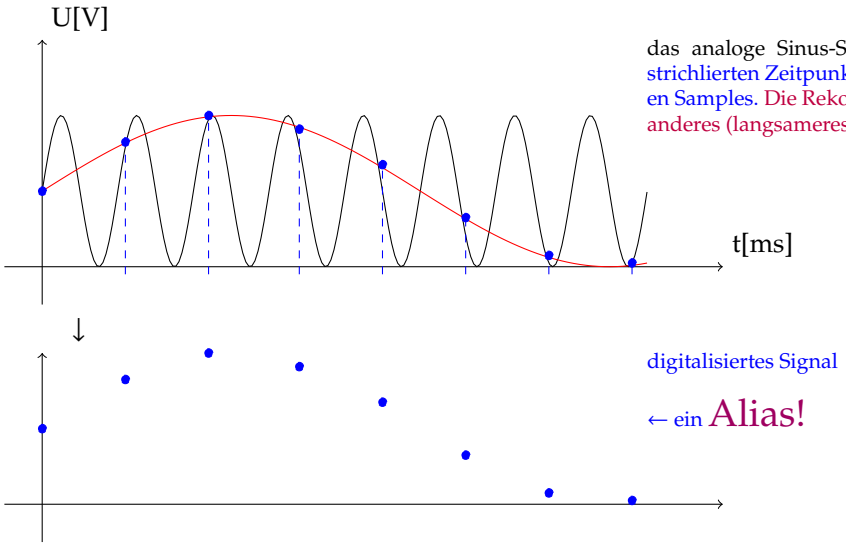
Demo – die 'fliegenden Wassertropfen' (n19aeitna):
→ https://youtu.be/ms-9GtBepNA (08Nov'22)

→ Dasselbe Phänomen gibt es akustisch als

Phantom-Signal,

dh. man hört auf der Aufnahme Töne,
die in Wirklichkeit nicht da waren!
Der Effekt heisst *'Aliasing'*, die Phantomsignale *'Aliases'*

Aliasing ist also ein Stroboskop-Effekt



Aliases haben dieselbe Amplitude wie die verursachenden Signale (s.Skizze)
→ich muss entweder für ausreichend hohe Abtastrate sorgen

$$f_{sampling} > 2 * f_{Nyquist}$$

Nyquist-Frequenz . . . halbe Abtastrate/sampling rate
oder
höhere Frequenzanteile ausreichend filtern/dämpfen

⇒ anti-Aliasing-Filter

Meist sind sehr sehr steilflankige Filter erforderlich. Zur CD-Digitalisierung etwa will man bis 20kHz übertragen, bei 22050Hz Nyquistfrequenz will man -96dB Dämpfung (Auflösung/Dynamikbereich der 16-Bit-Digitalisierung) - kaum zu schaffen!

Aliasing JScript Simulation
link → http://10.10.63.61/SMUE/shr/js/DSP/KSN15/Aliasing/jsAliasXhQbR.html
http://www.qsl.net/oe7csj/zuig/jsAliasXhQbR.html

27.5.4 Oversampling (Überabtastung)

Verdoppeln der Abtastrate bringt die Vorteile

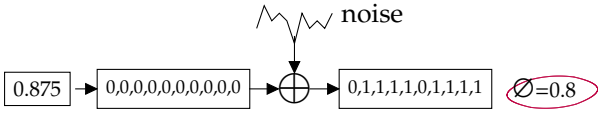
↪ viel mehr Frequenz-'Raum' für die Aliasing-Filter
zB. [20–96kHz] statt [20–22.05kHz] (192kHz oversampling)

↪ höhere Auflösung bei Mittelwertbildung ($\frac{a+b}{2}$):

$$\frac{0+0}{2} = 0; \quad \frac{0+1}{2} = \frac{1+0}{2} = 0.5; \quad \frac{1+1}{2} = 1$$

27.5.5 Dithering

Die Auflösung lässt sich steigern, indem dem Input noch vor der Digitalisierung ein **schwaches Störsignal** addiert wird, sodass die Amplitudengrenze zweier benachbarter Binärwerte wiederholt über- und unterschritten wird, der Binärwert somit *'zittert'*, (s. → Dithering) Bekannte Störsignale lassen sich dann digitalisiert wieder herausrechnen.



Unbekannte Störsignale wie zB. Rauschen lassen sich mittels Oversampling und Antialias-Filterung (Oversampling verschiebt das Rauschspektrum *'nach oben'*) unterdrücken.

Dithering ist besonders in der Bildverarbeitung gängig, wo zudem der Umstand hilft, dass die menschliche Augen-Auflösung farbabhängig ist (hell-dunkel: besser, grün: schlecht, rot: schlechter, blau: ganz schlecht) – man kann eine höhere Farbtiefe vorschwindeln bzw. die technische Farbauflösung zwecks Datenkompression reduzieren.

27.6 DSP Algorithmen

27.6.1 Decimation

Die Sample-Rate wird reduziert zB. durch Weglassungen

27.6.2 Interpolation

Ein Sample-Wert wird durch einen Mittelwert ersetzt

27.6.3 Decimation+Interpolation → Prozessgewinn

Die Mittelung steigert die Auflösung. Der Gewinn beträgt 6.02 dB (20*log 2) pro Bit

27.6.4 exponentielle Dämpfung $y_i = \alpha x_i + (1 - \alpha) y_{i-1}$

27.6.5 Mittelung (Oszi, Speki)

Oszilloskope und Spektrumanalysatoren bieten meist (der visuellen Rauschreduktion dienliche) Mittelwert-Funktionen zB.

$y = 1/2(sample[0] + sample[1])$

$y = 1/4(sample[0] + sample[1] + sample[2] + sample[3])$

• • •

$y = 1/16(sample[0]+sample[1]+sample[2]+sample[3]sample[4]+sample[5]+sample[6]+sample[7]sample[8]+sample[9] + sample[10] + sample[11]sample[12] + sample[13] + sample[14] + sample[15])$

usw.

27.6.6 .Multiplizieren, Mischen, Abtasten, XOR, Falten - alles dasselbe?

oder auch nicht?

TBD. (bedeutet 'to be done' ≡ in Arbeit)

27.6.7 DDC

'digital down conversion'
'Mischt das digitale Signal eines hoch abgetasteten, analogen Signals in das Basisband oder in einen Zwischenfrequenzbereich herunter.'

27.7 Signalanalyse

27.7.1 DFT

'discrete fourier transform'

$$F_n = \sum_{k=0}^{N-1} f_k e^{-j\pi nk/N}$$

27.7.2 iDFT

inverse DFT, DFT^{-1}

$$f_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{2j\pi kn/N}$$

Rücktransformation des digitalisierten Spektrums

27.8 Signalgenerierung

27.8.1 DAC s.→ s.Kap.14.8, 188

27.8.2 DDS

'direct digital synthesis' ersetzt Oszillatoren durch D/A-gewandelte Signalformen. DDS-ICs (zB. AD9830, AD9850, AD9952) speichern dazu Samples in fixem ROM.

28 z-Transformation

kurz *'z-TRF'*

Die z-Transformation wird größtenteils für die digitale Steuer- und Regelungstechnik und zur Berechnung digitaler Filter angewendet. Man kann sie aber auch zur Gewinnung von expliziten Formeln für rekursiv definierte Zahlenfolgen einsetzen.

(aus https://de.wikipedia.org/wiki/Z-Transformation, 01Nov'22)

Um mathematische Methoden und Erkenntnisse anwenden zu können,
ist der Kas (leider) mathematisch zu formulieren (XH)

28.0.1 Vorwort (XH)

a) `Euler'sche Identität:`

$\boxed{e^{jx} = cos(x) + j \cdot sin(x)}$

(ganz leicht durch Taylor-Reihenentwicklung zu zeigen:

$e^{x} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \dots$

$e^{jx} = 1 + jx + \frac{(jx)^2}{2!} + \frac{(jx)^3}{3!} \dots$

$= 1 + jx - \frac{x^2}{2!} - j\frac{x^3}{3!} \dots$

$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$

$jsin(x) = jx - j\frac{x^3}{3!} + j\frac{x^5}{5!} - \dots$

$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots)$

b) weiters sieht man öfter:

$\frac{e^{jx} + e^{-jx}}{2} = cos(x)$

$(= \frac{cos(x)+jsin(x)+cos(x)+(-j)sin(x)}{2} = 2cos(x)/2)$

$\frac{e^{jx} - e^{-jx}}{2} = jsin(x)$

$(= \frac{cos(x)+jsin(x)-cos(x)-(-j)sin(x)}{2} = 2jsin(x)/2)$

- Da e^{jx} als komplexe Zahl in der komplexen Zahlenebene als Vektor dargestellt wird, reden *manche* von einem `Zeiger`.
- In diesem e^{-jx} sehen *dieselben manchen* dann eine `negative Frequenz`. (Wenn Du (hoffentlich) nicht zu diesen *Manchen* gehörst, überlies solch esoterischen Schaum bitte einfach!)
- auch müssen sie der mathemtischen Integrität zuliebe das *Sampling* als Multiplikation der Samplewerte mit der →Dirac-Funktion δ (unendlich schmal, unendlich hoch, Fläche=1) darstellen. Dadurch wird aus der Laplace-Transformation die *z-Transformation* mit diesen z^{-1}, z^{-2} usw. (das 'z' könnte von seinem Erfinder *Zadeh* stammen?

---

gegeben:

$\mathcal{X}$(z)	...	Signal im Bildbereich
z	...	z-Variable; komplexe Zahl $\boxed{z = Ae^{j\omega}}$
k	...	Abtastfolge = Nummerierung eines Folgegliedes
n	...	Zahl bestimmter Abtastschritte
T_A	...	sampling-Intervall $T_A = \frac{1}{samplingrate'sr'}$
N	...	Anzahl von Sample-Werten
x	...	Folge (array) von samples (Abtast-Werte)
		x[0] ...Abtast-Wert zum Zeitpunkt t=0*T_A
		x[1] ...Abtast-Wert zum Zeitpunkt t=1*T_A
		... ...
		x[N-1]...Abtast-Wert zum Zeitpunkt t=(N-1)*T_A

Wenn ausser zu den Zeitpunkten
t= 0, t=1·T_A, t=2·T_A, 3·T_A, ... , N·T_A
ein Signal f(t) immer **Null** ist,

→ dann →

ist das so, als hätte man eine kontinuierliche Funktion mit einer Reihe von
→Dirac-Impulsen
zu den Zeitpunkten t= 0, 1·T_A, 2·T_A, 3·T_A, ... , N·T_A multipliziert.

Das Laplace-Integral

$$\mathcal{L}(s) = \int_{t=0}^{\infty} f(t) \cdot e^{-sT_A} \cdot dt$$

ist jetzt überall Null, – ausser
zu diesen Abtastzeitpunkten $0 \cdot T_A, 1 \cdot T_A, 2 \cdot T_A, ...$
↓
und kann folglich durch eine Summe

$$\mathcal{X}(k) = \sum_{n=0}^{N} x[n] \cdot e^{-\frac{j \cdot 2\pi \cdot n \cdot k \cdot T_A}{N}}$$

ersetzt werden.

Formal ersetzt man zur optischen Vereinfachung weiters

$$z ::= e^{sT_A}$$

$$\rightarrow$$

$$z^{-1} = \frac{1}{e^{sT_A}} = (e^{sT_A})^{-1} = e^{-1 \cdot sT_A} = e^{-sT_A}$$

$$z^{-2} = \frac{1}{(e^{sT_A})^2} = (e^{sT_A})^{-2} = e^{-2 \cdot sT_A}$$

$$...$$

$$z^{-n} = \frac{1}{(e^{sT_A})^n} = (e^{sT_A})^{-n} = e^{-n \cdot sT_A}$$

und nennt

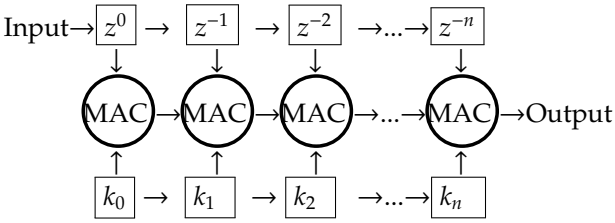
$$\mathcal{X}(z) = \sum_{n=0}^{\infty} x[n] \cdot z^{-n}$$

die **z- Transformierte** von x.
Man zeichnet wie Schieberegister

Input→$\boxed{z^0}$→$\boxed{z^{-1}}$→$\boxed{z^{-2}}$→...→$\boxed{z^{-n}}$→

und stellt sich vor, dass die Werte im Schieberegister sich zeitlich jeweils um ein $\Delta t = T_A$ unterscheiden.

Die Struktur



entspricht obiger Summe.
Neu ist lediglich die Beschriftung mit diesen 'z' und die z-Transformierten als Polynome:
$X(z) = k_0 \cdot z + k_1 \cdot z^{-1} + k_2 \cdot z^{-2} \ldots k_n \cdots z^{-n}$

Dazu bekommt man zu lesen:

Der z-Bereich kann nicht so einfach physikalisch interpretiert werden wie der Frequenzbereich, hat aber viele Vorteile aus mathematischer Sicht.

Während der Einheitsimpuls und ein sinusförmiges Eingangsignal (in der mathematischen Darstellung der komplexen Exponentialfunktion) einen Bezug zur physikalischen Welt haben, ist das beim Eingangssignal zn nicht der Fall. In Anlehnung an die Bezeichnungen Zeitbereich und Frequenzbereich, nennen wir die Darstellung mit z den z-Bereich.

Der z-Bereich ist eine abstrakte mathematische Welt, die eine Reihe von Eigenschaften hat, die bei der Untersuchung von Systemeigenschaften sehr hilfreich sind:

-Durch die z Transformation werden aus Differenzengleichungen gebrochen rationale Funktionen, es werden damit Polynome für die Systembeschreibung eingeführt. Polynome sind "gutartige" Funktionen, mit denen sich leicht rechnen lässt. Algebraische Operationen wie Division, Multiplikation und Faktorisierung von Polynomen entsprechen dem Zerlegen bzw. Zusammensetzen von LTI Systemen.

-Die rechnerisch aufwendige Operation der Faltung geht über in die Multiplikation.

-Gebrochen rationale Funktionen bestehen aus einem Zähler- und einem Nennerpolynom. Aus der Lage der Wurzeln (Nullstellen) dieser Polynome lassen sich Eigenschaften digitaler Systeme ableiten, insbesondere sind dadurch Aussagen über die Stabilität möglich.

Die z-Transformation hat für diskrete Systeme dieselbe Bedeutung wie die Laplace-Transformation für kontinuierliche Systeme.

[...]

Der z Bereich ist eine abstrakte Welt, die durch Pole und Nullstellen beschrieben und von der Systemfunktion $H(z) = Y(z)$ regiert wird. Im z Bereich $X(z)$ wird $x[n] = z^n$ als Eingangsgröße gewählt, wobei z eine beliebige komplexe Zahl und daher auch $z = e^{j\omega}$ sein kann. Wir wir sehen werden "liegt" der Frequenzbereich daher im z Bereich und ist ein Sonderfall des z Bereichs. Dennoch wird zwischen den beiden Bereichen unterschieden, da sie unterschiedlichen Charakter haben.

[...]

Die Systemfunktion H(z) ist die z-Transformierte der Impulsantwort h[n].

[...]

Die Systemfunktion eines digitalen Systems ist die z-Transformierte seiner Impulsantwort.

[...]

Eigenschaften der z-Transformation:

Die z Transformation ist eine lineare Transformation

Die Multiplikation einer Folge mit z^{-1} verschiebt die Folge um einen Schritt

$x[n-1] \Leftrightarrow z^{-1}X(z)$

Für eine Verzögerung um r Punkte erhalten wir

$x[n-r] \Leftrightarrow z^{-r}X(z)$

[...]

Wie wir wissen, sind für die Realisierung von digitalen Filtern Verzögerungsglieder (Speicher) erforderlich.

(die meinen damit die Schieberegister;
z^{-1} ist ein *'Verzoegerungselement'*)

In signal processing, cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. This is also known as a **sliding dot product or sliding inner-product**. It is commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology. The cross-correlation is similar in nature to the **convolution of two functions**. In an autocorrelation, which is the cross-correlation of a signal with itself, there will always be a peak at a lag of zero, and its size will be the signal energy.

(aus https://en.wikipedia.org/wiki/Cross-correlation)

In der Signalanalyse wird die **Kreuzkorrelationsfunktion** $R_{xy}(\tau)$ zur Beschreibung der Korrelation zweier Signale $x(t)$ und $y(t)$ bei unterschiedlichen Zeitverschiebungen τ zwischen den beiden Signalen eingesetzt. Kreuz steht hierbei für den Fall $x \neq y$ der Funktion:

$R_{xy}(t_1, t_2) = E\{\mathbf{X}(t_1) \cdot \mathbf{Y}(t_2)\}$

Handelt es sich um einen schwach stationären Prozess, so ist die Korrelationsfunktion nicht mehr von der Wahl der Zeitpunkte t_1 und t_2, sondern nur von deren Differenz $\tau = t_2 - t_1$ abhängig.

Die Kreuzkorrelations-Operation ist identisch mit der komplex konjugierten Faltung $\overline{f(-t)}$ (s. en:Cross-correlation#Properties). Insbesondere im Fachgebiet Maschinelles Lernen, wo man mit Convolutional Neural Networks arbeitet, wird aufgrund dieser Identität meistens die Kreuzkorrelation verwendet, diese aber als Faltung bezeichnet, weil sie leichter zu implementieren ist.

(aus https://de.wikipedia.org/wiki/Kreuzkorrelation)

Anwendung:
- Lösen von Differenzengleichungen
- Vereinfachung digitaler Filter durch Zerlegung
- Prüfsummensysteme
- Kryptografie

29 Datenformate

29.1 IEEE-754 floating point numbers

IEEE-754-1985
IEEE-754-2008
IEEE-754-2019
Die Norm IEEE 754 (ANSI/IEEE Std 754-1985; IEC-60559:1989 - International version) definiert Standarddarstellungen für binäre Gleitkommazahlen in Computern und legt genaue Verfahren für die Durchführung mathematischer Operationen, insbesondere für Rundungen, fest. Der genaue Name der Norm ist englisch *IEEE Standard for Binary Floating-Point Arithmetic for microprocessor systems* (ANSI / IEEE Std 754-1985).

Die aktuelle Ausgabe ist unter der Bezeichnung **ANSI** / **IEEE Std 754-2019 im Juli 2019** veröffentlicht worden. Weiter ist die Norm **IEEE 854-1987**, mit dem engl. Titel Standard for radix-independent floating-point arithmetic, in der **IEEE 754-2008 vollständig integriert** worden.

[...]

Überblick
In der Norm IEEE 754-1989 werden zwei Grunddatenformate für binäre Gleitkommazahlen mit 32 Bit (single precision) bzw. 64 Bit (double precision) Speicherbedarf und zwei erweiterte Formate definiert. Die IEEE 754-2008 umfasst die binären Zahlenformate mit 16 Bit als Minifloat, 32 Bit als single, 64 Bit als double und neu 128 Bit (quadruple). Zusätzlich kamen noch die dezimalen Darstellungen mit 32 Bit als Minifloat, 64 und 128 Bit hinzu.

Schließlich gab es Vorschläge und Implementierungen von weiteren Zahlenformaten, die nach den Prinzipien der IEEE 754-1989 Norm gestaltet sind und deshalb oft als IEEE-Zahlen bezeichnet werden, obwohl sie das nach der alten Definition streng genommen nicht sind. Dazu gehören die in den neuen Ausgaben integrierten Minifloats, die für die Ausbildung gedacht sind. Minifloats mit 16 Bit werden gelegentlich in der Grafikprogrammierung verwendet (auch von 11-Bit-FP hatma schon gheart, Anm:XH). Ebenso gehören auch mehrere nicht von IEEE 754-1989 definierte Zahlenformate mit mehr als 64 Bit, etwa das 80-Bit-Format (Extended Precision Layout...), welches die IA-32-Prozessoren intern in ihrer klassischen Gleitkommaeinheit (Floating Point Unit, FPU) verwenden, dazu.

Zahlenformate und andere Festlegungen des IEEE-754-Standards
IEEE 754 unterscheidet vier Darstellungen: einfach genaue (single), erweiterte einfach genaue (single extended), doppelt genaue (double) und erweiterte doppelt genaue (double extended) Zahlenformate. Bei den erweiterten Formaten ist nur jeweils eine Mindestbitzahl vorgeschrieben. Die genaue Bitzahl und der Biaswert bleiben dem Implementierer überlassen. Die Grundformate sind vollständig definiert.

Vor allem die Anzahl der Exponentenbits legt Maximum und Minimum der darstellbaren Zahlen fest. Die Anzahl der Mantissenbits bestimmt die (relative s. u.) Genauigkeit dieser Zahlen (und nur in geringem Maß das Maximum und das Minimum).

(aus https://de.wikipedia.org/wiki/IEEE_754, 22Aug'20)

*[...] Die Norm **IEEE 754** (ANSI/IEEE Std 754-1985; IEC-60559:1989 - International version)*
*definiert **Standarddarstellungen für binäre Gleitkommazahlen in Computern** und legt genaue Verfahren für die Durchführung mathematischer Operationen, insbesondere für Rundungen, fest. Der genaue Name der Norm ist englisch IEEE Standard for Binary Floating-Point Arithmetic for microprocessor systems (ANSI/IEEE Std 754-1985). Die aktuelle Ausgabe ist unter der Bezeichnung ANSI/IEEE Std 754-2008 im August 2008 veröffentlicht worden und umfasst neben der 754-1985 eine Erweiterung um zusätzlich ein binäres und zwei dezimale Datenformate. Weiter ist die Norm IEEE 854-1987, mit dem Titel englisch Standard for radix-independent floating-point arithmetic, in der IEEE 754-2008 vollständig integriert worden.*
Überblick
In der Norm IEEE 754-1989 werden zwei Grunddatenformate für binäre Gleitkommazahlen mit 32 Bit (single precision) bzw. 64 Bit (double precision) Speicherbedarf und zwei erweiterte Formate definiert. Die IEEE 754-2008 umfasst die binäre Zahlenformate mit 16 Bit als Minifloat, 32 Bit als single, 64 Bit als double und neu 128 Bit. Zusätzlich kamen noch die dezimale Darstellungen mit 32 Bit als Minifloat, 64 und 128 Bit hinzu. Schließlich gab es Vorschläge und Implementierungen von weiteren Zahlenformaten, die nach den Prinzipien der IEEE 754-1989 Norm gestaltet sind und deshalb oft als IEEE-Zahlen bezeichnet werden, obwohl sie das nach der alten Definition streng genommen nicht sind. Dazu gehören die in den neuen Ausgaben integrierten Minifloats, die für die Ausbildung gedacht sind. Minifloats mit 16 Bit werden gelegentlich in der Grafikprogrammierung verwendet. Dazu gehören auch mehrere nicht von IEEE 754-1989 definierte Zahlenformate mit mehr als 64 Bit, etwa das 80-Bit-Format (Extended Precision Layout...), welches die IA-32-Prozessoren intern in ihrer klassischen Gleitkommaeinheit (Floating Point Unit, FPU) verwenden.
[...]
Null
Null repräsentiert die absolute Null. Auch Zahlen, die zu klein sind, um dargestellt zu werden (Unterlauf), werden auf

Null gerundet. Ihr Vorzeichen bleibt dabei erhalten. Negative kleine Zahlen werden so zu −0,0 gerundet, positive Zahlen zu +0,0. Beim direkten Vergleich werden jedoch +0,0 und −0,0 als gleich angesehen.

Normalisierte Zahl

Die Mantisse besteht aus den ersten n wesentlichen Ziffern der Binärdarstellung der noch nicht normalisierten Zahl. Die erste wesentliche Ziffer ist die höchstwertige (d. h. am weitesten links stehende) Ziffer, die von 0 verschieden ist. Da eine von 0 verschiedene Ziffer im Binärsystem nur eine 1 sein kann, muss diese erste 1 nicht explizit abgespeichert werden; gemäß der Norm IEEE 754 werden nur die folgenden Ziffern gespeichert, die erste Ziffer ist eine implizite Ziffer oder ein implizites Bit (englisch „hidden bit"). Dadurch wird gewissermaßen 1 Bit Speicherplatz "gespart".

[...]

Unendlich

Der Gleitkommawert "Unendlich" repräsentiert Zahlen, deren Betrag zu groß ist, um dargestellt zu werden. Es wird zwischen +"Unendlich" und −"Unendlich" unterschieden. Die Berechnung von 1,0/0,0 ergibt per Definition ebenfalls +"Unendlich".

Keine Zahl (NaN)

Damit werden ungültige (oder nicht definierte) Ergebnisse dargestellt, z. B. wenn versucht wurde, die Quadratwurzel aus einer negativen Zahl zu berechnen. Einige „unbestimmte Ausdrücke" haben als Ergebnis „keine Zahl", zum Beispiel 0,0/0,0 oder „Unendlich" − „Unendlich". Außerdem werden NaNs in verschiedenen Anwendungsbereichen benutzt, um „Kein Wert" oder „Unbekannter Wert" darzustellen. Insbesondere der Wert mit dem Bitmuster 111...111 wird oft für eine „nicht initialisierte Gleitkommazahl" benutzt. IEEE 754 fordert zwei Arten von Nichtzahlen: stille NaN (NaNq – quiet) und signalisierende NaN (NaNs – signalling). Beide stellen explizit keine Zahlen dar. Eine signalisierende NaN löst im Gegensatz zu einer stillen NaN eine Ausnahme (Trap) aus, wenn sie als Operand einer arithmetischen Operation auftritt. IEEE 754 ermöglicht dem Anwender das Deaktivieren dieser Traps. In diesem Falle werden signalisierende NaN wie stille NaN behandelt. Signalisierende NaN können genutzt werden, um uninitialisierten Rechnerspeicher zu füllen, so dass jedes Verwenden einer uninitialisierten Variable automatisch eine Ausnahme auslöst. Stille NaN ermöglichen den Umgang mit Rechnungen, die kein Ergebnis erzeugen können, etwa weil sie für die angegebenen Operanden nicht definiert sind. Beispiele sind die Division Null durch Null oder der Logarithmus aus einer negativen Zahl. Stille und Signalisierende NaN unterscheiden sich im höchsten Mantissenbit. Bei stillen NaN ist dieses 1, bei signalisierenden NaN 0. Die übrigen Mantissenbits können zusätzliche Informationen enthalten, z. B. die Ursache der NaN. Dies kann bei der Ausnahmebehandlung hilfreich sein. Allerdings schreibt der Standard nicht fest, welche Informationen in den übrigen Mantissenbits enthalten sind. Die Auswertung dieser Bits ist daher plattformabhängig. Das Vorzeichenbit hat bei NaN keine Bedeutung. Es ist nicht spezifiziert, welchen Wert das Vorzeichenbit bei zurückgegebenen NaN besitzt.

Rundungen

IEEE 754 unterscheidet zunächst zwischen binären Rundungen und binär-dezimalen Rundungen, bei denen geringere Qualitätsforderungen gelten. Bei binären Rundungen muss zur nächstgelegenen darstellbaren Zahl gerundet werden. Wenn diese nicht eindeutig definiert ist (genau in der Mitte zwischen zwei darstellbaren Zahlen), wird so gerundet, dass das niederwertigste Bit der Mantisse 0 wird. Dies passiert statistisch in 50 % der Fälle, so dass der von Knuth beschriebene statistische Drift in längeren Rechnungen vermieden wird. Eine zu IEEE 754 konforme Implementierung muss drei weitere vom Programmierer einstellbare Rundungen bereitstellen: Rundung gegen +Unendlich (immer aufrunden), Rundung gegen −Unendlich (immer abrunden) und Rundung gegen 0 (Ergebnis immer betragsmäßig verkleinern).

Exceptions, Flags und Traps

Treten bei der Berechnung Ausnahmen (Exceptions) auf, werden Status-Flags gesetzt. Im Standard wird vorgeschrieben, dass der Benutzer diese Flags lesen und schreiben kann. Die Flags sind „sticky": werden sie einmal gesetzt, bleiben sie so lange erhalten, bis sie explizit wieder zurückgesetzt werden. Das Überprüfen der Flags ist beispielsweise die einzige Möglichkeit, 1/0 (=Unendlich) von einem Überlauf zu unterscheiden. Des Weiteren wird im Standard empfohlen, Trap Handler zu ermöglichen: Tritt eine Ausnahme auf, wird der Trap Handler aufgerufen, anstatt das Status-Flag zu setzen. Es liegt in der Verantwortung solcher Trap Handler, das entsprechende Status-Flag zu setzen oder zu löschen. Ausnahmen werden im Standard in 5 Kategorien eingeteilt: Überlauf, Unterlauf, Division durch Null, ungültige Operation und Ungenau. Für jede Klasse steht ein Status-Flag zur Verfügung.

Geschichte

In der 1960er und frühen 1970er Jahren hatte jeder Prozessor sein eigenes Format für Gleitkommazahlen und seine eigene FPU oder Gleitkommasoftware, mit der das jeweilige Format verarbeitet wurde. Dasselbe Programm konnte auf verschiedenen Rechnern unterschiedliche Resultate liefern. Die Qualität der verschiedenen Gleitkommaarithmetiken war ebenfalls sehr unterschiedlich. Intel plante um 1976 für seine Mikroprozessoren eine eigene FPU und wollte die bestmögliche Lösung für die zu implementierende Arithmetik. Unter der Federführung der IEEE begannen 1977 Treffen, um FPUs für Gleitkommaarithmetik für Mikroprozessoren zu normieren. Die erste Version der Norm wurde 1985 verabschiedet und 2008 erweitert.

[...]

(aus http://de.wikipedia.org/w/index.php?oldid=114932354, 22.Mar'13)

Allgemeines

Die Darstellung einer Gleitkommazahl

$$x = s \cdot m \cdot b^e$$

besteht aus:

- Vorzeichen *s* (fast ausnahmslos 1 Bit)
- Basis *b* (bei normalisierten Gleitkommazahlen nach IEEE 754 ist $b = 2$)
- Exponent *e* (r Bits), nicht zu verwechseln mit dem „biased exponent“ bzw. der Charakteristik
- Mantisse *m* (p Bits), manchmal als Signifikant bezeichnet

Das Vorzeichen $s = (-1)^S$ wird in einem Bit S gespeichert, sodass S = 0 positive Zahlen und S = 1 negative Zahlen markiert.
Der Exponent *e* ergibt sich aus der nichtnegativen Binärzahl *E* (*E* wird manchmal auch als Charakteristik oder biased exponent bezeichnet) durch Subtraktion eines festen Biaswertes *B*: $e = E - B$. Der Biaswert (engl: Verzerrung) berechnet sich durch $2^{r-1} - 1$, wobei *r* die Anzahl der Bits von *E* darstellt. Der Biaswert *B* dient also dazu, dass negative Exponenten durch eine vorzeichenlose Zahl (die Charakteristik *E*) gespeichert werden können, unter Verzicht auf alternative Kodierungen wie z. B. das Zweierkomplement. (vergleiche auch Exzesscode)
Schließlich ist die Mantisse $1 \leq m < 2$ ein Wert, der sich aus den p Mantissenbits mit dem Wert M als $m = 1 + M/2^p$ berechnet. Einfacher ausgedrückt denkt man sich an das Mantissenbitmuster M links eine "1," angehängt: m = 1,M.

- $s = (-1)^S$
- $e = E - B$

- $m = 1, M = 1 + M/2^p$

Dieses Verfahren ist möglich, weil durch Normalisierung (s. u.) die Bedingung $1 \leq m < 2$für alle darstellbaren Zahlen immer eingehalten werden kann. Da dann die Mantisse immer links mit 1. beginnt, braucht dieses Bit nicht mehr gespeichert zu werden. Damit gewinnt man ein zusätzliches Bit Genauigkeit.

Für Sonderfälle stehen spezielle Bitmuster zur Verfügung. Um diese Sonderfälle zu kodieren, sind zwei Exponentenwerte, der maximale ($E = 11\ldots111 = 2^r - 1$) und die Null ($E = 00\ldots000$) reserviert. Mit dem maximalen Exponentenwert werden die Sonderfälle NaN und ∞ kodiert. Mit Null im Exponenten wird die Gleitkommazahl 0 und alle denormalisierten Werte kodiert.

Werte außerhalb des normalen Wertebereichs (zu große bzw. zu kleine Zahlen) werden durch ∞ bzw. −∞ dargestellt. Diese Erweiterung des Wertebereichs erlaubt auch im Falle eines arithmetischen Überlaufs häufig ein sinnvolles Weiterrechnen. Neben der Zahl 0 existiert noch der Wert −0. Während $\frac{1}{0}$das Ergebnis ∞ liefert, ergibt $\frac{1}{-0}$den Wert −∞. Bei Vergleichen wird zwischen 0 und −0 nicht unterschieden.

Die Werte NaN (für engl. „not a number“, „keine Zahl“) werden als Darstellung für undefinierte Werte verwendet. Sie treten z. B. auf als Ergebnisse von Operationen wie $\frac{0}{0}$oder ∞ −∞. NaN werden in Signal-NaN (signalling NaN, NaNs) für Ausnahmebedingungen und stille NaN (quiet NaN, NaNq) unterteilt.

Als letzter Sonderfall füllen denormalisierte Zahlen (in IEEE 754r als subnormale Zahlen bezeichnet) den Bereich zwischen der betragsmäßig kleinsten normalisierten Gleitkommazahl und Null. Sie werden als Festkommazahlen gespeichert und weisen nicht dieselbe Genauigkeit auf wie die normalisierten Zahlen. Konstruktionsbedingt haben die meisten dieser Werte den Kehrwert ∞.

Zahlenformate und andere Festlegungen des IEEE-754-Standards

IEEE 754 unterscheidet vier Darstellungen: einfach genaue (single), erweiterte einfach genaue (single extended), doppelt genaue (double) und erweiterte doppelt genaue (double extended) Zahlenformate. Bei den erweiterten Formaten ist nur jeweils eine Mindestbitzahl vorgeschrieben. Die genaue Bitzahl und der Biaswert bleiben dem Implementierer überlassen. Die Grundformate sind vollständig definiert.

Die Anzahl der Exponentenbits legt den Wertebereich der darstellbaren Zahlen fest (s. u.). Die Anzahl der Mantissenbits legt die Genauigkeit dieser Zahlen fest.

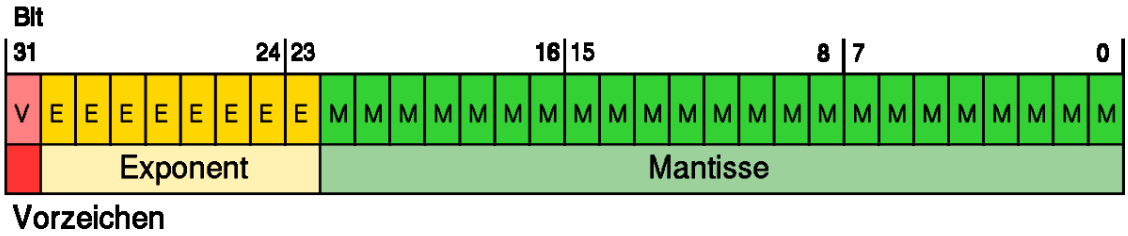
Die beiden letzten Beispiele demonstrieren ein minimales erweitertes Format.

Typ	Größe (1+r+p)	Exponent (r)	Mantisse (p)	Werte des Exponenten (e)	Biaswert (B)
single	32 bit	8 bit	23 bit	-126 ≤ *e* ≤ 127	127
double	64 bit	11 bit	52 bit	-1022 ≤ *e* ≤ 1023	1023
single extended	≥ 43 bit	≥ 11 bit	≥ 31 bit		
double extended	≥ 79 bit	≥ 15 bit	≥ 63 bit		
single extended, minimum	43 bit	11 bit	31 bit	-1022 ≤ *e* ≤ 1023	1023
double extended, minimum	79 bit	15 bit	63 bit	-16382 ≤ *e* ≤ 16383	16383

Für die angegebenen Formate ergibt sich die folgende Beschränkung des jeweiligen Zahlenbereichs. Die betragsmäßig kleinsten Zahlen sind hierbei nicht normalisiert. Der relative Abstand zweier Gleitkommazahlen ist größer als ϵ und kleiner gleich 2 $\cdot\epsilon$. Konkret ist der Abstand (und in diesem Fall auch der relative Abstand) der Gleitkommazahl 1 zur nächstgrößeren Gleitpunktzahl gleich 2 $\cdot\epsilon$. Dezimalstellen beschreibt die Anzahl der Stellen einer Dezimalzahl die ohne Genauigkeitsverlust gespeichert werden können. Die Mantisse ist rechnerisch durch das implizite Bit um eins größer als gespeichert.

Typ	ϵ	Dezimal-stellen	(betragsmäßig) kleinste Zahl	Größte Zahl
single	$2^{-(23+1)}$ $\approx 5{,}960 \cdot 10^{-8}$	7 … 8	$2^{-127} = 5{,}877 \cdot 10^{-39}$	$(1-2^{-24}) \times 2^{128}$ $\approx 3{,}403 \cdot 10^{38}$
double	$2^{-(52+1)}$ $\approx 1{,}110 \cdot 10^{-16}$	15 … 16	$2^{-1023} = 1{,}1125 \cdot 10^{-308}$	$(1-2^{-53}) \times 2^{1024}$ $\approx 1{,}798 \cdot 10^{308}$
single extended, minimum	$2^{-(31+1)}$ $\approx 2{,}328 \cdot 10^{-10}$	9 … 10	$2^{-31} \times 2^{-1022} = 2^{-1053}$ $\approx 1 \cdot 10^{-317}$	$(1-2^{-32}) \times 2^{1024}$ $\approx 1{,}798 \cdot 10^{308}$
double extended, minimum	$2^{-(63+1)}$ $\approx 5{,}521 \cdot 10^{-20}$	19 … 20	$2^{-63} \times 2^{-16382} =$ 2^{-16445} $\approx 4 \cdot 10^{-4951}$	$(1-2^{-64}) \times$ 2^{16384} $\approx 1{,}190 \cdot 10^{4932}$

Die Anordnung der Bits einer *single* zeigt die nachfolgende Abbildung. Die bei einer Rechenanlage konkrete Anordnung der Bits im Speicher kann von diesem Bild abweichen und hängt von der jeweiligen Bytereihenfolge (little/big endian) und weiteren Rechnereigenheiten ab.



Die Anordnung mit *Vorzeichen – Exponent – Mantisse* in genau dieser Reihenfolge bringt (innerhalb eines Vorzeichenbereiches) die dargestellten Gleitkommawerte in dieselbe Reihenfolge wie die durch dasselbe Bitmuster darstellbaren Signed-Integer-Werte. Damit können für die Vergleiche von Gleitkommazahlen dieselben Operationen wie für die Vergleiche von Signed-Integers verwendet werden. Kurz: die Gleitkommazahlen können lexikalisch sortiert werden.

Hierbei ist jedoch zu beachten, dass für steigende negative Signed-Integer-Werte der entsprechende Fließkommawert gegen minus unendlich geht, die Sortierung also umgekehrt ist.

Auch wenn in diesem Artikel hauptsächlich das Zahlenformat erörtert wird, liegt die Bedeutung der Norm IEEE 754 auch darin, dass für Gleitkommazahlen genaue Vorschriften für

- Rundung
- arithmetische Operationen
- Wurzelberechnung
- Konversionen
- Ausnahmebehandlung (Exception handling)

festgelegt wurden.

Beispiele

Berechnung Dezimalzahl → IEEE754-Gleitkommazahl

Die Zahl $18{,}4_{10}$ soll in eine Gleitkommazahl umgewandelt werden, dabei nutzen wir den Single IEEE-Standard.

1. Umwandlung der Dezimalzahl in eine duale Festkommazahl ohne Vorzeichen

```
18,4

18/2 = 9 Rest 0 (Least-Significant Bit)
 9/2 = 4 Rest 1
 4/2 = 2 Rest 0
 2/2 = 1 Rest 0
 1/2 = 0 Rest 1 (Most-Significant-Bit)
     = 10010

0,4*2 = 0,8 -0 (Most-Significant-Bit)
0,8*2 = 1,6 -1
0,6*2 = 1,2 -1
0,2*2 = 0,4 -0
0,4*2 = 0,8 -0
0,8*2 = 1,6 -1 (Least-Significant-Bit)
      *
      *
      *
      = 0,0110011001100110011...

18,4  = 10010,0110011001100110011...
```

2. Normalisieren und Bestimmen des Exponenten

```
Bias = 1x '0' + (r-1)x '1', für r=8: 01111111
```

```
10010,0110011001100... * 2^(01111111-01111111) =
1001,0011001100... * 2^(10000000-01111111) =
100,10011001100... * 2^(10000001-01111111) =
10,0100110011001100... * 2^(10000010-01111111) =
1,0010011001100... * 2^(10000011-01111111)

Mantisse: 1,0010011001100...
Exponent mit Bias: 10000011
```

3. Vorzeichen-Bit bestimmen

```
positiv → 0
```

4. Die Gleitkommazahl bilden

```
1 Bit Vorzeichen + 8 Bit Exponent + 23 Bit Mantisse
0 10000011 00100110011001100110011 → die Vorkomma-Eins wird als Hidden Bit weggelassen;
                                      da dort immer eine 1 steht, braucht man diese nicht zu speichern.
```

Berechnung IEEE754-Gleitkommazahl → Dezimalzahl

Nun soll die Gleitkommazahl von oben wieder in eine Dezimalzahl zurück gewandelt werden, gegeben ist also folgende IEEE754-Zahl:

```
0 10000011 00100110011001100110011
```

1. Berechnung des Exponenten

```
Umwandeln des Exponenten in eine Dezimalzahl:
10000011 -> 131

Da dies aber der Biased Exponent ist, der zuvor um den Bias verschoben wurde, wird nun der Bias
wieder abgezogen:
131-127 = 4 ist also der Exponent
```

2. Berechnung der Mantisse

```
Da es sich um eine normalisierte Zahl handelt, wissen wir, dass sie eine 1 vor dem Komma hat:
1,00100110011001100110011

Nun muss das Komma um 4 Stellen nach rechts verschoben werden:
10010,0110011001100110011
```

3. Umwandlung in eine Dezimalzahl

```
Vorkommastellen:
10010 (2) = 18 (10)

Nachkommastellen:
0,0110011001100110011 (2) ≈ 0.39999961853 (10)
(Um den Wert der Nachkommazahl zu erhalten, muss man denselben Prozess durchführen wie bei ganzen Zahlen,
nur in umgekehrter Richtung. Also von links nach rechts. Dabei muss der Exponent negativ sein und
mit einer 1 beginnen.
In der Form 0*2^-1+1*2^-2+1*2^-3+0*2^-4+0*2^-5+1*2^-6+1*2^-7+0*2^-8+0*2^-9+1*2^-10+1*2^-11+0*2^-
12+0*2^-13+1*2^-14+1*2^-15+0*2^-16+0*2^-17+1*2^-18+1*2^-19)
Weil die 0,4 im binären eine periodische Darstellung hat, kann sie nicht mehr genau umgewandelt
werden
```

4. Vorzeichen

```
Vorzeichenbit ist eine 0, also ist es eine positive Zahl
```

5. Dezimalzahl "zusammensetzen"

```
18,39999961853
```

30 Datenkompression

30.1 Codierung

Ein *Code* ist **keine** *Chiffre (cypher)*!
Code hat **nichts mit 'geheim'** zu tun.

- ASCII-Code ist ein Code
- Unicode ist ein Code
- Barcode ist ein Code
- Morsecode ist ein Code
- Programmcode zB. 'C', 'Python', 'SQL' ist Code
- Prozessor-Maschinencode ist Code
- römische Zahlen sind ein Code

30.2 Kryptologie

- Kryptologie = Wissenschaft der Geheimchiffrierung
  von altgriechisch *'krypto'* ≡ 'verborgen, versteckt'
- Chiffrierung = Encryption, Cypher
- verschlüsseln = chiffrieren, encrypt, encypher
- entschlüsseln = dechiffrieren, decrypt, decypher
- Chiffre, Cypher = der verschlüsselte Text (Message)
- Chiffrierung = Methode + Schlüssel (Key)
- das Geheimnis soll im Schlüssel und nicht in der Methode liegen
- Primzahlen + XOR-Verknüpfung → "Scrambler"
- Grobunterscheidung
  - symmetrische Chiffrierung: EncryptionKey = DecryptionKey
  - public key systems: publicKey + privateKey, unsymmetrisch

$$\text{msg} \rightarrow E_{msg} \rightarrow \text{cypher} \rightarrow D_{cypher} \rightarrow \text{msg}$$

- 'Hash': Decryption unmöglich - Divisionsreste. zB. MD5, SHA-1, -2, -3

30.3 Informationstheorie

nach C.E.Shannon und H.Nyquist

- Daten-Nachricht-Information
  1 **Daten** sind Attribut-Werte (Merkmals-Ausprägungen)
  2 **Nachrichten** sind transportierte Daten (Sender,Empfänger)
  3 **Information** ist eine Nachricht mit **Informationsgehalt**
  Redundanz ::= informationslose Nachricht
  Information + Redundanz ≥ Nachricht
- → https://de.wikipedia.org/wiki/Informationsgehalt

  Informationsgehalt (nach C.E.Shannon und H.Nyquist):

  $$\texttt{I(N) = - ld(P(N))}$$

  mit:

```
N ..... Nachricht
I(N) ... Informationsgehalt der Nachricht N
         Maßeinheit: "Shannon" [sh] - (frueher [bit])
P(N) ... Wahrscheinlichkeit der Nachricht N
ld ..... "logarithmus dualis":
         ld(x) = log2(x) = 3,32*log10(x)
```

  $\texttt{(3,32 =}\ \frac{1}{log_{10}(2)} = \frac{ln(10)}{ln(2)}\ \texttt{)}$

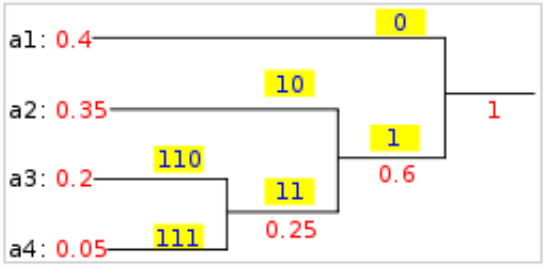
- optimale Binärcodierung:
  Nachrichtenlänge in [Bit] = Informationsgehalt in [sh]

30.4 Huffman tree

zum Erzeugen optimaler Binärcodierungen: sei zB. gegeben:
a1 .. a4 : Nachrichten mit ihren relativen Häufigkeiten
Verfahren:

1 sortieren nach Häufigkeit
2 seltenste zwei zusammenfassen
3 verschmelzen und Häufigkeiten addieren
4 wiederhole ab (1) bis fertig

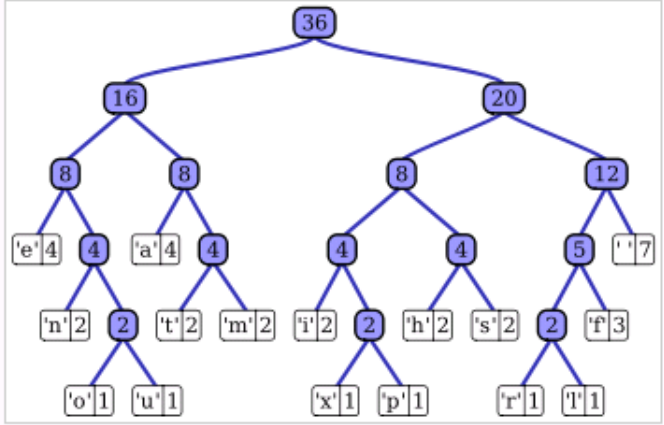
→ *http://en.wikipedia.org/wiki/Huffman_tree*



A source generates 4 different symbols $\{a_1, a_2, a_3, a_4\}$ with probability $\{0.4; 0.35; 0.2; 0.05\}$. A binary tree is generated from left to right taking the two less probable symbols, putting them together to form another equivalent symbol having a probability that equals the sum of the two symbols. The process is repeated until there is just one symbol. The tree can then be read backwards, from right to left, assigning different bits to different branches. The final Huffman code is:

Symbol	Code
a1	0
a2	10
a3	110
a4	111

The standard way to represent a signal made of 4 symbols is by using 2 bits/symbol, but the entropy of the source is 1.73 bits/symbol. If this Huffman code is used to represent the signal, then the average length is lowered to 1.85 bits/symbol; it is still far from the theoretical limit because the probabilities of the symbols are different from negative powers of two.



Huffman tree generated from the exact frequencies of the text "this is an example of a huffman tree". The frequencies and codes of each character are below. Encoding the sentence with this code requires 135 bits. (This assumes that the code tree structure is known to the decoder and thus does not need to be counted as part of the transmitted information.)

30.5 File Format MP3

von Ina Desnica, Jun'05

1987 begann das Fraunhofer IIS die Arbeit über wahrnehmende Toncodierung im Rahmen des EUREKA Projektes EU147. Durch die Kooperation mit der Universität von Erlangen (Prof. Dieter Seitzer), hat das Fraunhofer IIS endgültig einen sehr leistungsfähigen Algorithmus erfunden, der heute als ISO-MPEG Audio Layer 3 standadisiert ist. Mit dem Standardformat, in dem Audiodateien in der Regel am Computer vorliegen, kann man die Online-Übertragung nicht bewerkstelligen. Dazu sollte man sich veranschaulichen, welche Datenrate für die Wiedergabe von Audiodateien erforderlich ist. Üblicherweise gelten für unkomprimierte Audiodateien auf einer Audio-CD folgende Eckwerte: eine Samplefrequenz von 44,1 KHz sowie eine Auflösung von 16 Bit sowie Stereo. Prinzipiell heißt dies, dass die Abtastung eines Signals 44.100 Mal pro Sekunde erfolgt, das Ergebnis jeder Messung einen 16-Bit-Wert ergibt und das Ganze mal zwei Kanäle (Stereo) passiert. Hieraus ergibt sich eine Bitrate von 1,4 MBit/Sekunde für ein Stereo-Audiosignal.

Zum Vergleich:
Bei MP3 genügt eine Datenrate von 128 KBit/s zur Übertragung von Musik in annähernd CD-Qualität. Auch die Speicherkapazität ist nicht zu vernachlässigen: Auf einer Audio-CD kommen rund 10 MByte auf eine Minute Musik, MP3-Dateien in guter Qualität geben sich mit zirka einem Zehntel dessen zufrieden. Sprich: Mit MP3 lassen sich rund 10 Stunden Musik auf einer Daten-CD unterbringen. Um eine so hohe Kompressionsrate ohne signifikante Qualitätsverluste zu erreichen, ist eine komplexe Bearbeitung des Audiosignals erforderlich. Damit sind die wesentlichen Merkmale und Eigenschaften für die Verwendung des MP3-Formats gegeben,

30.5.1 2. Bezeichnung MP3:

Die Bezeichnung „MP3" für diese populäre Gruppe der Dateiformate ist auf die ursprünglichen Standardbenennungen „MPEG-I Layer 3" und teilweise auch „MPEG-II Layer 3" zurückzuführen. Diese beiden Standards stammen aus der international gebräuchlichen Familie der "Moving Pictures Experts Group" — eine Möglichkeit der komprimierten Digitalisierung von Audio- und Videosignalen, die heute hauptsächlich zur Online-Multimedia-Übertragung und beim digitalen Rundfunk und Fernsehen verwendet wird.

Man muss allerdings Unterscheidungen vornehmen, um Verwechslungen vorzubeugen. Die erste nötige Unterscheidung ist eine durch die stetige Weiterentwicklung des Kompressions-Algorithmus wesentliche Endnumerierung des Kürzels MPEG. Es gibt MPEG-I, MPEG-II und auch MPEG IV. MPEG-III war zuerst ein eigenständiges Projekt, die Technologie floß dann aber doch noch in die Verbesserungen an MPEG-II ein. MPEG-I brachte die erste Kodierung von Audio und Video. Während die Videokompression weitgehend im MPEG-II Format weiterentwickelt wurde, ist MPEG-I als Forschungsebene für die Audiokompression maßgebend. MPEG-IV sollte Verbesserungen sowohl im Video- als auch im Audiobereich bringen. Es ist also die MPEG-I Technologie, die als Grundlage der MP3-Dateien anzusehen ist.

30.5.2 3. Die drei Layer

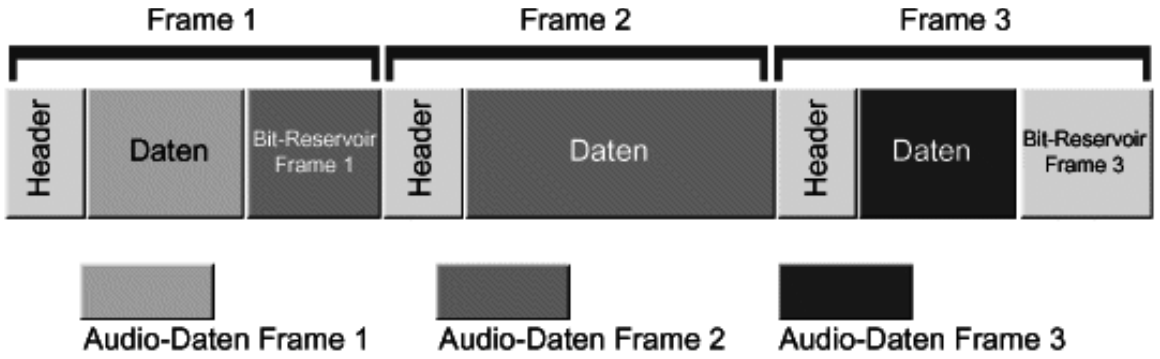
Verfahren	Kompression	Datenmenge	Anwendungsbereiche
Original-daten	keine, 1:1	1411,200 KBit/s = 172 KByte/s	Audio-CD, etc.
MPEG Layer-1	1:4	384 KBit/s = 48KByte/s	Digital Compact Cassette (DCC)
MPEG Layer-2	1:6 bis 1:8	256-192 KBit/s = 32-24 KByte/s	Digital Audio Broadcast (DAB), CD-I, DVD
MPEG Layer-3	1:10 bis 1:12	128-112 KBit/s = 16-14 KByte/s	ISDN, Satellitenradio, Internet-Audio

Eine weitere Unterscheidung ist bei der Bezeichnung des „Layers" zu treffen. Die Forschungen an der MPEG-I Audiokompression werden in drei sogenannte Layer unterteilt, wobei MPEG-I Layer 1 die ältesten und MPEG-I Layer 3 die jüngsten Forschungsergebnisse präsentiert. Die Komplexität der Kodierung und Verringerung der Dateigröße bei gleichbleibender Qualität steigen von Layer 1 zu Layer 3. Mit Layer 1 kann eine Datenkompression von 1:4 erreicht werden, ohne einen qualitativen Verlust hinzunehmen. Layer 2 erreicht dagegen schon eine Kompressionsrate von 1:6 bis 1:8 und Layer 3 sogar eine Kompression von 1:10 bis zu 1:12. MPEG-I Layer 2 und Layer

3 werden am häufigsten verwendet und sind an den Dateiendungen .mp2 und .mp3 zu erkennen. Für alle drei Layer gilt, dass die Codierung stark asymmetrisch ist. Das bedeutet: der Aufwand ist beim MP3-Encoder (= das Programm zur Komprimierung und Umwandlung der Audiosignale) deutlich höher als beim MP3-Decoder (= das Programm zum Entpacken und Ablesen der komprimierten MP3-Dateien). Da alle während des Kodierens errechneten Werte in der MP3-Datei abgespeichert werden, beschränkt sich die Arbeit des Decoders auf die Interpretation dieser Werte. Gemeinsam ist allen drei Layern die Unterstützung folgender Sampleraten: 32, 44,1 oder 48 kHz.

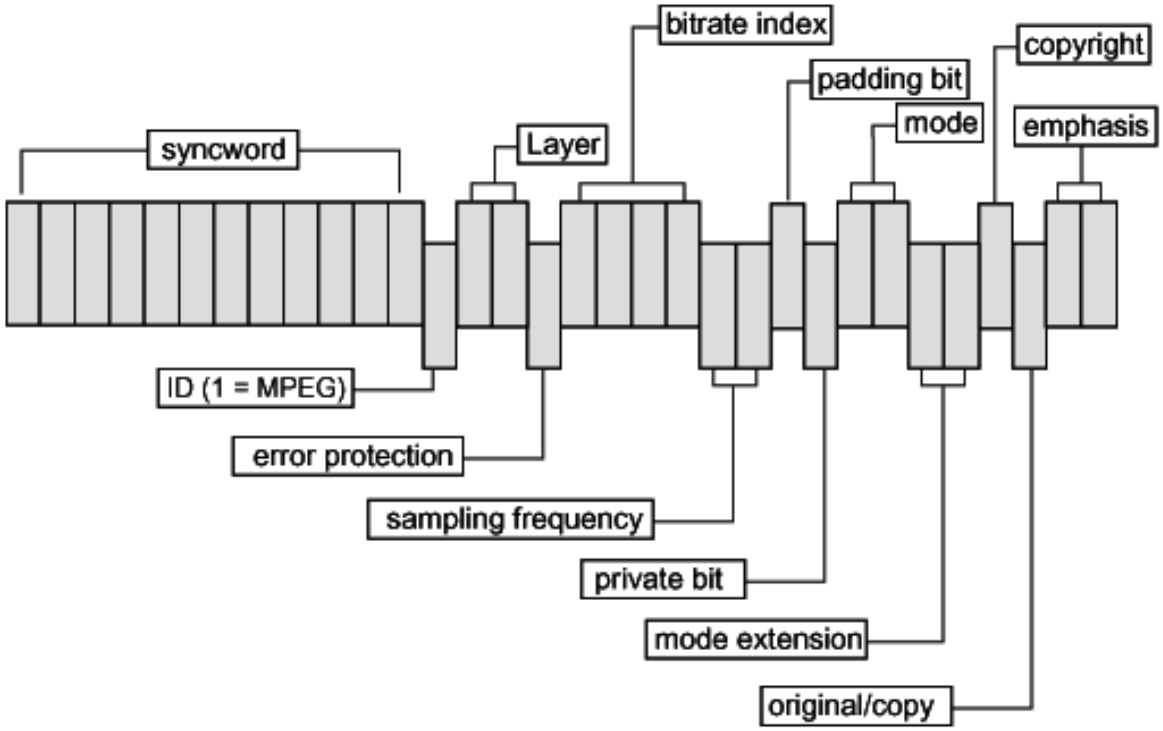
30.5.3 Aufbau der Datei

Der Aufbau der MP3-Datei erfolgt in sogenannten „Frames“. Jeder Frame enthält eine feste Anzahl von Abtastwerten (Samples). Bei Level 3 sind dies 1152 Abtastwerte pro Frame (32 Subbänder x 36 Samples – siehe auch Kompressionsverfahren!). Ein Frame besteht aus einem Header, einem Prüfsummencheck, den eigentlichen Audiodaten sowie unter Umständen einem so genannten Bit-Reservoir. Ein solches Reservoir entsteht, wenn sich die Samples innerhalb des Frames so komprimieren lassen, dass nicht die komplette theoretische Bit-Anzahl eines Frames benötigt wird.



(Bildquelle: Schwerpunkfacharbeit J. DESNICA)

Auf diese Reservoirs kann der Encoder zurückgreifen, wenn bei einem späteren Frame die vorhandenen Bits nicht ausreichen. Hierbei muss man zwei Begriffe unterscheiden: Framegröße und Framelänge. Die Framegröße wird durch die Anzahl der Samples bestimmt und ist innerhalb eines Layers konstant. Im Format Layer 1 sind dies stets 384 Samples pro Frame, bei Layer 2 und 3 kommen 1152 auf ein Frame. Die Länge des Frames kann sich jedoch bei Layer 3 durch das Wechseln der Bitrate oder das nicht aufgefüllte Bit-Reservoir unterscheiden. Vor Layer 3 existierte die Möglichkeit eines Bit-Reservoirs nicht. Theoretisch ist auch eine zerstückelte MPEG-Sounddatei abspielbar. Dies gilt jedoch nicht für nach Layer 3 kodierte Dateien, da hier die Frames auf Grund des Bit-Reservoirs verschachtelt sind. Ebenfalls im Frame enthalten sind die vorher erwähnten Informationen bezüglich des Skalierungsfaktors und der Bit Allocation, um alle Samples wieder rekonstruieren zu können. Interessant ist auch die Einführung der Frameheaders. Da ein herkömmlicher Dateiheader, wie er bei anderen Dateiformaten durchaus üblich ist, bei MP3-Dateien nicht existiert, enthält jeder einzelne Frame einen eigenen Header. Die ersten 32-Bit des Frames werden davon belegt und zwar in folgender Reihung:



(Bildquelle: Schwerpunkfacharbeit J. DESNICA)

Syncword (12 Bit) ID (1 Bit): Offiziell ist die ID-Signatur nur ein Bit groß, um die verwendete MPEG-Version zu kennzeichnen. In der Urfassung steht der Wert 1 für MPEG 1 und der Wert 0 für MPEG 2. Mit dem inoffiziellen Standard MPEG 2.5 reicht ein Bit als Kennung jedoch nicht mehr aus. Anwendungen, die diesen Standard bereits unterstützen, verwenden das zwölfte Bit des Syncwords als zusätzliche MPEG-ID. Ist dieses auf 0 gesetzt, steht dies für MPEG 2.5. Standardmäßig sind alle Bits des Syncwords gesetzt.

Layer (2 Bit): Gibt an, nach welchem Layer die Codierung erfolgt ist.

error protection (1 Bit): Wenn das Error Protection Bit gesetzt ist, folgt nach dem Frameheader eine 16-Bit-Prüfsumme zur Feststellung von Datenverlust.

bitrate_index (4 Bit): Hier steht, in welcher Bitrate die Datei codiert wurde.

sampling_frequency (2 Bit): Hier wird die Sampling Rate in Hz festgelegt. Richtet sich unter anderem nach der in der MPEG-ID festgelegten MPEG-Version.

padding_bit: Dieses Bit wird dazu benutzt, um die Bitraten exakt zu bestimmen. Ist es auf 0 gesetzt, ist das Frame nicht aufgefüllt.

private_bit (1 Bit): Steht zur freien Verfügung. Beispielsweise um in Applikationen ein bestimmtes Ereignis auszulösen.

mode (2 Bit): Hier erfolgt die Angabe über den Stereomodus. Es wird unterschieden, ob die Datei Stereo, Joint Stereo, Zweikanal oder Mono codiert ist.

mode extension (2 Bit): Ist nur von Bedeutung, wenn bei der Kodierung der Joint-Stereo-Modus gewählt wurde. Während bei Layer 1 und 2 hier die Zuordnung der Frequenzbänder an den Joint-Stereo-Modus erfolgt, gilt für Layer-3-Dateien die Festlegung welcher Joint-Stereo-Modus zum Einsatz kommt (intensity stereo oder M/S-stereo). Beim Layer 3 bestimmt der Dekompressionsalgorithmus die Frequenzbänder.

copyright (1 Bit): Ist das Copyright-Bit gesetzt, ist die MP3-Datei urheberrechtlich geschützt.

original/copy (1 Bit): Gibt an, ob es sich um ein Original oder eine Kopie handelt.

emphasis (2 Bit): Hier wird die verwendete Rauschunterdrückung angegeben. Der Einsatz ist allerdings nicht sehr gebräuchlich.

Während also die ersten 32 Bit eines jeden Frames mit Informationen zur Datei an sich belegt sind, können am Ende der gesamten MP3-Datei 128 Byte im ID3-Tag Daten zum Musikstück im Allgemeinen aufgelistet werden – eine Art Inhaltsangabe ähnlich dem CD-Text bei Audio-CDs.

Länge (Byte)	Beschreibung
3	TAG-Identifizierung, hier muss TAG stehen
30	Titel des Stückes
30	Interpret
30	CD-Titel
4	Erscheinungsjahr
30	Kommentar
1	Genre

Das „Genre“ ist ein nummerisches Feld. Es existiert eine Liste, in der bestimmte Nummern den Genres von Acid Punk bis Southern Rock zugeordnet sind. Mittlerweile wird außerhalb der MP3-Spezifikation der ID3-Tag von unabhängigen Interessensgruppen weiterentwickelt. Da der ID3-Tag immer am Ende der MP3-Datei steht, ist es relativ einfach, ihn auszulesen beziehungsweise zu editieren. So werden zum Beispiel in neueren Varianten des ID3-Tags Bytes des Kommentarfelds geopfert, um die Original-Track-Nummerierung der CD ebenfalls speichern zu können. Inzwischen gibt es bereits zahlreiche Share- und Freewareprogramme, die sich der Bearbeitung des ID3-Tags annehmen. Beliebte Funktionen sind dabei die automatische Erzeugung des ID3-Tag aus Verzeichnis- und Dateinamen sowie umgekehrt das Benennen von Dateien auf Grund vorliegender ID3-Tag-Informationen.

30.5.4 Kompressionsverfahren

Some typical performance data of MPEG Layer-3 are:

sound quality	bandwidth	mode	bitrate	reduction ratio
telephone sound	2.5 kHz	mono	8 kbps *	96:1
better than short wave	4.5 kHz	mono	16 kbps	48:1
better than AM radio	7.5 kHz	mono	32 kbps	24:1
similar to FM radio	11 kHz	stereo	56...64 kbps	26...24:1
near-CD	15 kHz	stereo	96 kbps	16:1
CD	>15 kHz	stereo	112..128kbps	14..12:1

*) Fraunhofer IIS uses a non-ISO extension of MPEG Layer-3 for enhanced performance ("MPEG 2.5")

Zunächst muß man wissen, daß es sich beim MPEG-Audiocodieren nicht um ein Kompressionsverfahren im eigentlichen Sinne handelt, sondern um eine verlust- behaftete Codierung (Reduktion). Der Unterschied zwischen Kompression und Reduktion ist, dass bei einer Reduktion wie der des Mp3-Formates, eine Herstellung der Ausgangsdatei nicht mehr möglich ist, da Toninformationen komplett gelöscht werden. Das heißt, der Informationsgehalt wird tatsächlich verringert und läßt sich bei der Decodierung nicht mehr exakt rekonstruieren. Im Gegensatz dazu bleiben bei der Codierung mittels Kompression die Informationen des Audiosignals erhalten und können beim Decodieren wieder vollständig hergestellt werden. Der Kompressionsvorgang kann in mehrere Schritte eingeteilt werden: 1. Teilung des Audiosignals Bei der MPEG-Audiokodierung wird zuerst das zu kodierende Audiosignal vom Zeit- in den Frequenzbereich überführt. Während dieses Vorgangs wird es zunächst in 32 Frequenzbänder (Subbands) unterteilt. Bis zum Layer 2 hatten alle 32 Subbänder die gleiche Größe von 625 Hz. Seit dem Layer 3 sind die Subbänder an die Eigenschaften des menschlichen Gehörs angepasst. Hierfür sorgt ein sogenannter polyphaser Filter, sprich es geschieht eine gleichzeitige Verminderung und Filterung der Abtastwerte. Jedes Subband repräsentiert dabei einen bestimmten Ausschnitt des Frequenzspektrums. Das in Bänder zerlegte Audiosignal bietet dem Algorithmus deutlich mehr Angriffsfläche, als das uniforme Audiosignal. Der MP3-Encoder knöpft sich jedes Subband einzeln vor und untersucht es gemäß dem Psychoakustischen Modell (siehe Psychoakustik weiter unten) auf verzichtbare Frequenzen. In einem ersten Schritt fallen die Subbänder weg, deren Pegel unterhalb dieser Verdeckungsfunktion liegt. Jedes Subband erfährt nun eine Abtastung, deren Resultat 16-Bit-große Samples sind. Bei der Teilung dieser Samples in kleine Teilbereiche entsteht der nächste Ansatzpunkt zur Datenreduktion. Jedes Sample besteht zwar aus 16 Bit, jedoch sind nicht unbedingt alle 16 nötig, um den Pegel darzustellen. So können beispielsweise die führenden Nullen eines 16-Bit-Samples entfallen. Ergibt sich bei einem Sample der Wert 0000011101010101, so stutzt der

Algorithmus das Ergebnis auf 11101010101. Um aus diesen reduzierten Angaben wieder die originalen 16 Bit rekonstruieren zu können, benötigt der Decoder zwei Angaben: den Skalierungsfaktor sowie die Bit Allocation. Der Skalierungsfaktor gibt an, an welcher Stelle die verbliebenen Bits des Samples sich im ursprünglichen Zustand befunden haben. Die Bit Allocation enthält die Information, wie viele Bits im Sample verblieben sind, da man ja nicht mehr mit einer festen Zahl von 16 Bit rechnen kann. Würde man nun diese Informationen für jedes Sample einzeln ablegen, wäre nicht viel gewonnen. Daher teilen sich je zwölf Samples diese Werte. 1. Kompression Layer III Bereits Layer 1 und 2 unterteilten in Subbänder, Layer 3 geht jedoch einen Schritt weiter. Da es nämlich bei der in Schritt 1 erwähnten Aufteilung in Zeit- und Frequenzbereich zu einer Überlappung der 32 Subbänder kommen kann, wodurch ein Ton einer festen Frequenz zwei Bänder beeinflussen kann (man spricht hier vom sogenannten ÄliasingEffekt), erfolgt beim Layer 3 eine modifizierte diskrete Cosinus-Transformation, die sich um die Zeit-/Frequenztransformation kümmert. Geht es um die Bearbeitung von Sound, lassen sich grundsätzlich zwei Perspektiven unterscheiden. Entweder das Audiosignal wird als eine Serie von Samples betrachtet, die einem analogen Signal entsprechen. Oder die Unterscheidung erfolgt nicht über die Zeit, sondern über die Frequenzen. Je nach Art der gewünschten Manipulation eignet sich die eine oder andere Perspektive besser. Das Mischen von Signalen oder das Erhöhen der Amplitude geschieht leichter in einer zeitorientierten Basis. Frequenzmanipulationen fallen naturgemäß bei der nach Frequenzen aufgeschlüsselten Sichtweise leichter. Um die Daten von einer in die andere Perspektive zu konvertieren, bedient man sich einer speziellen Transformation. Um ein Optimum zwischen Zeit- und Frequenzunterscheidung zu erreichen, bildet Layer 3 zwei verschiedene Blocklängen: eine lange mit 36 Samples und eine kurze mit zwölf Samples. In der Praxis heißt dies: Bei den die niedrigen Frequenzen kommen lange Blöcke zum Einsatz. Die langen Blöcke würden jedoch bei höheren Frequenzen keine ausreichende Auflösung erlauben, hier finden die kurzen Blöcke Verwendung. Im so genannten Mixed Block Mode kommen für die beiden Frequenzbänder mit den niedrigsten Frequenzen lange Blöcke zum Einsatz. Für die 30 verbliebenen Frequenzbänder sind die kurzen Blöcke an der Reihe. Damit erlaubt dieser Modus eine bessere Frequenzauflösung bei den niedrigen Frequenzen, ohne eine Tribut an die Abtastrate bei den hohen Frequenzen zu zahlen. 2. Kodierung mittels Huffmann-Verfahren Als letzter Schritt der Komprimierung erfolgt beim Layer III eine Huffman-Kodierung. Dieser Algorithmus kommt beispielsweise auch bei Packprogrammen wie ZIP zum Einsatz. Bei Huffman ist die Häufigkeit bestimmter Werte entscheidend. Häufig auftauchende Werte erhalten eine kurze Bitfolge, selten auftretende Werte hingegen eine Lange. Daher ermittelt der Algorithmus zunächst die Verteilung der Werte innerhalb der zu komprimierenden Daten. Um einen sogenannten Huffman-Baum zu ermitteln, beginnt man mit den beiden seltensten Werten. Ihnen wird eine "0"beziehungsweise eine "1ßugewiesen. Es erfolgt eine Zusammenfassung der beiden Werte, in der Reihenfolge sind sie nun durch die Summe ihrer Häufigkeit repräsentiert. Das gleiche geschieht mit den nächsten beiden seltensten Werten. Dieser Vorgang ist beendet, wenn nur noch ein Wert übrig ist. Das Ergebnis dieser Vorgehensweise ist eine Baumstruktur. Anhand dieser Struktur erfolgt die Kodierung. Jede Verzweigung nach links erhält eine 0, jede Rechtsverzweigung ist durch eine "1"gekennzeichnet. In unserem kleinen Beispiel wäre der weniger häufige Wert 4 durch die Bitfolge 010 repräsentiert. Der häufigste Wert 6 bekommt dagegen eine schlichte 1 zugeordnet. Vor der Huffman-Kodierung steht die Anordnung der Subbänder. Die Frequenz gibt die Reihenfolge der Bänder vor. Die Subbänder mit niedrigeren Frequenzen enthalten üblicherweise deutlich mehr Werte, als diejenigen der hohen Frequenzen, wo in der Regel 0 oder Werte in der Nähe von 0 erscheinen. Der Encoder unterteilt die Subbänder in drei Bereiche. Jeder Bereich erhält einen eigenen Huffman-Baum, um den optimalen Kompressionsfaktor zu erreichen. Zunächst klammert der Encoder die Frequenzbänder mit den hohen Frequenzen aus, eine Kodierung ist hier nicht notwendig, da sich ihre Größe aus denen der anderen zwei Regionen ableiten lässt. Die zweite Region enthält Frequenzbänder, in denen die Werte von -1 bis 1 häufig auftauchen. Der dritte Bereich enthält alle verbleibenden Werte und wird ein weiteres Mal in drei Regionen unterteilt, von denen jede einen eigenen Huffman-Baum zugewiesen bekommt. Welcher Huffman-Baum zum Einsatz kommt, wird innerhalb der MP3-Datei abgelegt.

30.6 WAV File Format

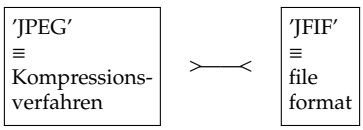
```
/* general extended waveform format structure
 * Use this for all NON PCM formats
 * (information common to all formats):
 */

typedef struct waveformat\_extended\_tag {
  WORD wFormatTag;       /* format type */
  WORD nChannels;        /* number of channels (i.e. mono, stereo...) */
  DWORD nSamplesPerSec;  /* sample rate */
  DWORD nAvgBytesPerSec; /* for buffer estimation */
  WORD nBlockAlign;      /* block size of data */
  WORD wBitsPerSample;   /* Number of bits per sample of mono data */
  WORD cbSize;           /* The count in bytes of the extra size */} WAVEFORMATEX;
```

wFormatTag	Defines the type of WAVE file.
nChannels	Number of channels in the wave, 1 for mono, 2 for stereo
nSamplesPerSec	Frequency of the sample rate of the wave file. This should be 11025, 22050,or 44100. Other sample rates are allowed, but not encouraged. This rate is also used by the sample size entry in the fact chunk to determine the length in time of the data.
nAvgBytesPerSec	Average data rate. Playback software can estimate the buffer size using the <nAvgBytesPerSec> value.
nBlockAlign	The block alignment (in bytes) of the data in <data-ck>. Playback software needs to process a multiple of <nBlockAlign> bytes of data at a time, so that the value of <nBlockAlign> can be used for buffer alignment.
wBitsPerSample	This is the number of bits per sample per channel data. Each channel is assumed to have the same sample resolution. If this field is not needed, then it should be set to zero.
cbSize	The size in bytes of the extra information in the WAVE format header not including the size of the WAVEFORMATEX structure. As an example, in the IMA ADPCM format cbSize is calculated as sizeof(IMAADPCMWAVEFORMAT) - sizeof(WAVEFORMATEX) which yields two.

Defined wFormatTags

WAVE_FORMAT_G723_ADPCM	0x0014	/* Antex Electronics Corporation */
WAVE_FORMAT_ANTEX_ADPCME	0x0033	/* Antex Electronics Corporation */
WAVE_FORMAT_G721_ADPCM	0x0040	/* Antex Electronics Corporation */
WAVE_FORMAT_APTX	0x0025	/* Audio Processing Technology */
WAVE_FORMAT_AUDIOFILE_AF36	0x0024	/* Audiofile, Inc. */
WAVE_FORMAT_AUDIOFILE_AF10	0x0026	/* Audiofile, Inc. */
WAVE_FORMAT_CONTROL_RES_VQLPC	0x0034	/* Control Resources Limited */
WAVE_FORMAT_CONTROL_RES_CR10	0x0037	/* Control Resources Limited */
WAVE_FORMAT_CREATIVE_ADPCM	0x0200	/* Creative Labs, Inc */
WAVE_FORMAT_DOLBY_AC2	0x0030	/* Dolby Laboratories */
WAVE_FORMAT_DSPGROUP_TRUESPEECH	0x0022	/* DSP Group, Inc */
WAVE_FORMAT_DIGISTD	0x0015	/* DSP Solutions, Inc. */
WAVE_FORMAT_DIGIFIX	0x0016	/* DSP Solutions, Inc. */
WAVE_FORMAT_DIGIREAL	0x0035	/* DSP Solutions, Inc. */
WAVE_FORMAT_DIGIADPCM	0x0036	/* DSP Solutions, Inc. */
WAVE_FORMAT_ECHOSC1	0x0023	/* Echo Speech Corporation */
WAVE_FORMAT_FM_TOWNS_SND	0x0300	/* Fujitsu Corp. */
WAVE_FORMAT_IBM_CVSD	0x0005	/* IBM Corporation */
WAVE_FORMAT_OLIGSM	0x1000	/* Ing C. Olivetti & C., S.p.A. */
WAVE_FORMAT_OLIADPCM	0x1001	/* Ing C. Olivetti & C., S.p.A. */
WAVE_FORMAT_OLICELP	0x1002	/* Ing C. Olivetti & C., S.p.A. */
WAVE_FORMAT_OLISBC	0x1003	/* Ing C. Olivetti & C., S.p.A. */
WAVE_FORMAT_OLIOPR	0x1004	/* Ing C. Olivetti & C., S.p.A. */
WAVE_FORMAT_IMA_ADPCM (WAVE_FORM_DVI_ADPCM)		/* Intel Corporation */
WAVE_FORMAT_DVI_ADPCM	0x0011	/* Intel Corporation */
WAVE_FORMAT_UNKNOWN	0x0000	/* Microsoft Corporation */
WAVE_FORMAT_PCM	0x0001	/* Microsoft Corporation */
WAVE_FORMAT_ADPCM	0x0002	/* Microsoft Corporation */
WAVE_FORMAT_ALAW	0x0006	/* Microsoft Corporation */
WAVE_FORMAT_MULAW	0x0007	/* Microsoft Corporation */
WAVE_FORMAT_GSM610	0x0031	/* Microsoft Corporation */
WAVE_FORMAT_MPEG	0x0050	/* Microsoft Corporation */
WAVE_FORMAT_NMS_VBXADPCM	0x0038	/* Natural MicroSystems */
WAVE_FORMAT_OKI_ADPCM	0x0010	/* OKI */
WAVE_FORMAT_SIERRA_ADPCM	0x0013	/* Sierra Semiconductor Corp */
WAVE_FORMAT_SONARC	0x0021	/* Speech Compression */
WAVE_FORMAT_MEDIASPACE_ADPCM	0x0012	/* Videologic */
WAVE_FORMAT_YAMAHA_ADPCM	0x0020	/* Yamaha Corporation of America */

30.7 .jpeg File Format



"JPEG ist die gebräuchliche Bezeichnung für die 1992 vorgestellte Norm ISO/IEC 10918-1 bzw. CCITT Recommendation T.81, die verschiedene Methoden der Bildkompression beschreibt. Die Bezeichnung "JPEG" geht auf das Gremium Joint Photographic Experts Group zurück, das die JPEG-Norm entwickelt hat. JPEG schlägt verschiedene Komprimierungs- und Kodierungsmethoden vor, darunter verlustbehaftete und verlustfreie Kompression, verschiedene Farbtiefen sowie sequenzielle oder progressive Modi (normaler Bildaufbau bzw. allmähliche Verfeinerung). Weithin verbreitet ist nur die verlustbehaftete Komprimierung bei sequenziellem oder progressivem Modus und 8-Bit-Farbkanälen. Die JPEG-Norm beschreibt lediglich Bildkompressionsverfahren, legt aber nicht fest, wie die so entstandenen Daten gespeichert werden sollen. Gemeinhin werden mit "JPEG-Dateien" oder "JPG-Dateien" Dateien im Grafikformat JPEG File Interchange Format (JFIF) bezeichnet. JFIF ist jedoch nur eine Art, JPEG-Daten abzulegen; SPIFF und JNG sind weitere, wenn auch wenig gebräuchliche, Möglichkeiten. JPEG/JFIF unterstützt eine maximale Bildgröße von 65.535 × 65.535 Pixel.

Zusätzlich zum in ISO/IEC 10918-1 definierten verlustbehafteten Modus gibt es noch die verbesserte, verlustfreie Komprimierungsmethode JPEG-LS, die in einer anderen Norm festgelegt wurde. Außerdem existiert noch die JBIG-Norm zur Komprimierung von Schwarzweißbildern.

Die JPEG-Norm definiert 41 verschiedene Unterdateiformate, von denen aber meist nur eines unterstützt wird (und welches auch fast alle Anwendungsfälle abdeckt). Die Kompression erfolgt durch das Anwenden mehrerer Verarbeitungsschritte, von denen vier verlustbehaftet sind.

- Farbmodellumrechnung vom (meist) RGB-Farbraum ins YCbCr-Farbmodell (analog zu CCIR 601). (theoretisch verlustfrei, nach CCIR 601 verlustbehaftet).
- Tiefpassfilterung und Unterabtastung der Farbabweichungssignale Cb und Cr (verlustbehaftet).
- Einteilung in 8×8-Blöcke und diskrete Kosinustransformation dieser Blöcke (theoretisch verlustfrei, durch Rundungsfehler aber verlustbehaftet).
- Quantisierung (verlustbehaftet).

- Umsortierung.
- Entropiekodierung.

Die Datenreduktion erfolgt durch die verlustbehafteten Verarbeitungsschritte in Zusammenwirken mit der Entropiekodierung. Kompressionen bis etwa 1,5 – 2 Bit/Pixel sind visuell verlustfrei, bei 0,7 – 1 Bit/Pixel sind noch gute Ergebnisse erzielbar, unter 0,3 Bit/Pixel wird JPEG praktisch unbrauchbar, das Bild wird zunehmend von unübersehbaren Kompressionsartefakten (Blockbildung, stufige Übergänge, Farbeffekte an Graukeilen) überdeckt. Der Nachfolger JPEG 2000 ist wesentlich weniger für diese Art von Artefakten anfällig. Sieht man als Quellformat 24-Bit-RGB-Dateien an, erhält man Kompressionsraten von 12 bis 15 für visuell verlustfreie Bilder und bis zu 35 für noch gute Bilder. Die Qualität hängt aber neben der Kompressionsrate noch von der Art der Bilder ab. Rauschen und regelmäßige feine Strukturen im Bild verringern die maximal mögliche Kompressionsrate. Der JPEG Lossless Mode zur verlustfreien Kompression verwendet ein anderes Verfahren (prädiktiver Koder und Entropiekodierung).

Farbmodellumrechnung
Das Ausgangsbild, welches meist als RGB-Bild vorliegt, wird in das YCbCr-Farbmodell umgerechnet. Grundsätzlich wird dabei das YPbPr-Schema nach CCIR 601 verwendet:

$$\begin{bmatrix} Y' \\ Pb \\ Pr \end{bmatrix} \approx \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}168736 & -0{,}331264 & 0{,}5 \\ 0{,}5 & -0{,}418688 & -0{,}081312 \end{bmatrix} \cdot \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

Da die R'G'B'-Werte bereits digital als 8-Bit-Zahlen im Bereich 0..255 vorliegen, müssen die YPbPr-Komponenten lediglich verschoben (renormiert) werden, wodurch die Komponenten Y' (Luminanz), Cb (color blueness) und Cr (color redness) entstehen:

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} \approx \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}168736 & -0{,}331264 & 0{,}5 \\ 0{,}5 & -0{,}418688 & -0{,}081312 \end{bmatrix} \cdot \begin{bmatrix} R'_d \\ G'_d \\ B'_d \end{bmatrix}$$

Die Komponenten liegen nun wiederum im Wertebereich 0..255. Bei der Umrechnung des Farbmodells entstehen die üblichen Rundungsfehler durch begrenzte Rechengenauigkeit.

Tiefpassfilterung der Farbdifferenzsignale
Die Farbabweichungssignale Cb und Cr werden meist in reduzierter Auflösung gespeichert. Dazu werden sie tiefpassgefiltert und unterabgetastet (im einfachsten Fall durch eine Mittelwertbildung). Meist wird eine vertikale und horizontale Unterabtastung jeweils um den Faktor 2 verwendet (YCbCr 4:2:0), die die Datenmenge um den Faktor 4 reduziert. Bei dieser Umwandlung wird die Tatsache ausgenutzt, dass die Ortsauflösung des menschlichen Auges für Farben deutlich geringer ist als für Helligkeitsübergänge.

Blockbildung und diskrete Kosinustransformation
Jede Komponente (Y, Cb und Cr) des Bildes wird in 8×8-Blöcke eingeteilt. Diese werden einer zweidimensionalen diskreten Kosinustransformation (DCT) unterzogen:

$$F_{xy} = \tfrac{1}{4} C_x C_y \sum_{m=0}^{7} \sum_{n=0}^{7} f_{mn} \cos \tfrac{(2m+1)x\pi}{16} \cos \tfrac{(2n+1)y\pi}{16}$$

mit

$$C_x, C_y = \begin{cases} \frac{1}{\sqrt{2}} & \text{wenn } x, y = 0 \\ 1 & \text{sonst} \end{cases}$$

Statt 64 Einzelpunkte wird jeder 8×8-Block als Linearkombination dieser 64 Blöcke dargestellt. Diese Transformation lässt sich unter Nutzung der schnellen Fourier-Transformation (FFT) mit sehr wenig Aufwand implementieren. Die DCT ist eine orthogonale Transformation, weist gute Energiekompressionseigenschaften auf und es gibt eine inverse Transformation, die IDCT (was auch bedeutet, dass die DCT verlustfrei ist, es gingen keine Informationen verloren, da die Daten lediglich in eine für die weitere Verarbeitung günstigere Form gebracht wurden).

Quantisierung
Wie bei allen verlustbehafteten Kodierungsverfahren wird die eigentliche Datenreduktion (und Qualitätsverschlechterung) durch eine Quantisierung erreicht. Dazu werden die DCT-Koeffizienten durch die Quantisierungsmatrix geteilt (elementweise dividiert) und danach auf die nächstliegende Ganzzahl gerundet:

$$F^Q(x,y) = \text{round}\left(\frac{F(x,y)}{Q(x,y)}\right) F^Q(x,y) = \text{round}\left(\frac{F(x,y)}{Q(x,y)}\right)$$ Bei diesem Rundungsschritt findet eine Irrelevanzreduktion statt. Die Quantisierungsmatrix ist sowohl für die Qualität als auch für die Kompressionsrate verantwortlich. Sie ist in JPEG-Dateien im Header abgespeichert (DQT-Marker). Optimal ist die Quantisierungsmatrix, wenn sie in etwa die Empfindlichkeit des Auges für die entsprechenden Ortsfrequenzen repräsentiert. Für grobe Strukturen ist das Auge empfindlicher, daher sind die Quantisierungswerte für diese Frequenzen kleiner als die für hohe Frequenzen.

Umsortierung und Differenzkodierung des Gleichanteils
Die 64 Koeffizienten der diskreten Cosinus-Transformation werden anhand der Frequenz sortiert. Dadurch ergibt sich eine zickzackförmige Reihenfolge, beginnend mit dem Gleichanteil mit der Frequenz 0. Nach dem englischen Direct Current (für Gleichstrom) wird er mit DC abgekürzt, hier bezeichnet er die mittlere Helligkeit. Die Koeffizienten mit hohem Wert stehen nun meist zuerst und kleine Koeffizienten weiter hinten. Dies optimiert die Eingabe der nachfolgenden Lauflängenkodierung. Weiterhin wird der Gleichanteil noch einmal differentiell zum Block links daneben kodiert und auf diese Weise die Abhängigkeiten zwischen benachbarten Blöcken berücksichtigt.

Entropiekodierung
Als Entropiekodierung wird meist eine Huffman-Kodierung verwendet. Der JPEG-Standard erlaubt auch eine arithmetische Kodierung. Obwohl diese zwischen 5 und 15 Prozent kleinere Dateien generiert, wird sie aus patentrechtlichen Gründen kaum verwendet, zudem ist diese Kodierung deutlich langsamer.

Die JPEG-Dekodierung
Die Dekompression (meist Dekodierung genannt) erfolgt invers zur Kompression:
- Entropie-Dekodierung
- Umsortierung
- Requantisierung
- Inverse Diskrete Kosinustransformation.
- Überabtastung und Tiefpassfilterung der Farbdifferenzsignal U und V (verlustbehaftet)
- Farbmodellumrechnung vom YCbCr-Farbmodell in den Zielfarbraum (meist RGB)

Die Dekompression ist zwar (weitgehend) verlustfrei, allerdings tritt das Inverse-Dekoder-Problem auf. Aus dekodierten Daten ist es nur schwierig möglich, die ursprüngliche Datei zu rekonstruieren. Ein Dekodier-Kodier-Vorgang verändert die Datei und ist damit nicht verlustfrei, es treten wie beim analogen Überspielen Generationsverluste auf. Die Generationsverluste von JPEG sind allerdings vergleichsweise klein, wenn wieder die gleiche Quantisierungstabelle zum Einsatz kommt und die Blockgrenzen die gleichen sind. Bei geeigneten Randbedingungen kann man sie bei JPEG sogar vermeiden. Bei JPEG-2000 ist das nicht mehr der Fall (überlappende Transformationen, wie sie bei JPEG-2000 wie auch in der Audiodatenkompression zum Einsatz kommen, erfordern dafür utopische Rechenleistungen).

Inverse diskrete Kosinustransformation
Zur DCT existiert die inverse Transformation, die IDCT:

$$f_{xy} = \tfrac{1}{4} \sum_{m=0}^{7} \sum_{n=0}^{7} C_m C_n F_{mn} \cos\left[\tfrac{(2x+1)m\pi}{16}\right] \cos\left[\tfrac{(2y+1)n\pi}{16}\right]$$

mit den gleichen Korrekturfaktoren C_k wie bei der DCT.

Farbmodellumrechnung
Die Rückrechnung vom YCbCr-Farbmodell in den RGB-Farbraum erfolgt über die inverse Matrix der Hinrechnung.

Progressives JPEG Ein JPEG-Bild besteht aus Koeffizienten. Diese speichern keine Pixel, sondern Annäherungen des gesamten Bildinhalts eines 8×8-Bildblocks. Beim Progressive JPEG werden erst die ersten Koeffizienten jedes Bildblocks, dann die zweiten usw. der Reihe nach abgespeichert, so dass die Annäherung an das Originalbild immer besser wird. Wie beim Interlacing, das bei GIF angewendet wird, liegt der Zweck darin, dem Benutzer, noch bevor die gesamte Datei geladen ist, schnell ein grobes Vorschaubild zu geben. Dies ist besonders dann sinnvoll, wenn das Laden eines Bildes länger als eine halbe bis ganze Sekunde dauert bzw. man nur ein Vorschaubild benötigt. Jedoch werden große Bilder trotzdem meistens im normalen JPEG-Modus zum Download angeboten.

Verlustfreie visuelle Nachbearbeitung
Verluste beim wiederholten Rotieren und Speichern eines JPEG-Bildes mit "krummer" Auflösung 1021 x 767 (nicht

durch 16 Teilber). Das wiederholte Rotieren von JPEG-Bildern mit durch 16 teilbarer Auflösung (zB. 1024x768) bei Verwendung immer der gleichen Quantisierungsmatrix ist dagegen (bei ordnungsgemäßer Implementierung) verlustfrei. Obwohl eine Dekodierung und Rekodierung meist verlustbehaftet ist, lassen sich einige Bildmanipulationen (prinzipiell) ohne unerwünschte Datenverluste durchführen:

- Bilddrehungen um 90°, 180° und 270°
- horizontale und vertikale Bildspiegelungen
- Beschneiden von Rändern um Vielfache von 16 Pixeln (bzw. 8 Pixel bei Schwarzweißbildern oder Farbbildern ohne Unterabtastung)

Dazu ist die Entropiekodierung und die Zickzack-Umsortierung rückgängig zu machen. Die Operationen erfolgen dann auf Grundlage der DCT-Koeffizienten (Umsortieren, Weglassen nicht benötigter Blöcke). Danach erfolgen wieder die Zickzack-Umsortierung und die Entropiekodierung. Es erfolgen keine verlustbehafteten Arbeitsschritte mehr. Nicht jedes Programm führt diese Operationen verlustfrei durch, es benötigt dazu spezielle dateiformatspezifische Verarbeitungsmodule. Bei den verbreiteten Bildbearbeitungsprogrammen ist das meist nicht der Fall, da diese in der Regel die Datei zunächst in ein Bitmuster dekodieren und dann mit diesen Daten arbeiten. Beispielsweise das für Windows und Linux verfügbare Konsolenprogramm jpegtran vermag es, alle diese Operationen verlustfrei auszuführen, ebenso das GUI-gestützte IrfanView für Windows. Bilder, deren Auflösung nicht ein Vielfaches von 16 Pixeln (bzw. 8 Pixel bei Schwarzweißbildern oder Farbbildern ohne Unterabtastung) beträgt, sind problematisch. Sie weisen unvollständige Blöcke auf, das heißt, Blöcke, die nicht alle synthetisierten Pixel verwenden. JPEG erlaubt solche Blöcke aber nur am rechten und am unteren Bildrand. Einige dieser Operationen verlangen daher einmalig, dass diese Randstreifen verworfen werden.

Verlustfreie erweiterte Kompression der Daten
Die Firma Dropbox hat ein Programm entwickelt, welches mittels arithmetischen Kodierens den Platzbedarf von bereits vorhandenen JPEG-Dateien nochmals um durchschnittlich ungefähr 20 Prozent verkleinert. Das Programm heißt Lepton und steht unter der Apache-2.0-Lizenz, einer Open-Source-Lizenz.

Visuelle Qualität und verwandte Formate
Die JPEG-Kompression ist für natürliche (Raster-)Bilder entwickelt worden, wie man sie in der Fotografie oder bei computergenerierten Bildern vorfindet. Ungeeignet ist JPEG für

- digitale Strichzeichnungen zB. Screenshots oder Vektrografiken, die viele hochfrequente Bildteile wie harte Kanten enthalten
- Schwarzweißbilder mit 1 Bit pro Bildpunkt
- gerasterte Bilder (Zeitungsdruck)

Für diese Bilder sind Formate wie GIF, PNG oder JBIG weitaus besser geeignet. Ein nachträgliches Heraufsetzen des Qualitätsfaktors vergrößert zwar den Speicherbedarf der Bilddatei, bringt aber verlorene Bildinformation nicht mehr zurück. Die Quantisierungstabellen können beliebig gewählt werden und sind nicht genormt. Viele Bildbearbeitungsprogramme lassen aber den Benutzer einen pauschalen Qualitätsfaktor zwischen 0 und 100 auswählen, der gemäß einer Formel in der vom JPEG-Komitee herausgegebenen JPEG-Bibliothek in eine Quantisierungstabelle umgewandelt wird. Auch bei Qualitätsfaktoren wie ”100” oder ”100 %” findet immer noch eine Quantisierung und damit ein – bei für JPEG ungeeigneten Bildern erheblicher – Qualitätsverlust statt. Eine JPEG-Transformation ist im Allgemeinen nicht idempotent. Das Öffnen und anschließende Speichern einer JPEG-Datei führt zu einer neuen verlustbehafteten Kompression.

JPEG XL
Daneben ist mit JPEG XL ein weiteres Nachfolgeformat in Entwicklung. Es integriert eine Reihe von Funktionen, für die bisher andere Bildformate verwendet wurden und bietet eine verbesserte Komprimierung, die auf den experimentellen Formaten PIK und FUIF basiert. Im Januar 2021 wurde ein Format Freeze bekanntgegeben. Im Mai 2021 wurde gemeldet, dass das Format in Vorabversionen von Chrome, Edge und Firefox (noch mit zusätzlichen Schaltern gesichert) ausprobiert werden kann.

Patentfragen
Mehrere Firmen (s. Patent-Trolle) haben bereits versucht, aus ihren (zumeist zu Unrecht erteilten) Patenten Forderungen gegen Softwarehersteller zu erheben, deren Produkte JPEG-Dateien lesen oder erstellen können. Bisher wurden alle entsprechenden Patente nachträglich entzogen. Dennoch konnten die Forderer in außergerichtlichen Einigungen hohe Millionenbeträge einnehmen.

Implementierungen
Eine sehr wichtige Implementierung eines JPEG-Codecs ist die freie Programmbibliothek libjpeg. Sie wurde 1991 erstmals veröffentlicht und war ein Schlüssel für den Erfolg des Standards. Aus ihr entstanden auch die optimierten Abspaltungen libjpg-turbo und MozJPEG. Die Bibliothek und ihre Abkömmlinge werden in einer unüberschaubaren Anzahl von Anwendungen verwendet. Im Mai 2019 publizierte ITU und ISO/IEC Referenzimplementierungen des JPEG Standards unter ISO/IEC 10918-7 und ITU-T T.873. Diese Referenzsoftware besteht aus zwei Implementierungen, nämlich der libjpeg-turbo und einer von der ISO-Arbeitsgruppe SC29WG1 selbst gepflegten Referenzimplementierung und ITU-T T.873 die nicht nur den DCT-Prozess von JPEG, sondern auch die selten benutzten verlustlosen und hierarchischen Prozesse implementiert. Im März 2017 hat Google mit ”Guetzli” einen neuen Open-Source-JPEG-Encoder vorgestellt. Dieser komprimiert Bilddateien besser als bisherige Methoden, so soll Guetzli laut Google um bis zu 35 Prozent kleinere JPEG-Dateien erzeugen als herkömmliche Encoder und dabei weniger störende Kompressionsartefakte bilden. Die Kodiergeschwindigkeit (0,01 MPixel/s) liegt allerdings um etwa vier Größenordnungen unter der von Standard-JPEG-Encodern (100 MPixel/s) und ist damit in der Praxis nur für sehr häufig benutzte statische Bilder geeignet.

Das JPEG File Interchange Format (JFIF) ist ein 1991 von Eric Hamilton entwickeltes Grafikformat zur Speicherung von Bildern, die nach der JPEG-Norm komprimiert wurden. Als Dateinamenserweiterung wird meistens jpg, seltener jpeg oder jpe verwendet. JFIF ist das im World Wide Web am weitesten verbreitete Grafikformat für Fotos. Die JPEG-Norm legt nur den Komprimierungsvorgang fest, nicht aber, wie die entstandenen Bilddaten abgelegt werden müssen, damit sie zwischen verschiedenen Computerprogrammen ausgetauscht werden können. Zum Beispiel legt JPEG nicht fest, welcher Farbraum benutzt werden soll. JFIF ist eine Methode, JPEG-Daten abzulegen; weitere Möglichkeiten sind das von der Joint Photographic Experts Group veröffentlichte Still Picture Interchange File Format (SPIFF) und JPEG Network Graphics (JNG). Gemeinhin bezeichnet ”JPEG-Datei” eine JFIF-Datei. JFIF macht nur von einem kleinen Teil der von JPEG angebotenen Möglichkeiten Gebrauch: so ist als Farbmodell nur noch YCbCr und nicht mehr RGB zugelassen; zur Entropiekodierung ist nur Huffman-Kodierung erlaubt. Hinzugekommen sind einige Mechanismen zur Synchronisation und Resynchronisation bei Übertragungsfehlern.

Die Größe von JPEG-Bildern ist bei JFIF auf 65.535×65.535 Pixel beschränkt. JFIF-Bilddateien werden durch einen SOI-Marker FF D8 eingeleitet. Auf diesen Marker folgt das JFIF-Tag: FF E0 00 10 4A 46 49 46 00. Die Sequenz 4A 46 49 46 ist die ASCII-Repräsentation von ”JFIF”. Gängige Bildprogramme können auch Dateien ohne JFIF-Tag verarbeiten. Die JFIF-Dateien sind in Segmente (bei TIFF als Tag bezeichnet) unterteilt. Die Segmente sind generell in folgender Form aufgebaut: Ein FF xx s1 s2 leitet das Tag ein. FF ist ein hexadezimaler Wert und entspricht dezimal 255. Das xx bestimmt die Art des Tags. 256 * s1 + s2 gibt die Länge des Segmentes an. In den Segmenten können sich weitere Bilder verbergen, oft ein kleines Vorschaubild für die schnelle Vorschau. Einzelne Segmente können mehrfach vorkommen.

[...]
Da einen Dekoder nur Daten zwischen einer Anfangsmarke und einer Endmarke interessieren, ist es auch möglich, Exif, IPTC-IIM-Standard-Informationen und sogar ganze XML- oder andere Dateien nach dem Datacontainer-Verfahren in der JFIF-Datei unterzubringen. Die Erweiterungen sind aber nicht JFIF-konform. Es wird empfohlen solche Dateien vor einer Veröffentlichung eines Bildes zu entfernen, wenn sie private Daten enthalten.

Exif-Metadaten
In Bilddateien im JPEG-Format können auch Metadaten im Exif-Format abgespeichert werden; viele Digitalkameras speichern hier Informationen über Hersteller und Softwareversion ab. Lizenzdaten können ebenfalls gespeichert werden. Aufmerksamkeit erregte die Angewohnheit von ei-

nigen Bildaufnahme- (z. B. Software in Digitalkamera) und Bildbearbeitungsprogrammen, Vorschaubilder im Exif-Feld zu speichern, die bei einer Veränderung der Bilddatei nicht zwangsläufig angepasst werden; so können nicht zur Veröffentlichung bestimmte Teile der Bilddatei in niedriger Auflösung trotzdem an die Öffentlichkeit geraten.

Verfahren

- Konvertierung des Bildes in ein geeignetes Farbmodell (nicht im JPEG-Standard vorgeschrieben, wird aber im Regelfall durchgeführt)
- →DCT – Diskrete Cosinus Transformation (zur DFT verwandt)
- Quantisierung
- Kodierung der Koeffizienten

Konvertierung des Bildes in ein geeignetes Farbmodell
Der JPEG-Standard schreibt kein Farbmodell vor. Einige eignen sich jedoch mehr als andere. In der Regel YUV-Farbraum (technisch entstanden, Farbfernsehen/Übertragung). Helligkeit und Kontrast werden in einem einzigen Kanal (Luminanz) gespeichert. Die Farbwerte in zwei anderen Kanälen (Chrominanz).

Y:= 0.299*R + 0.587*G + 0.114*B
U:= B – Y
V:= R – Y

Das Auge kann Helligkeitsunterschiede weit besser wahrnehmen als Farbunterschiede. Die Farbkanäle können daher in ihrem Informationsgehalt verringert werden. 2 oder benachbarte Pixel in den Farbkanälen können gemittelt, und zu einem einzigen zusammengefasst werden (4:2:2 oder 4:1:1 Subsampling). So wird die Datenmenge verringert, ohne für das Auge erkennbare Unterschiede zu erzeugen (das war auch beim Analogfernsehen: Bandbreite des PAL-Restseitenbandes: 5,5MHz, Bandbreite des Farbsignals: 1,3MHz. Anmkg:XH).

DCT – Diskrete Cosinus Transformation

- Spezielle Form der Fourier-Transformation
- DCT ist kein Kompressionsverfahren
- Sie wird in vielen Vorstufen verwendet (jpg, mp3)
- Bild wird in 8x8 Matrizen aufgeteilt (alle folgenden Schritte, arbeiten mit diesen Matrizen – es wird keine Gesamtbildbetrachtung vorgenommen)
- Frequenzanalyse der Helligkeitsschwankungen (trennt grobe und feine Strukturen eines Bildes)
- Hohe Frequenz – schneller Helligkeitswechsel; niedrige Frequenz – langsamer Helligkeitswechsel (jeweils von links nach rechts und oben nach unten)

Hierdurch wird eine 8x8 Ortsmatrix in eine 8x8 Frequenzmatrix umgewandelt. Der Informationsgehalt bleibt der gleiche (die Informationen werden nur in anderer Form dargestellt).

Quantisierung
in den meisten Bildern sind viele Koeffizienten nahezu 0. Setzt man diese Koeffizienten zu 0, dann ist dies nach der Dekompression kaum bemerkbar! Weiterhin sind einige Koeffizienten wichtiger als andere.

- So ist z.B. die Grundhelligkeit und der Helligkeitsverlauf am wichtigsten (niedrige Frequenzen)
- Weniger wichtig ist die Textur des Bildes (mittlere Frequenzen)
- Sehr feine hochfrequente Details sind nahezu nicht beobachtbar

Die Koeffizienten werden also durch unterschiedlich starke Quantisierung "gleich wichtig" gemacht. Die Quantisierung erfolgt mit einer Quantisierungsmatrix. Jeder Koeffizient der DCT-Matrix wird durch einen entsprechenden Wert (aus der Quantisierungsmatrix) geteilt. Es gibt keine vorgeschriebenen Standardtabellen für Quantisierungsmatrizen. Somit lässt sich die Bildqualität über die Quantisierung steuern. Dies hat zur Folge, dass die genutzte Quantisierungsmatrix im Bild mit transportiert werden muss, um es an anderer Stelle wieder dekodieren zu können. Dies erhöht zwar den Speicherbedarf, aber erhöht auch die Flexibilität des Standards und wird somit ohne weiteres geduldet.

Die DC-Koeffizienten (Matrixinhalt an Position 1,1) benachbarter Blöcke unterscheiden sich nur wenig und werden daher als Differenz zum Vorgängerblock übertragen. Jetzt macht man sich die längeren Nullketten, die dann von einem von Null verschiedenen AC-Koeffizienten gefolgt werden, zu Nutzen. Anstatt nämlich jeden Koeffizienten einzeln zu speichern, wird jeder AC-Koeffizient, der nicht Null ist, in Kombination mit der Anzahl der Nullen, die ihm vorausgegangen sind, angegeben (Run- Length-Encoding RLE). Alle Nullen, denen kein AC-Koeffizient mit Wert ungleich Null folgt, werden weggelassen. Das "End of ..." Symbol wird dann bereits vor dem 64. Wert gesetzt. Die Entropie-Kodierung ist der letzte Komprimierungsschritt. Die durch RLE zusammengefassten Ketten werden meistens mittels Huffmann kodiert.

(aus https://de.wikipedia.org/wiki/JPEG_File_Interchange_Format und https://de.wikipedia.org/wiki/JPEG, 10.Dez'22/17.Apr'17)

30.8 PNG File Format (verlustfrei)

"Portable Network Graphics (PNG, englisch für *Portable Network Graphics*) ist das meistverwendete verlustfreie Grafikformat im Internet. Es ist ein universelles, vom World Wide Web Consortium (W3C) anerkanntes Grafikformat für Rastergrafiken mit verlustfreier Verringerung des Platzbedarfs von Daten. PNG wurde als freier Ersatz für das ältere, bis zum Jahr 2006 mit Patentforderungen belastete Graphics Interchange Format (GIF) entworfen und ist weniger komplex als das Tagged Image File Format (TIFF). PNG unterstützt neben unterschiedlichen Farbtiefen auch Transparenz per Alphakanal.

Geschichte
Die Entwicklung des PNG-Formats begann Ende 1994, verglichen mit anderen Grafikformaten wie TIFF, GIF und JFIF ("JPEG") also recht spät. Auslöser waren Lizenzforderungen der Softwarefirma Unisys für den von GIF verwendeten Lempel-Ziv-Welch-Algorithmus (LZW). Am 4. Januar 1995 legte Thomas Boutell einen frühen Entwurf (PBF Draft 1) vor. Die erste richtige PNG-Spezifikation (Version 1.0) von Thomas Boutell und Tom Lane wurde bereits am 1. Oktober 1996 offizielle Empfehlung des World Wide Web Consortium (W3C). Am 14. Oktober 1996 erhielt PNG von der Internet Assigned Numbers Authority (IANA) den MIME-Typ image/png zugewiesen. Am 15. Januar 1997 wurde PNG von der Internet Engineering Task Force (IETF) als RFC 2083 verabschiedet. Am 31. Dezember 1998 erschien die von Adam Costello und Glenn Randers-Pehrson überarbeitete PNG-Spezifikation Version 1.1. Am 11. August 1999 veröffentlichte Glenn Randers-Pehrson schließlich die bisher letzte Version 1.2. Diese wurde am 10. November 2003 zum ISO-Standard ISO/IEC 15948:2003 erhoben und gleichzeitig zur zweiten Ausgabe der W3C-Empfehlung.

Eigenschaften

- Farbmodi und Präzision
  Das PNG-Format ermöglicht Graustufen-, Vollfarb- und Farbpaletten-Modus, sowie einen Graustufen- und einen Farbmodus mit Alpha-Kanal (Farb-Typen 0, 2, 3, 4 und 6).
- Farbtiefen
  Bei Graustufenbildern kann die Auflösung 1, 2, 4, 8 oder 16 Bit pro Pixel betragen, bei Farbbildern 8 (RGB8) oder 16 Bit (RGB16) pro Farbkanal. Farbbilder können alternativ mit dem Farbpalettenmodus mit bis zu 256 indizierten Farben gespeichert werden. Die indizierten Farben sind aus dem vollen RGB8-Spektrum frei wählbar.
- Transparenz
  PNG-Dateien können Transparenzinformationen enthalten, entweder in Form eines Alphakanals, als einzelne transparente

Farbe oder als ergänzende Transparenzpalette zu einer vorhandenen Farbpalette, die zu jeder Palettenfarbe einen Transparenzwert enthält. Ein Alphakanal ist eine zusätzliche Information, die für jedes Pixel angibt, wie viel vom Hintergrund des Bildes durchscheinen soll (0 entspricht 100 % Durchscheinen, 255 bzw. 65535 entspricht 0 % Durchscheinen). PNG unterstützt Alphakanäle mit 8 oder 16 Bit, was 256 beziehungsweise 65.536 Abstufungen der Transparenzstärke entspricht. Das PNG-Format erlaubt somit, unabhängig vom Hintergrund die Kanten von Text und Bildern zu glätten. Man kann echte Schlagschatten verwenden, die im Hintergrund ausblenden, oder Bilder erzeugen, die beliebig geformt erscheinen — wenn das Anzeigeprogramm das PNG-Format beherrscht.

- Metadaten und Datenblöcke
  PNG-Dateien sind aus verschiedenen Datenblöcken (englisch "chunks") mit jeweils unterschiedlichen Funktionen aufgebaut, die durch eine Zeichenkette aus vier Buchstaben gekennzeichnet werden (beispielsweise tEXt für textuelle Informationen). Neben den Pflicht-Blocktypen IHDR, IDAT, PLTE und IEND, die jede Implementierung unterstützen muss, sind weitere optionale Datenblöcke standardisiert. Diese können Metadaten zu den Bildinhalten und andere Zusatzinformationen enthalten, etwa zur Farbkorrektur. Diese können mit Programmen wie TweakPNG bearbeitet werden. Anwendungen können auch private Datenblöcke für eigene Zwecke definieren. Adobe Fireworks verwendet PNG als Anwendungsformat und nutzt private Datenblöcke, um darin verschiedene weitere Informationen abzulegen. Dabei besteht allerdings auch Verwechslungsgefahr mit den wesentlich kleineren, normalen PNG-Dateien
- Komprimierung
  Die verlustfreie Datenkompression in PNG basiert auf mehreren, teils optionalen Verarbeitungsschritten. Zuerst können mit einem Vorfilter die Werte benachbarter Bildpunkte dekorreliert werden, um sie besser komprimierbar über eine Differenz zu Nachbarwerten beschreiben zu können. Dann kann mit einer Substitutionskompressionsmethode versucht werden, wiederkehrende Bildmuster zu erkennen und durch kürzere Rückverweise auf ein vorheriges Auftreten zu ersetzen. Abschließend wird eine Entropiekodierung angewendet, die Auftrittswahrscheinlichkeiten einzelner Werte ausnutzt, indem sie die Werte nach Wahrscheinlichkeit sortiert durch Codes variabler Länge ersetzt.
- Dekorrelation
  Ein PNG-Bild mit 256 Farben, das dank Vorfilter nur 251 Byte groß ist. Das gleiche Bild als GIF-Datei wäre mehr als dreizehnmal so groß. In der Regel korreliert der Farbwert eines Bildpunktes mit Werten von Nachbarpunkten, das heißt es besteht eine Abhängigkeit oder Ähnlichkeit. Um diese Korrelationen auszunutzen, unterstützt PNG Vorfilter, die die Ausgangsdaten zunächst dekorrelieren. Dadurch werden Bildpunkte über die Differenz zu Nachbarpunkten beschrieben (Delta-Kodierung). Zu jeder Bildzeile kann eine von 5 Filtermöglichkeiten bestimmt werden (siehe unten). Die Auswahl erfolgt aus Geschwindigkeitsgründen oft heuristisch. Diese Filter ersetzen auf umkehrbare Weise die Farbwerte der Bildpunkte durch (ebenso viele, ebenso große) Differenzwerte. Dieses Differenzsignal hat in der Regel eine wesentlich geringere Dynamik, also Werte mit im Schnitt kleineren Beträgen. Diese sind von der abschließenden Entropiekodierung effektiver zu komprimieren. Je uniformer die Bildinhalte ausfallen, desto gewinnbringender funktioniert dieser Mechanismus. Bei der Dekodierung werden nach der Dekomprimierung der Daten umgekehrte Versionen der Filter angewandt, um die eigentlichen Bilddaten wiederherzustellen. Diese Möglichkeit ist einer der Gründe für die geringe Größe von PNG-Dateien. Manche Kodierer probieren zur Verbesserung der Kompression mehrere Filter durch. Dies ist besonders bei den zahlreichen Werkzeugen zur Optimierung der PNG-Kompression eine gängige Technik. In vielen Fällen bietet der nach seinem Erfinder Alan W. Paeth benannte Paeth-Predictor die besten Ergebnisse. Mit diesem wird versucht, aus den links, oben und linksoben benachbarten Bildpunkten automatisch den ähnlichsten für die Differenzbildung zu nutzen. Die Funktion wählt den Bildpunkt, der links+oben-linksoben am nächsten kommt.
- Substitutionskompression und Entropiekodierung
  erfolgen nach dem populären Deflate-Verfahren, da dieses ohne Belastung durch Softwarepatente frei verwendbar ist. Es umfasst Substitutionskompression nach Storer, Szymanski, Lempel und Ziv (LZSS-Algorithmus) und Entropiekodierung nach Huffman. Viele Programme binden für die Deflate-Kodierung und -Dekodierung (Codec) die freie Deflate-Bibliothek zlib ein, welche ursprünglich extra für PNG geschaffen wurde. Die Deflate-Komprimierung kann üblicherweise (wie auch in anderen Anwendungen – beispielsweise bei der ZIP-Kompression) im Ausgabeprogramm in 10 Stufen von 0 (keine) bis 9 (beste) eingestellt werden. Bislang ist Deflate die einzige unterstützte Methode. Es ist aber absichtlich Raum für Erweiterungen gelassen worden, um in zukünftigen PNG-Versionen auch andere, effizientere oder schnellere Algorithmen zu unterstützen. Um Abwärtskompatibilität zu existierenden PNG-fähigen Programmen zu gewährleisten, ist derzeit eine Aufnahme anderer Verfahren in den Standard jedoch nicht geplant.
- Kompatibilität

- Farbprofile
  Bis auf Safari und Firefox (nach Aktivierung) unterstützt bisher kein Browser eingebettete Farbprofile (iCCP-Blöcke). Sie bieten daher kein vollständiges Farbmanagement. Dadurch, dass Safari als einziger Browser auch vollständige Farbprofile wiedergibt, ist eine einheitliche und plattformübergreifende Darstellung bei Bildern im PNG-Format mit eingebettetem Farbprofil zurzeit nicht möglich. Zumindest für den Browser Firefox ist diese Funktion aber für die Zukunft geplant. Eingebettete Gammakorrekturwerte (gAMA-Blöcke) hingegen werden von den meisten aktuellen Browsern fehlerfrei erkannt und verarbeitet.
- Transparenz
  Der Microsoft Internet Explorer hatte bis zur Version 6 Probleme mit der Darstellung von transparenten PNG-Dateien mit Alphakanal. PNG-Dateien mit binärer ("ja/nein"-)Transparenz wurden jedoch fehlerfrei dargestellt. Es gibt aber Umgehungslösungen zur Nutzung des Alphakanals in älteren Versionen des Internet Explorers. Die neuesten Versionen der Browser Mozilla Firefox, Konqueror, Safari und Opera sowie der Internet Explorer ab der Version 7 unterstützen PNG jedoch weitgehend fehlerfrei.
- Rechtliches
  Das PNG-Format unterliegt keiner Patentbeschränkung. Jeder Softwarehersteller kann daher ohne Zahlung von Lizenzgebühren Programme veröffentlichen, die PNG lesen und schreiben.

[...]"

(aus https://de.wikipedia.org/w/index.php?title=Portable_Network_Graphics&oldid=164599203, 17Apr'17)

30.9 .mpeg File Format

Die **Moving Picture Experts Group (MPEG)** (engl."Expertengruppe für bewegte Bilder") ist eine Gruppe von Experten, die sich mit der Standardisierung von Videokompression und den dazugehörenden Bereichen, wie Audiodatenkompression oder Containerformaten, beschäftigt. Umgangssprachlich wird mit "MPEG" meistens nicht die Expertengruppe, sondern ein spezieller MPEG-Standard bezeichnet. Drei- oder viermal jährlich kommt die MPEG in fünftägigen Treffen zusammen. Etwa 350 Experten aus 200 Unternehmen und Organisationen aus 20 Ländern nehmen an

diesen Treffen, den MPEG-Meetings, teil. Die Gruppe wurde 1988 von Leonardo Chiariglione, damals Vizepräsident der Media Group des italienischen Forschungszentrums CSELT, gegründet. Er trat im Juni 2020 zurück. Die MPEG ist ein Teil des ISO/IEC JTC1/SC29 (International Organization for Standardization/International Electrotechnical Commission, Joint Technical Committee 1, Subcommittee 29) und dort seit Juni 2020 in verschiedene Arbeitsgruppen aufgeteilt (vormals "Working Group 11"). Die Standards werden mit der Internationalen Fernmeldeunion (ITU) abgeglichen und größtenteils in gemeinsamen Arbeitsgruppen entwickelt. Prominentestes Beispiel ist der MPEG-4 AVC Standard, der im Wortlaut identisch als ITU-T H.264 verabschiedet wurde.

Name	Erscheinungsjahr	Bemerkung
H.261	1988	H.261 wurde entwickelt ohne Zusammenarbeit mit MPEG; Bildtelefonie, Videokonferenzen über ISDN
MPEG-1	1993	Progressives Video-Format mit mehreren Layern. Wird unter anderem für Video-CDs verwendet. Zum Audio-Teil von MPEG-1 gehört MP3 (MPEG-1 Layer 3)
MPEG-2 /H.262	1994/95	Video- und Tonformate in Fernsehqualität. Wird auch für DVD-Videos und DVB verwendet. (Bitrate: bis 15 Mbit/s)
H.263	1995/96	H.263 wurde entwickelt ohne Zusammenarbeit mit MPEG
MPEG-3	(nie erschienen)	Hätte der Standard für HDTV werden sollen. Es genügte aber eine Erweiterung von MPEG-2. MPEG-3 wurde daher nicht mehr verabschiedet.
MPEG-4	1998/99/2000/01	ISO/IEC 14496: Gegenüber MPEG-2 deutlich stärkere Video-Kompression. Die erste Version des Standards kam 1998 heraus. Es folgte Version 2 1999/00 und die Version 3 2001. MPEG-4 beschreibt u. a. ein komplexes, an QuickTime angelehntes Container-Format, eine 3D-Sprache ähnlich VRML und nicht-rechteckige Video-Objekte. Es enthält auch Unterstützung für Digital Rights Management.
MPEG-4 AVC /H.264	2002	H.264: Offizielle Terminologie des ITU. MPEG-4-AVC oder MPEG-4 Teil 10 (offizielle MPEG Terminologie) ISO/IEC 14496-10 AVC. Zu Beginn (2002) auch H.26L genannt. Die Arbeit des JVT oder "JVT CODEC", Advanced Video Code (AVC), JM2.x, JM3.x und JM4.x genannt.
MPEG-7	2002	Ein System zur Beschreibung von multimedialen Inhalten (u. a. Metadaten). Katalogisierung, Inventarisierung und Wiederfindung von multimedialen Daten sind die zentralen Schlagworte.
MPEG-21		ISO/IEC 21000: Ein sog. "Multimedia Framework".

MPEG standardisiert nur den Bitstream (Abfolge der Bits) und den Dekodierer (als sogenannte Terminal-Architektur). Der Kodierer wird nicht standardisiert, so dass Raum für Effizienzsteigerungen bleibt. Es werden Musterimplementierungen (verification models) vorgeschlagen, die aber weder besonders schnell noch besonders effizient sind, da sie lediglich die Machbarkeit zeigen. Daher schreiben kommerzielle Anbieter die Implementierungen von MPEG-Kodierern von Grund auf neu, um entweder effizientere, qualitativ bessere Umsetzung des Originalmaterials in den codierten Datenstrom oder eine schnellere Implementierung zu erreichen. Die MPEG spezifiziert sowohl Containerformate als auch Codecs. Dadurch können beispielsweise in MPEG-2 kodierte Videospuren auch in (technisch allerdings unterlegenen) AVI-Containern abgelegt werden und nicht nur in hauseigenen MPEG-Containern.

MPEG-1 (ISO/IEC 11172) ist ein Standard der Moving Picture Experts Group (MPEG) zur verlustbehafteten Video- und Audiodatenkompression. MPEG-1 wurde in den 1980er-Jahren entwickelt (1991 vorgestellt) und hat das Ziel, Filme auf die beschränkte Datenrate einer mit normaler Geschwindigkeit abgespielten Audio-CD zu komprimieren (bis 1,5 Mbit/s). Das Ergebnis, mit dementsprechend eher bescheidener Qualität, wird Video-CD genannt. Die Video-Kompression von MPEG-1 wurde 1994 durch MPEG-2 deutlich verfeinert und verbessert.

Videokodierungsverfahren
Das Bildformat von MPEG-1 ähnelt dem JPEG-Format. Die Parameter werden allerdings genau festgelegt:

- Bildgröße maximal 768 × 576 Pixel
- Verhältnis der Höhe zu Breite der Pixel (14 Seitenverhältnisse definiert)
- Bildwechselfrequenz in Hertz
- Bilder liegen im YCbCr-Format als 3×8-bit-Werte pro Pixel vor:
- Y: Luminanz/Helligkeitskomponente (16: schwarz, 223: weiß)
- Cr: Rot-Grün-Farbdifferenzkomponente (−112: grün, +112 rot)
- Cb: Blau-Gelb-Farbdifferenzkomponente (−112: gelb, +112 blau)

Bildtyp	Zweck	Kompression
Intra-Bild (englisch intra coded picture, I-frame)	Ein I-Bild entspricht einem Standbild. Es dient als Anker für den wahlfreien Zugriff.	gering (ähnlich wie bei JPEG, jedoch in Echtzeit)
P-Bild (englisch predictive coded picture, P-frame)	P-Bilder benötigen Informationen von vorausgegangenen I-Bildern oder P-Bildern.	größere Kompression als bei I-Bildern
B-Bild (englisch bidirectional coded picture, B-frame)	B-Bilder sind abhängig von vorausgegangenen und folgenden I-Bildern oder P-Bildern.	größte Kompression
D-Bild (englisch DC direct coded picture, D-frame)	D-Bilder dienen dem schnellen Vorlauf.	nur ein Farbwert wird pro 8×8-Block gespeichert

Bei der Bildverarbeitung werden die Bilder unterschiedlich stark komprimiert und zu unterschiedlichen Zwecken genutzt. Die I-Bilder werden unabhängig von anderen Bildern komprimiert – sie benötigen am meisten Speicherplatz, lassen sich aber unabhängig von vorangegangenen Bildern dekodieren. Daher sind sie notwendig, um (nahezu) beliebig in einem Video springen zu können. Andere Bilder werden in Abhängigkeit von den anderen Bildern in dem Videostrom kodiert und benötigen dadurch weniger Speicherplatz. Diese Bildtypen werden dann abhängig vom Encoder, dessen Einstellungen und gelegentlich auch vom Bildmaterial unterschiedlich häufig verwendet und treten typischerweise zyklisch als sogenannte Bildergruppe (englisch Group of Pictures, GoP) auf. Eine Gruppe reicht dabei von einem I-Bild zum nächsten. Die Gruppen haben häufig eine Länge von einer halben Sekunde.

Audiokodierungsverfahren
Teil des Standards sind auch drei Audiokodierungsverfahren. In ansteigender Komplexität und Qualität sind das Layer 1, 2 und 3. Der Audio Layer 1 – auch bekannt als MP1 – wurde von Philips als Low-Complexity-Variante des Audio Layer 2 in den Standard eingebracht. Die digital compact cassette von Philips, die gleichzeitig mit Sonys MiniDisc auf den Markt kam und mittlerweile nicht mehr hergestellt wird, nutzte dieses Verfahren mit einer Datenrate von 384 kbps. Der Audio Layer 2 – auch bekannt als MP2 oder Musicam – war der etablierte Standard im Radiowesen. Nahezu alle professionellen digitalen Zuspielgeräte verwendeten MPEG-1 Audio Layer 2 in der Kompression von 256 kbit/s (128 kbit pro Sekunde und Kanal), da es sich dann leicht über die in Europa gut verbreitete ISDN-Infrastruktur übertragen ließ. Audio Layer 2 wird auch auf Video-CDs und Super-Video-CDs sowie (selten und nur für Europa zugelassen) auf DVDs und beim digitalen Fernsehen eingesetzt.

Der Audio Layer 3 – besser bekannt als MP3 – wurde von der Fraunhofer-Gesellschaft und anderen entwickelt und war, wie auch Layer 1 und Layer 2, nicht lizenzfrei. Das heißt, Hersteller, die einen Encoder für MP3 entwickeln und verkaufen wollten, mussten dafür Lizenzgebühren entrichten. Nicht kommerziell vertriebene Encoder (wie etwa LAME) sind lizenzkostenfrei. Das vergleichsweise gute Verhältnis von Größe einerseits und Qualität andererseits hat in den 1990er-Jahren zu einem Siegeszug des MP3-Formats geführt. Es war Basis des Aufblühens von Online-Tauschbörsen (wie Napster) und mobilen Musikabspielgeräten (MP3-Spieler) auf Basis von Flash-Speichern oder Festplatten. MP3 wurde zu den nachfolgenden Advanced-Audio-Coding-Standards (aus MPEG-2 und MPEG-4) weiterentwickelt.

30.10 Dithering in Bildverarbeitung

Das Dithering (von englisch *to dither* ≡'schwanken','zittern') ist eine Technik in der Computergrafik, um bei Bildern, die aufgrund technischer Restriktionen mit verringerter Farbtiefe wiedergegeben werden müssen, die Illusion einer größeren Farbtiefe zu erzeugen. Dithering wird auch als Fehlerdiffusion bezeichnet.

Bei einem *'geditherten'* Bild werden die fehlenden Farben durch eine bestimmte Pixel-Anordnung aus verfügbaren Farben nachgebildet und dadurch harte Übergänge zwischen den Farben vermieden. Das menschliche Auge nimmt das Dithering als Mischung der einzelnen Farben wahr.

Die häufigste Verwendung findet Dithering bei der Farbreduktion. Aus der Ferne betrachtet sind die Unterschiede kaum wahrnehmbar.

Verfahren

Für das Dithering gibt es eine Vielzahl an unterschiedlichen Algorithmen, die oft auch mit einer Fehlerstreuung arbeiten. Auftretende Rundungsfehler oder Überläufe werden dabei auf benachbarte Bildpunkte übertragen, um eine feinere (=glattere) Darstellung zu erreichen. Beispiele sind:
- Floyd-Steinberg
- Jarvis-Algorithmus
- Stucki

30.11 Dithering in Audiotechnik

Dithering (engl. für „Zittern") beschreibt eine Methode, die in der digitalen Audiotechnik die Wirkung von Quantisierungsfehlern abmildern kann. Diese Fehler treten sowohl bei der Digitalisierung als auch bei digitalen Rechenoperationen mit Signalen auf. Statt störender Verzerrungen entsteht ein weniger störendes gleichmäßiges Rauschen, das dem eines analogen Verstärkers ähnelt.

Das Problem: **systematische Rundungsfehler**

Ohne Dithering wird das als Quantisierungsrauschen bezeichnete Fehlersignal seinem Namen nur gerecht, falls sich das Nutzsignal von Sample zu Sample typischerweise um mehrere Quantisierungsstufen verändert, also bei hoher Aussteuerung mit nicht zu kleinen Frequenzen. Dann sind aufeinander folgende Fehler voneinander statistisch unabhängig, sodass das Quantisierungsrauschen sich als breitbandiges weißes Rauschen darstellt. Als relativ zum Nutzsignal kleiner Untergrund wird es kaum wahrgenommen, siehe typische System-Dynamikwerte. Mit abnehmender Aussteuerung fällt nicht nur die Maskierung des Rauschens durch das Nutzsignal geringer aus, siehe Signal-Rausch-Verhältnis, sondern für typische, geordnete Nutzsignale wie Musik treten Korrelationen des Quantisierungsfehlers auf; insbesondere haben aufeinander folgende Quantisierungsfehler abschnittsweise gleiche Vorzeichen, sodass sich die Rauschenergie zu niedrigeren Frequenzen verlagert und deutlich hörbar in Spektrallinien konzentriert. Diese Linien sind Oberwellen und Intermodulationen der Frequenzen im Nutzsignal. Schuld an diesen Verzerrungen ist letztlich die Nichtlinearität der Quantisierungskennlinie.

Eine Lösung: **zufällig auf- oder abrunden**

Durch Zugabe von geeignetem Dither-Rauschen zum Signal erhöht sich zwar die Rauschenergie, aber die Kennlinie wird linearisiert, sodass die Verzerrungen verschwinden und sehr leise Signalanteile besser (oder überhaupt erst) wahrgenommen werden. Das Dither-Rauschen besteht aus zufälligen Werten aus einem Bereich von der Größenordnung einer Quantisierungsstufe. Es werden verschiedene Wahrscheinlichkeitsverteilungen (Wahrscheinlichkeitsdichte 'probability density function' (PDF)) für die Rauschwerte angewandt. Das minimale Rauschen (bezüglich der Varianz der PDF), das für eine vollständige Linearisierung der Kennlinie hinzugefügt werden muss, ist gleichverteilt (rechteckig, RPDF-Dither) über genau eine Stufenhöhe. Dadurch wird bei der Quantisierung zufällig auf- oder abgerundet, mit einer Wahrscheinlichkeit, die linear davon abhängt, auf welcher Höhe der Signalwert zwischen zwei Stufen steht. 'RPDF-Dither' wirkt wie eine Pulsweitenmodulation (PWM), bis auf den Zufallsaspekt. Auf diesen kann in Verbindung mit Überabtastung verzichtet werden, wenn die dadurch entstehenden Artefakte bei unhörbar hohen Frequenzen liegen (vgl. Sigma-Delta-Wandler und Rauschformung).

Anwendung

Dithering findet in verschiedenen Bereichen der digitalen Audiotechnik Anwendung, und zwar immer, wenn das Signal bearbeitet wird und danach die Bittiefe reduziert wird und/oder die Samplingrate verändert wird: Beim Mastering, beim Erstellen von Audiodateien (WAV oder MP3) von DVDs, beim Verwenden älterer Sampler mit eingeschränkter Dynamik (zum Beispiel 8 Bit), oder auch nach der Bearbeitung mit Effektgeräten. Als Ergebnis ist oft ein Signal in CD-Qualität mit üblichen 16 Bit und 44,1 kHz gewünscht.

Anstatt die "überflüssigen" Bits einfach abzuschneiden ("Truncation") oder zu runden (round to even), wenn z. B. ein 24-Bit-Signal in ein 16-Bit-Signal gewandelt wird, sollte Dithering betrieben werden, da sonst die oben beschriebene Problematik des Quantisierungsfehlers auftritt. Anschaulich betrachtet führt das Rauschen in Verbindung mit einer Rundung zu einem Signal, das in langfristiger Betrachtung dem vorherigen Wert entspricht. Durch Filterung und Mittelwertbildung können damit die fehlenden Bits teilweise rekonstruiert werden. Effektgeräte und Audio-Software arbeiten intern oft mit deutlich höher auflösenden 32-bit-Integer- oder 32 bit Gleitkommazahlen. Bei der Rückwandlung sollte dann Dithering verwendet werden.

Am Beispiel einer Bittiefenreduktion von einer 24-Bit-Wortlänge auf die üblichen 16-Bit-Wortlängen einer Audio-CD wird messbar, dass Dithering gerade bei niedrigen Pegeln wesentlich mehr Information ermöglicht und somit den Dynamikumfang des Nutzsignals vergrößert. Es gibt einen Rauschteppich, allerdings ist dieser in großem Abstand zum Nutzsignaln nicht mehr mit diesem korreliert und wird psychoakustisch nicht mehr als auffällige Verzerrung des Nutzsignals wahrgenommen.

Bei der Verwendung hoher Abtastraten lässt sich das Rauschen in höhere Bereiche des Spektrums verschieben, sodass dieses bei einer späteren Wandlung gänzlich gefiltert wird. Anschaulich lässt sich dieser Prozess damit beschreiben, dass feinstufiger gezittert wird. Tieffrequente Anteile in den Signalen lassen sich damit praktisch vollständig rekonstruieren.

(aus https://de.wikipedia.org/wiki/Dithering(Bildbearbeitung) und https://de.wikipedia.org/wiki/Dithering(Audiotechnik), 17.Apr'17)

31 numerische Integration

Die Technik der *numerischen Integration* wird auch *'FE' Finite Elements* genannt. Dabei ist das *endliche ('finite') 'delta_t'* das 'finite Element'. Im Gegensatz dazu ist beim symbolischen Integral $\int f(t)\,dt$ das *'dt'* unendlich klein (*'infinit'*) — deswegen spricht man ja von *'Infinitesimalrechnung'*.

Wir *'digitalisieren'* hier die Berechnung (anstelle eines Signals) und erzielen auch die Ungenauigkeiten (*'Quantisierungsfehler'*) wie beim Sampling (Abtastung), so als hätten wir das wirklich exakte Ergebnis (einen Funktionsverlauf) nachträglich digitalisiert:

- zu großes 'delta_t' ist wie Unterabtastung
- die begrenzte Zahlenauflösung von 'float' und 'double' wirkt wie eine begrenzte ADC-Auflösung
- nur rechnen wir mit ungenauen Zwischenergebnissen weiter und machens no schlimmer ...

Aber es geht elektrisch!
Man braucht nur die Aufgabe programmieren — Rechnen musses der Computer.

(Wegen der im Folgenden gezeigten Akkumulation von numerischen Ungenauigkeiten und Rundungsfehlern liebe iXH, ihn *'Complutzger'* zu nennen :^))

31.1 ProgrammCode numInt1.c

Listing 57: numerical integrator C code

```
/*   dic4anumerIntDbl01.c: n12aganst@15Apr16
  *  compile: 'gcc dic4anumerInt01.c -o numInt1.bin -lm'
   * run:     './numInt1.bin'
    *stop:    Ctl+'C'  (== Strg+'C')    */
#include <stdio.h>
#include <math.h>        //f. sin(), M_PI, M_PIl...
#include <time.h>        //f. clock_gettime, timespec
int main(){
  float       intF;
  double      intL,  tdiff;
  long double intLL, tLL, dtLL=1.0, dyLL, m2pi=M_PI+M_PI;
  struct timespec von, bis;
  printf("numerical integration of int{ sin(t)*dt }\n");
  printf("dt           cputime [s]  intF                intL                intLL\n");
  printf("------------ ------------ ------------------- ------------------- -------------------\n")
      ;
  for( ;; dtLL/=2.0 ){                          //redo_with_decreasing_dt:
    clock_gettime(CLOCK_MONOTONIC, &von);       //take clock reading
    printf("%12.10llf", dtLL );
    intF=0.0F; intL=0.0;    intLL=0.0L;
    for( tLL=0. ; tLL < m2pi; tLL+= dtLL){      //  run_the_integral
      dyLL  = ( sin(tLL)*dtLL );
      intF  += dyLL;
      intL  += dyLL;
      intLL += dyLL;
    }
    clock_gettime(CLOCK_MONOTONIC, &bis);       //take clock reading
    tdiff= (bis.tv_sec-von.tv_sec) + (bis.tv_nsec-von.tv_nsec)/1000000000.0;
    printf(":%011.6lf %20.17lf %20.17lf %23.20Lf\n", tdiff, intF, intL, intLL );
  }
}
```

31.2 numInt1.bin Output

Listing 58: numInt1.bin output

```
$ ./numInt1.bin
numerical integration of int{ sin(t)*dt }
dt           cputime [s]  intF                intL                intLL
------------ ------------ ------------------- ------------------- -------------------
1.0000000000:0000.000082 -0.10325383394956589 -0.10325384847654717 -0.10325384847654708875
0.5000000000:0000.000014 -0.03085740469396114 -0.03085742504472100 -0.03085742504472092917
0.2500000000:0000.000022 -0.00359983788803220 -0.00359969089790729 -0.00359969089790739688
0.1250000000:0000.000040 -0.00152407283894718 -0.00152383634883839 -0.00152383634883823584
0.0625000000:0000.000076 -0.00048636054270901 -0.00048644797911804 -0.00048644797911834552
0.0312500000:0000.000148 -0.00002863035297196 -0.00002836660185320 -0.00002836660185321666
0.0156250000:0000.000375 -0.00001302143482462 -0.00001324690964771 -0.00001324690964762630
0.0078125000:0000.000783 -0.00000634084017292 -0.00000568709212102 -0.00000568709212107982
0.0039062500:0000.001564 -0.00000198801740225 -0.00000190719050218 -0.00000190719050168115
0.0019531250:0000.002516  0.00000241318548433 -0.00000001724147767 -0.00000001724147793520
0.0009765625:0000.004632 -0.00000135774598675 -0.00000000854137257 -0.00000000854137027375
0.0004882812:0000.009740  0.00000011084119933 -0.00000000419132757 -0.00000000419131645238
```

```
0.0002441406:0000.024267 -0.00000684441147314 -0.00000000201627522 -0.0000000201628953894
0.0001220703:0000.046118 -0.00000239473047259 -0.00000000092877722 -0.0000000092877609743
0.0000610352:0000.095524  0.00000017254924956 -0.00000000038503316 -0.0000000038501936015
0.0000305176:0000.186133 -0.00000225220423999 -0.00000000011311667 -0.0000000011314101236
0.0000152588:0000.373979  0.00000043245921688 -0.00000000001627977 -0.0000000001624955410
0.0000076294:0000.753931  0.00000019896361891 -0.00000000000651248 -0.0000000000648759141
0.0000038147:0001.507083 -0.00000004202514248 -0.00000000000155343 -0.0000000000160664044
0.0000019073:0003.039227 -0.00162730331066996 -0.00000000000047767 -0.0000000000040914553
0.0000009537:0006.067224 -0.00428860122337937 -0.00000000000002312 -0.0000000000009837656
0.0000004768:0012.024375 -0.01084140874445438 -0.00000000000031162 -0.0000000000002640035
0.0000002384:0024.047794 -0.04007273912429810 -0.00000000000014802 -0.0000000000000556012
0.0000001192:0048.092125 -0.05627871304750443  0.00000000000002565 -0.0000000000000159845
0.0000000596:0096.070992 -1.00000000000000000  0.00000000000323700 -0.0000000000000008858
0.0000000298:0192.629370 -0.50000000000000000 -0.00000000000677233 -0.0000000000000029821
0.0000000149:0384.740141 -0.25000000000000000 -0.00000000000784378  0.0000000000000044195
0.0000000075:0769.224315 -0.12500000000000000  0.00000000000518189 -0.0000000000000153791
0.0000000037:1538.340951 -0.06250000000000000  0.00000000000098455 -0.0000000000000293013
0.0000000019:3077.354070 -0.03125000000000000  0.00000000000456387 -0.0000000000000288706
```

31.3 Prinzip – Pseudocode

Listing 59: numerical integral pseudocode

```
main(){
  double
        t=0,
        delta_t = 0.1, /*zB. Sekunden*/
        t_max=10.0,    /*zB. Sekunden*/
        sum=0,
        y;
  t=0;
  while( t < t_max ){
    y = funktion(t) * delta_t;
    sum = sum + y;
    t = t + delta_t
  }
  printf("%g\n", sum);
}
double funktion( double t ){
  ...
}
```

31.4 zB. Mondraketenstart

Listing 60: moon rocket launch FEsim

```
/* Start der Saturn V Mondrakete,
   numerische FE-Simulation:
   Startgewicht: 3000 Tonnen
   Schub       : 3400 Tonnen
   Flugdauer   : 150 Sekunden
   Verbrauch   : 2000 Tonnen == 2000T/150s == 13300 kg/s */
#include <stdio.h>
main(){
  double
        t = 0.0,
        delta_t = 0.1,        /* [s] */
        t_max   = 150.0,      /* [s] */
        m       = 3000*1000, /* [kg] */
        a       = 0.0,        /* [m/s/s] 'acceleration' */
        g       = 9.81,       /* [m/s/s] 'geos'=='die Erde' */
        F       = 0.0,        /* F='Force' */
        v       = 0.0;        /* v='Velocity' */
  t=0;
  while( t < t_max ){
    F = 3400 * 1000 * 9.81;
    a = F/m - g;
    v = v + a*delta_t;  /* y = funktion(t) * delta_t */
                        /* sum = sum + y */
    m = m - 13300 * delta_t;
    t = t + delta_t;
  }
  printf("v(Brennschluss)=%g m/s == %g km/h\n", v, v*3.6);
}
```

Übungsaufgaben:

- ⊕ Rechne und protokolliere laufend die Höhe über Grund
- ⊕ Was muss man ändern, wenn man einen Katapult-Start mit einer Anfangsgeschwindigkeit von 1.0 [Mach] simulieren will?
- ⊕ Rechne es mit delta_t = 0.01 und 1.0 [s]
- ⊕ Baue die abnehmende Erdanziehung mit ein

⊕ in Wirklichkeit fliegt die gar nit (lang) senkrecht aui, sondern draht si in Richtung der Erdrotation, um 'Fliehkraft' zu gewinnen. Dersimuliersch des?

31.5 f(x) = f'(x)

$$f' ::= df/dx \tag{2a}$$

mit

$$f = f' \qquad (Angabe)$$

$$\rightarrow \quad f_{[x]} = df_{[x]}/dx$$

$$\rightarrow \quad f_{[x]} * dx = df_{[x]} \tag{2b}$$

ausserdem:

$$f_{[x]} = f_{[x-1]} + df_{[x-1]} \tag{2c}$$

und

$$x = (x-1) + dx \tag{2d}$$

ergibt den Algorithmus:

$$Loop: \quad df = f * dx$$
$$f = f + df$$
$$x = x + dx$$
$$gotoLoop \tag{2e}$$

Listing 61: f=f'

```
/* numIntf1.c, XH@QfB
   numerische FE-Simulation: f = f'
   compile: 'gcc numIntf1.c -o nf1'
   run:     './nf1'
 */
#include <stdio.h>
int Ausgabe(double,double);
int main(){
  double
        x       = 0.0,      /*Integral-Untergrenze*/
        x_max   = 2.0,      /*Integral-Obergrenze*/
        dx      = 0.01,
        f       = 1.0,      /*DGL-AnfangsBedingung*/
        f1, df;
  while( x < x_max ){
    Ausgabe(x,f);
    f1 = f;             /*Angabe*/
    df = f1*dx;         /*aus f1 = df/dx */
    f  = f+df;          /*logo*/
    x  = x+dx;          /*logo*/
  }
}
int Ausgabe(double x,double f){
  static int n=0; int i;
  if(0 == n++){
    printf("+---------1---------2---------3---------4---------5---------6---------7---------8----\n!
        ");
  }
  if(n%5)return;
  for(i=0; i<f*10; i++) putchar('.');
  printf("f=%8.5lf\n!", f);
}
```

Listing 62: f=f' output ASCIIart

```
+---------1---------2---------3---------4---------5---------6---------7---------8---
!. . . . . f= 1.04060
!. . . . . f= 1.09369
!. . . . . f= 1.14947
!. . . . . f= 1.20811
!. . . . . f= 1.26973
!. . . . . f= 1.33450
!. . . . . f= 1.40258
!. . . . . f= 1.47412
!. . . . . f= 1.54932
!. . . . . f= 1.62835
!. . . . . f= 1.71141
!. . . . . f= 1.79871
!. . . . . f= 1.89046
!. . . . . f= 1.98689
!. . . . . f= 2.08825
!. . . . . f= 2.19477
!. . . . . f= 2.30672
!. . . . . f= 2.42439
!. . . . . f= 2.54806
!. . . . . f= 2.67803
!. . . . . f= 2.81464
!. . . . . f= 2.95822
!. . . . . f= 3.10911
!. . . . . f= 3.26771
!. . . . . f= 3.43440
!. . . . . f= 3.60958
!. . . . . f= 3.79371
!. . . . . f= 3.98723
!. . . . . f= 4.19062
!. . . . . f= 4.40438
!. . . . . f= 4.62905
!. . . . . f= 4.86517
!. . . . . f= 5.11335
!. . . . . f= 5.37418
!. . . . . f= 5.64832
!. . . . . f= 5.93644
!. . . . . f= 6.23926
!. . . . . f= 6.55752
!. . . . . f= 6.89202
!. . . . . f= 7.24358
```

31.6 f = -f''

$$f^{(n)} ::= \frac{df^{(n-1)}}{dx} \qquad n'teAbleitung$$

bei

$$\frac{df(x)}{dx} ::= \lim_{dx\to 0} \frac{f(x+dx)}{dx} \qquad Grenzwert$$

$$f' ::= f^{(1)}(x) \qquad 1'teAbleitung$$

$$f'' ::= f^{(2)}(x) \qquad 2'teAbleitung$$

Variablen

$$f0, f ::= f^{(0)}(x) \qquad 0'teAbleitung$$

$$f1 ::= f^{(1)}(x) \qquad 1'teAbleitung$$

$$f2 ::= f^{(2)}(x) \qquad 2'teAbleitung$$

diskretisiert

$$f2_{[x]} = df1_{[x]}/dx$$

$$\rightarrow \quad f2_{[x]} * dx = df1_{[x]}$$

$$f1_{[x]} = df0_{[x]}/dx$$

$$\rightarrow \quad f1_{[x]} * dx = df0_{[x]}$$

natürlich

$$f1_{[x]} = f1_{[x-1]} + df1_{[x-1]}$$

und wieder

$$x = (x-1) + dx$$

Algorithmus:

$$Loop: \qquad f2 = (Formel, zB\ '\pi/2 + \sin xf0 - \rho * f1/x')$$

$$df1 = f2 * dx$$

$$f1 = f1 + df1$$

$$df0 = f1 * dx$$

$$f0 = f0 + df0$$

$$x = x + dx$$

$$EndLoop$$

Listing 63: f=-f''

```
/* numInt-f2.c, XH@QfB
   numerische FE-Simulation: f = -f''
   compile: 'gcc numInt-f2.c -o nf2'
   run:     './nf2'
 */
#include <stdio.h>
int Ausgabe(double,double);
int main(){
  double
        x       = 0.0,     /*Integral-Untergrenze*/
        x_max   = 12.56,   /*Integral-Obergrenze*/
        dx      = 0.01,
        f2,
        f1      = 0.0,
        df1, df,
        f       = 1.0;     /*DGL-AnfangsBedingung*/
  while( x < x_max ){
    Ausgabe(x,f);
    f2  = -f;            /*Angabe*/
    df1 = f2*dx;         /*aus f2 = df1/dx */
    f1  = f1+df1;        /*logo*/
    df  = f1*dx;         /*aus f1 = df/dx */
    f   = f+df;          /*logo*/
    x   = x+dx;          /*logo*/
  }
}
int Ausgabe(double x,double f){
  static int n=0; int i;
  if(0 == n++){
    printf("+---------1---------0.5-------0---------0.5-------1.0-------1.5-------2.0-------2.5-\n!
        ");
  }
  if(n%33)return;
  for(i=0; i<10+(1.0+f)*20; i++) putchar('.');
  printf("f=%8.5lf\n!", f);
}
```

Listing 64: f=-f'' output ASCIIart

```
+---------1---------0.5-------0---------0.5-------1.0-------1.5-------2.0-------2.5-
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.94766
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.79306
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.55287
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.25301
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . f=-0.07414
!. . . . . . . . . . . . . . . . . . . . . . . . f=-0.39330
!. . . . . . . . . . . . . . . . . . . f=-0.67001
!. . . . . . . . . . . . . f=-0.87442
!. . . . . . . . . . . f=-0.98446
!. . . . . . . . . . . f=-0.98827
!. . . . . . . . . . . . . f=-0.88542
!. . . . . . . . . . . . . . . . . f=-0.68702
!. . . . . . . . . . . . . . . . . . . . . . f=-0.41448
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . f=-0.09722
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.23054
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.53342
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.77874
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.94001
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.99985
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.95178
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.80100
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.56378
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.26572
!. . . . . . . . . . . . . . . . . . . . . . . . . . . f=-0.06101
!. . . . . . . . . . . . . . . . . . . . . . f=-0.38117
!. . . . . . . . . . . . . . . . . f=-0.66019
!. . . . . . . . . . . . f=-0.86796
!. . . . . . . . . . f=-0.98207
!. . . . . . . . . . f=-0.99019
!. . . . . . . . . . . . f=-0.89146
!. . . . . . . . . . . . . . . . f=-0.69653
!. . . . . . . . . . . . . . . . . . . . f=-0.42642
!. . . . . . . . . . . . . . . . . . . . . . . . . . f=-0.11031
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.21772
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.52225
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.77041
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.93544
!. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . f= 0.99952
```

32 PLL phase locked loop, phasenstarre Regelschleife

Die 'phasenstarre Regelschleife' (phase locked loop) erzeugt aus dem Vergleich der Phasenlagen zweier Signalfrequenzen ein Fehlersignal wie die klassische Regelung (control loop).

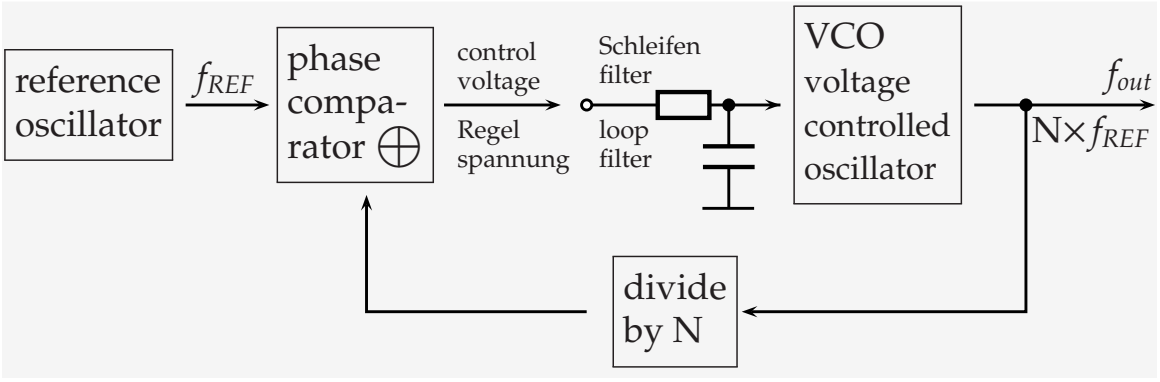
Man teilt nun die Frequenz eines *freilaufenden (free running)*, spannungsgesteuerten Oszillators (VCO, voltage controlled oscillator) um den Faktor $\frac{1}{N}$ 'herunter' und regelt dieses *phasenstarr* zu einem *stabilen Referenzsignal* nach.

- Man erhält die
  N-fache Ausgangsfrequenz mit der
  Stabilität des Referenz-Signals.
- Faktor N **variierbar**,
  → Ausgangsfrequenz **in Stufen** schaltbar
- Referenzfrequenz verstimmen,
  → ganze PLL *zieht* mit.
- Fehlerspannung mit NF-Signal (engl. AF) addieren,
  → Frequenzmodulation (FM) der PLL.
- umgekehrt: Frequenzmodulierte Referenz
  → Fehlersignal enthält demodulierte FM!

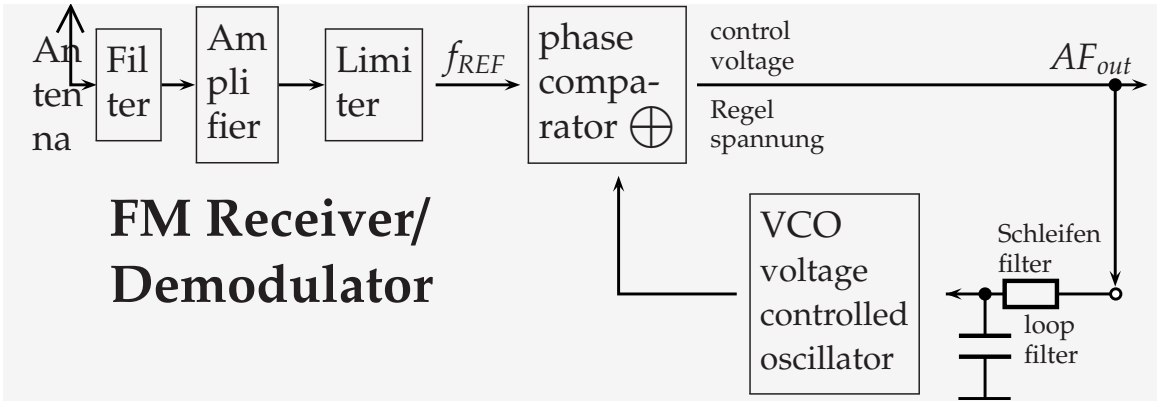
Die PLL wird deswegen gerne
a) zur Frequenzsynthese und
b) FM (de-)modulation
verwendet.

Begriffe:
die PLL ist *eingerastet*, *locked*, *fängt sich*

32.1 PLL schema


reference oscillator
f_{REF}
phase compa-rator
control voltage
Regel spannung
Schleifen filter
loop filter
VCO voltage controlled oscillator
f_{out}
N×f_{REF}
divide by N

32.2 FM demod:


An ten na
Fil ter
Am pli fier
Limi ter
f_{REF}
phase compa-rator
control voltage
Regel spannung
AF_{out}
FM Receiver/ Demodulator
VCO voltage controlled oscillator
Schleifen filter
loop filter

FAIRCHILD SEMICONDUCTOR™

April 1994
Revised April 1999

74VHC4046
CMOS Phase Lock Loop

General Description

The VHC4046 is a low power phase lock loop utilizing advanced silicon-gate CMOS technology to obtain high frequency operation both in the phase comparator and VCO sections. This device contains a low power linear voltage controlled oscillator (VCO), a source follower, and three phase comparators. The three phase comparators have a common signal input and a common comparator input. The signal input has a self biasing amplifier allowing signals to be either capacitively coupled to the phase comparators with a small signal or directly coupled with standard input logic levels. This device is similar to the CD4046 except that the Zener diode of the metal gate CMOS device has been replaced with a third phase comparator.

Phase Comparator I is an exclusive OR (XOR) gate. It provides a digital error signal that maintains a 90 phase shift between the VCO's center frequency and the input signal (50% duty cycle input waveforms). This phase detector is more susceptible to locking onto harmonics of the input frequency than phase comparator I, but provides better noise rejection.

Phase comparator III is an SR flip-flop gate. It can be used to provide the phase comparator functions and is similar to the first comparator in performance.

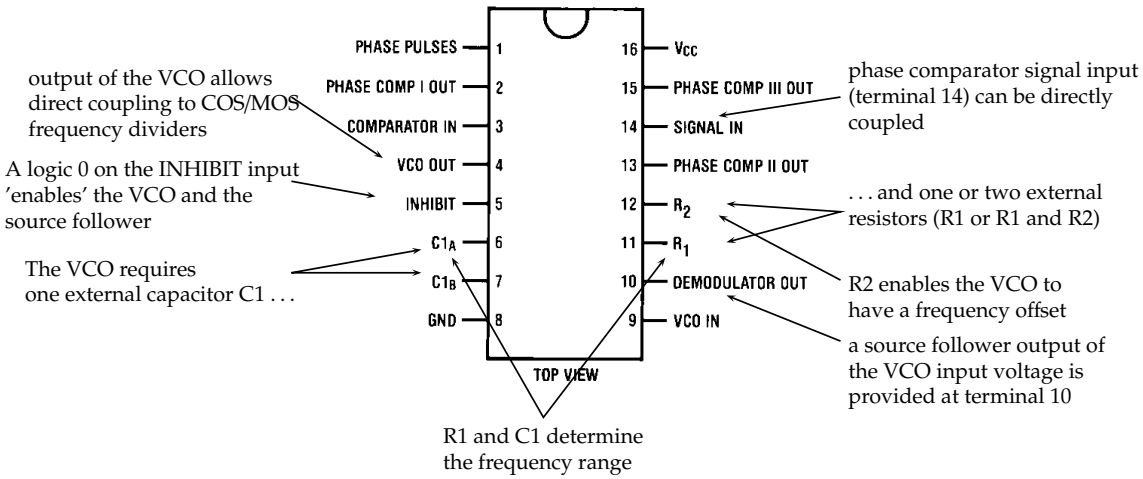
Phase comparator II is an edge sensitive digital sequential network. Two signal outputs are provided, a comparator output and a phase pulse output. The comparator output is a 3-STATE output that provides a signal that locks the VCO output signal to the input signal with 0 phase shift between them. This comparator is more susceptible to noise throwing the loop out of lock, but is less likely to lock onto harmonics than the other two comparators.

In a typical application any one of the three comparators feed an external filter network which in turn feeds the VCO input. This input is a very high impedance CMOS input which also drives the source follower. The VCO's operating frequency is set by three external components connected to the $C1_A$, $C1_B$, R_1 and R_2 pins. An inhibit pin is provided to disable the VCO and the source follower, providing a method of putting the IC in a low power state.

The source follower is a MOS transistor whose gate is connected to the VCO input and whose drain connects the Demodulator output. This output normally is used by tying a resistor from pin 10 to ground, and provides a means of looking at the VCO input without loading down modifying the characteristics of the PLL filter.

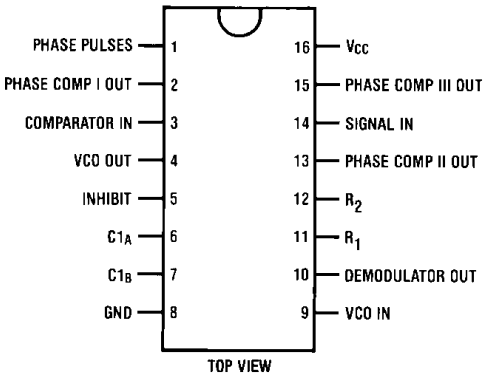
Features

- Low dynamic power consumption: (V_{CC} = 4.5V)
- Maximum VCO operating frequency: 12 MHz (V_{CC} = 4.5V)
- Fast comparator response time (V_{CC} = 4.5V)
  - Comparator I: 25 ns
  - Comparator II: 30 ns
  - Comparator III: 25 ns
- VCO has high linearity and high temperature stability
- Pin and function compatible with the 74HC4046

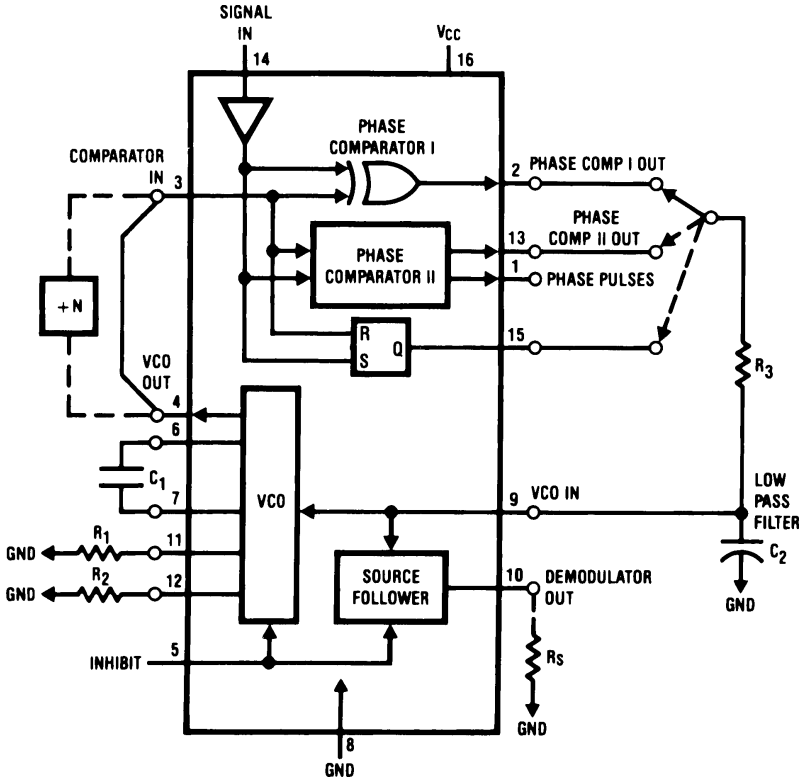


74VHC4046

Connection Diagram



Block Diagram



(from data sheets 'SGS-Thomson HCC/HCF4046B', 'FAIRCHILD SEMICONDUCTOR 74VHC4046':)

32.2.1 VCO Section

The VCO requires one external capacitor C1 and one or two external resistors (R1 or R1 and R2). Resistor R1 and capacitor C1 determine the frequency range of the VCO and resistor R2 enables the VCO to have a frequency offset if required.

The high input impedance ($10^{12}\Omega$) of the VCO simplifies the design of low-pass filters by permitting the designer a wide choice of resistor-to-capacitor ratios. In order not to load the low-pass filter, a source-follower output of the VCO input voltage is provided at terminal 10 (DEMODULATED OUTPUT). If this terminal is used, a load resistor (R_S) of $10k\Omega$ or more should be connected from this terminal to VSS . If unused this terminal should be left open. The VCO can be connected either directly or through frequency dividers to the comparator input of the phase comparators.

A full COS/MOSlogic swing is available at the output of the VCO and allows direct coupling to COS/MOS frequency dividers such as the HCC/HCF4024B, HCC/HCF4018B, HCC/HCF4020B, HCC/HCF4022B, HCC/HCF4029B,and HBC/HBF4059A. One or more HCC/HCF4018B (Presettable Divide-by-N Counter) or HCC/HCF4029B (Presettable Up/Down Counter), or HBC/HBF4059A (Programmable Divide-by-N Counter), together with the HCC/HCF4046B (Phase-Locked Loop) can be used to build a micropower low-frequency synthesizer.

A logic 0 on the INHIBIT input 'enables' the VCO and the source follower, while a logic 1 turns off both to minimize stand-by power consumption.

32.2.2 Phase Comparators

The phase-comparator signal input (terminal 14) can be direct- coupled provided the signal swing is within COS/MOS logic levels [logic '0' ≤ 30% (VDD–VSS), logic 1 ≥ 70% (VDD-VSS)]. For smaller swings the signal must be capacitively coupled to the self-biasing amplifier at the signal input. Phase comparator I is an exclusive-OR network; it operates analagously to an overdriven balanced mixer. To maximize the lock range, the signal- and comparator- input frequencies must have a 50% duty cycle. With no signal or noise on the signal input, this phase comparator has an average output voltage equal to VDD/2. The low pass filter connected to the output of phase comparator I supplies the averaged voltage to the VCO input, and causes the VCO to oscillate at the center frequency (f_0). The frequency range of input signals on which the PLL will lock if it was initially out of lock is defined as the frequency capture range ($2\ f_c$). The frequency range of input signals on which the loop will stay locked if it was initially in lock is defined as the frequency lock range ($2\ f_L$). The capture range is ≤ the lock range. With phase comparator I the range of frequencies over which the PLL can acquire lock (capture range) is dependent on the low pass filter characteristics, and can be made as large as the lock range. Phase-comparator I enables a PLL system to remain in lock in spite of high amounts of noise in the input signal. One characteristic of this type of phase comparator is that it may lock onto input frequencies that are close to harmonics of the VCO center frequency. A second characteristic is that the phase angle between the signal and the comparator input varies between 0° and 180°, and is 90° at the center frequency. Fig. (a) shows the typical, triangular, phase-to-output response characteristic of phase-comparator I. Typical waveforms for a COS/MOS phase-locked-loop employing phase comparator I in locked condition of f_0 is shown in fig. (b).

Phase-comparator II is an edge-controlled digital memory network. It consists of four flip-flop stages, control gating, and a three-stage output-circuit comprising p- and n- type drivers having a common output node. When the p-MOS or n-MOS drivers are ON they pull the output up to VDD or down to VSS, respectively. This type of phase comparator acts only on the positive edges of the signal and comparator inputs. The duty cycles of the signal and comparator inputs are not important since positive transitions control the PLL system utilizing this type of comparator. If the signal input frequency is higher than the comparatorinput frequency, the p-type output driver is maintained ON most of the time, and both the n- and p- drivers OFF (3 state) the remainder of the time. If the signal input frequency is lower than the comparator input frequency, the n-type output driver is maintained ON most of the time, and both the n- and p- drivers OFF (3 state) the remainder of the time. If the signal and comparator input frequencies are the same, but the signal input lags the comparator input in phase, the n-type output driver is maintained ON for a time corresponding to the phase difference. If the signal and comparator input frequencies are the same, but the comparator input lags the signal in phase, the p-type output driver is maintained ON for a time corresponding to the phase difference. Subsequently, the capacitor voltage of the low pass filter connected to this phase comparator is adjusted until the signal and comparator inputs are equal in both phase and frequency. At this stable point both p- and n- type output drivers remain OFF and thus the phase comparator output becomes an open circuit and holds the voltage on the capacitor of the low pass filter constant. Moreover the signal at the 'phase pulses' output is a high level which can be used for indicating a locked condition. Thus, for phase comparator II, no phase difference exists between signal and comparator input over the full VCO frequency range. Moreover, the power dissipation due to the low pass filter is reduced when this type of phase comparator is used because both the p- and n- type output drivers are OFF for most of the signal input cycle. It should be noted that the PLL lock range for this type of phase comparator is equal to the capture range, independent of the low pass filter. With no signal present at the signal input, the VCO is adjusted to its lowest frequency for phase comparator II. Fig. (c) shows typical waveforms for a COS/MOS PLL employing phase comparator II in a locked condition.

Figure a : Phase-Comparator I Characteristics at Low-Pass Filter Output.

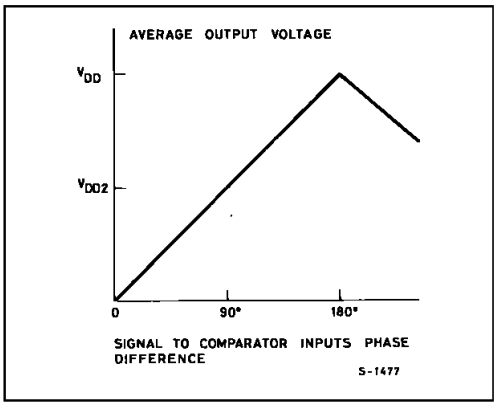


Figure b : Typical Waveforms for COS/MOS Phase Locked-Loop Employing Phase Comparator I in Locked Condition of f_o.

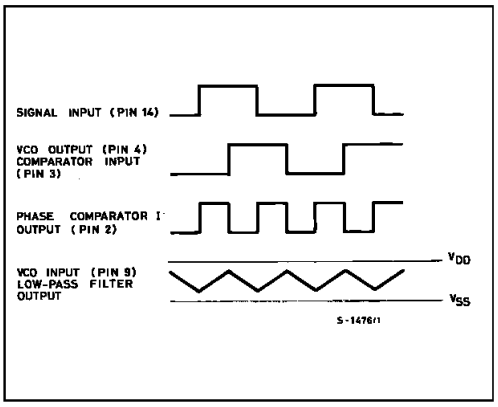
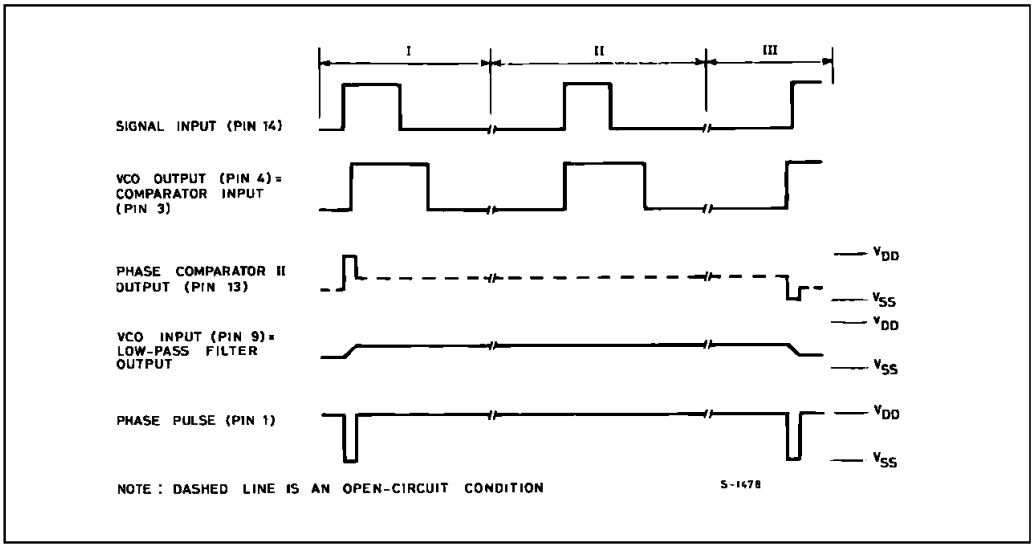


Figure C : Typical Waveforms For COS/MOS Phase-locked Loop Employing Phase Comparator II In Locked Condition.



Man sehe sich auch die 'DSPLL' ICs Si570/Si571 und Verwandte an.

33 ?

33.0.1 kT, kTB, $\sqrt{4kTBR}$

in Watt: [dBW]:
10*log(kT) = 10*ln(kt)/ln(10) = -203.930 dBW

in mW: [dBm]:

10*log(kt)+30 = -173.930 dBm $\equiv \frac{-174dBm}{Hz}$

bei 500Hz equiv.Bandbreite:
10*log(kt*500Hz)+30 = -146.940 dBm;

Rauschspannung eines R=50Ω ($U^2 = P * R$): $\sqrt{4 * k * T * B * 50\Omega}$

An R=50Ω Last (Anpassung ≡ halbe Spannung): $\sqrt{k * T * 50\Omega * 500Hz} \approx 10nV$

jedes (Mess-)Gerät mit 50Ω Eingang entspricht 50Ω Last
k=1.38E-23, T=293.15K

33.0.2 -174dBm/Hz

33.0.3 Bandbreite kontra MDS

34 Testen

34.1 White-Box-Test

Der Begriff White-Box-Test (seltener auch Glass-Box-Test) bezeichnet eine Methode des Software-Tests, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems entwickelt werden. Im Gegensatz zum Black-Box-Test ist für diesen Test also ein Blick in den Quellcode gestattet. D.h., es wird am Code geprüft.

Ein Beispiel für einen White-Box-Test ist ablaufbezogenes Testen (Kontrollflussorientierte Testverfahren), bei welchem der Ablaufgraph im Vordergrund steht. Qualitätskriterium des Tests ist es, sicherzustellen, dass Testfälle in Bezug auf die Überdeckung des Quellcodes gewisse Hinlänglichkeitskriterien erfüllen. Gängig sind dabei u. a. folgende Maße (bzw. Qualitätskriterien):

- Zeilenüberdeckung: Ausführung aller Quellcode-Zeilen
- Anweisungsüberdeckung bzw. Knotenüberdeckung: Ausführung aller Anweisungen
- Zweigüberdeckung bzw. Kantenüberdeckung: Durchlaufen aller möglichen Kanten von Verzweigungen des Kontrollflusses
- Bedingungsüberdeckung bzw. Termüberdeckung (mehrere Varianten): Durchlaufen aller möglichen ausschlaggebenden Belegungen bei logischen Ausdrücken in Bedingungen
- Pfadüberdeckung (mehrere Varianten): Betrachtung der Pfade durch ein Modul

Die Zahl der benötigten Testfälle für die einzelnen Maße unterscheidet sich z.T. deutlich. Kantenüberdeckung wird im Allgemeinen als minimales Testkriterium angesehen. Je nach Art und Struktur der zu testenden Software können andere Maße für ein System als Ganzes oder für Module sinnvoll sein.

Selbst wenn ein Softwaresystem in Bezug auf ein Hinlänglichkeitskriterium erfolgreich getestet wurde, schließt das nicht aus, dass es Fehler enthält. Dies liegt in der Natur des White-Box-Tests begründet und kann eine der folgenden Ursachen haben:

Der White-Box-Test leitet Testfälle nicht aus der Spezifikation des Programms her, sondern aus dem Programm selbst. Getestet werden kann nur die Korrektheit eines Systems, nicht, ob es eine geforderte Semantik erfüllt.

Auch wenn alle Programmpfade getestet worden sind, bedeutet dies nicht, dass ein Programm fehlerfrei arbeitet. Der Fall, dass im Graphen des Kontrollflusses Kanten fehlen, wird nicht erkannt.

Zusammenfassend kann man sagen, dass White-Box-Tests alleine als Testmethodik nicht ausreichen. Eine sinnvolle Testreihe sollte Black-Box-Tests und White-Box-Tests kombinieren. Nach der Überdeckungsmessung der Testfälle des Black-Box-Tests (durch ein geeignetes Werkzeug) werden durch Betrachten der nicht überdeckten Codeteile neue Testfälle aufgestellt, um die Überdeckung zu erhöhen.

Will man ein System auch in seinen Teilsystemen testen, benötigt man dazu Kenntnisse über die innere Funktionsweise des zu testenden Systems. White-Box-Tests eignen sich besonders gut, um in Erscheinung getretene Fehler zu lokalisieren, d. h., die fehlerverursachende Komponente zu identifizieren und als Regressionstest ein Wiederauftreten des Fehlers bereits in der Komponente zu vermeiden.

Weil die Entwickler der Tests Kenntnisse über die innere Funktionsweise des zu testenden Systems besitzen müssen, werden White-Box-Tests von demselben Team, häufig sogar von denselben Entwicklern entwickelt wie die zu testenden Komponenten. Spezielle Testabteilungen werden für White-Box-Tests in der Regel nicht eingesetzt, da der Nutzen speziell für diese Aufgabe abgestellter Tester meist durch den Aufwand der Einarbeitung in das System eliminiert wird.

34.2 Vergleich mit Black-Box-Tests

White-Box-Tests werden eingesetzt, um Fehler in den Teilkomponenten aufzudecken und zu lokalisieren, sind aber aufgrund ihrer Methodik kein geeignetes Werkzeug, Fehler gegenüber der Spezifikation aufzudecken. Für letzteres benötigt man Black-Box-Tests. Zu bedenken ist auch, dass zwei Komponenten, die für sich genommen korrekt gemäß ihrer jeweiligen Teilspezifikation arbeiten, zusammen nicht zwangsläufig eine korrekte Einheit gemäß der Gesamtspezifikation bilden. Dies kann durch Black-Box-Tests leichter festgestellt werden als durch White-Box-Tests.

Im Vergleich zu Black-Box-Tests sind White-Box-Tests wesentlich einfacher in der Durchführung, da sie keine besondere organisatorische Infrastruktur benötigen.

Vorteile von White-Box-Tests gegenüber Black-Box-Tests
: Testen von Teilkomponenten und der internen Funktionsweise
: Geringerer organisatorischer Aufwand
: Automatisierung durch gute Tool-Unterstützung

Nachteile von White-Box-Tests gegenüber Black-Box-Tests
: Erfüllung der Spezifikation nicht überprüft
: Eventuell Testen "um Fehler herum"

Zudem sei genannt, dass die Unterscheidung zwischen Black-Box-Test und White-Box-Test teilweise von der Perspektive abhängt. Das Testen einer Teilkomponente ist aus Sicht des Gesamtsystems ein White-Box-Test, da für das Gesamtsystem aus der Außenperspektive keine Kenntnisse über den Systemaufbau und damit die vorhandenen Teilkomponenten vorliegen. Aus Sicht der Teilkomponente wiederum kann derselbe Test unter Umständen als Black-Box-Test betrachtet werden, wenn er ohne Kenntnisse über die Interna der Teilkomponente entwickelt und durchgeführt wird.

34.3 Testabdeckung

34.3.1 Maschinenbau

Je nach Art, Aufwand und Nutzen der Tests werden einige Tests stichprobenartig, andere Tests vollständig durchgeführt. Ein einfach und automatisch durchzuführender Test wird mit jedem Produkt durchgeführt, da seine Kosten die Produktionskosten nur geringfügig erhöhen. Ein Crashtest mit einem Fahrzeug wird jedoch natürlich nur mit Stichproben durchgeführt, da das getestete Produkt anschließend unbrauchbar wird.

Für 1000 produzierte Fahrzeuge könnte dies z. B. bedeuten, dass besonders aufwendige Tests und Crashtests nur mit einem einzigen Fahrzeug durchgeführt werden, während weniger aufwendige Tests mit einer größeren Zahl oder gar allen Fahrzeugen durchgeführt werden.

Notwendige, aber aufwendige Tests werden in ihrer Häufigkeit und damit der Testabdeckung variiert. Liefert ein Test überwiegend oder ausschließlich positive Ergebnisse, wird seine Zahl verringert. Liefert ein Test negative Ergebnisse, wird er häufiger eingesetzt, bis die Veränderungen an der Produktion zu einer deutlichen Steigerung positiver Ergebnisse und damit wieder einer höheren Produktqualität geführt hat.

Die Kosten-Nutzen-Rechnung solcher Tests wird mit Hilfe der Stochastik durchgeführt. Wird z. B. nur mit 5 von 1000 Fahrzeugen ein Test durchgeführt, ob die elektrischen Fensterheber einwandfrei funktionieren, lässt sich mit Hilfe der Stochastik die statistische Relevanz und die Wahrscheinlichkeit einer Fehleinschätzung aufgrund des Testergebnisses berechnen.

34.3.2 Softwaretechnik

Für die Testabdeckung in der `Softwaretechnik (engl. Test Coverage bzw. Code Coverage)` spielt die Stochastik praktisch keine Rolle, da es sich bei Computerprogrammen nicht um seriengefertigte Einzelprodukte handelt, bei denen Tests mit Stichproben durchgeführt werden. Stattdessen werden Tests anhand der Spezifikation (Eigenschaften der Schnittstelle) oder der inneren Struktur einer zu testenden Software-Einheit definiert.

In der Softwaretechnik wird die Testabdeckung für unterschiedliche Bereiche der Software ermittelt. Zu diesen gehört die Abdeckung des Codes, der Daten oder der Fachlichkeit. Um eine möglichst hohe Testabdeckung zu erreichen, müssen je nach abzudeckendem Bereich unterschiedliche Testfälle geschrieben werden. Nicht für alle kontrollflussorientierten Testverfahren ist beim Softwaretest die Angabe eines Maßes für die Testabdeckung möglich, da die Bestimmung der Anzahl der möglichen Testfälle für reale Probleme oft nicht möglich ist.

Eine vollständige Testabdeckung der Fachlichkeit stellt eine Ausnahme dar, weil die Anzahl möglicher Testfälle sehr schnell ungeheuer groß wird (durch kombinatorische Explosion). Ein vollständiger Funktionstest für eine einfache Funktion, die zwei 16-Bit-Werte als Argument erhält, würde schon 2^{16+16}, also ca. 4 Milliarden Testfälle bedeuten, um die Spezifikation vollständig zu testen. Stattdessen beschränkt man sich auf eine Auswahl sinnvoll erscheinender Tests für Grenzfälle. Beispiel: Eine Wurzelfunktion für rationale Zahlen könnte z. B. mit sämtlichen Elementen der Menge $-Double.MAX_VALUE; -1; -Double.MIN_VALUE; 0; Double.MIN_VALU$ getestet werden. Als sinnvolle Auswahl von Testfällen für eine angemessene Testabdeckung gelten in der Regel verschiedenartige gültige Argumente, für Komponenten mit Robustheitsanforderung zusätzlich Grenzelemente (gerade noch gültige Argumente und gerade ungültige Argumente). Es hat sich zudem als erfolgreich erwiesen, im Fehlerfall das den Fehler auslösende Argument in die Menge der Testelemente aufzunehmen.

Eine weitgehend vollständige Testabdeckung des Codes hingegen ist häufig das Ziel für Modultests bzw. Komponententests: Durch hochgradige Testabdeckung für 'kleine' Funktionseinheiten ergibt sich die Anzahl der insgesamt erforderlichen Testfälle lediglich aus der Addition dieser Testfälle und nicht aus der Kombinatorik der größeren Funktionalität. Doch auch hier kann dieses Ziel wegen verbleibender Restrisiken (unerwartete 'Fernwirkung' von Fehlern) und auch aus Kosten-Nutzen-Gründen in den meisten Fällen nicht zu 100 % erreicht werden.

Da eine hohe Codeabdeckung auch mit Tests erreicht werden kann, die nichts überprüfen, hat die Codeabdeckung für die Qualität der Tests eine nur eingeschränkte Aussagekraft. Um die Qualität von Tests sicherzustellen, wird daher üblicherweise mit Mutationstests gearbeitet. Dazu werden in den zu testenden Code automatisch Fehler bzw. Mutationen eingebaut und dann die Tests ausgeführt. Schlagen daraufhin Tests fehl, die davor funktioniert haben, so wurde die Mutation erkannt. Die Qualität der Tests kann aus dem Anteil der erkannten Mutationen abgeleitet werden. Die damit errechnete Testabdeckung bestimmt also die Abdeckung des Codes, der bei einer Mutation zu einem Fehlschlagen eines der Tests geführt hatte.

34.3.3 Testtiefe

Unter Testtiefe ist die Art und Anzahl der Prüfpunkte zu verstehen. Grob kann zwischen integrativen Blackbox-Tests, Greybox-Tests und Whitebox-Tests unterschieden werden, wobei Blackbox-Tests die geringste Anzahl an Prüfpunkten während der Testausführung beinhalten und Whitebox-Tests die höchste Anzahl.

34.3.4 Normen

Die Codeabdeckung wird auch von diversen internationalen Normen für Empfehlungen bzw. Mindestanforderungen herangezogen:

IEEE 1008 („Software Unit Testing"): 100 % Statement-Abdeckung als Anforderung für Fertigstellung der Modultests, 100 % Verzweigungs-Abdeckung für Module mit kritischem Code oder ungenügende Spezifikation.

ISO 26262 ("Road vehicles - Functional safety"): Je nach Kritikalität wird entsprechende Testabdeckung für Modultests und Integrationstests empfohlen bzw. stark empfohlen.

IEC 61508 ("Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems"): Empfiehlt bzw. empfiehlt stark eine 100 % Abdeckung von Eingängen, Statements, Verzweigungen, Bedingungen je nach Sicherheitsanforderungsstufe, beispielsweise wird bei der kleinsten Stufe eine 100 % Testabdeckung der Eingänge stark empfohlen, die der Statements, Verzweigungen und Bedingungen empfohlen.

DO-178B ("Software Considerations in Airborne Systems and Equipment Certification"): Verlangt eine 100 % Testabdeckung von Statements, Entscheidungen, Veränderten Entscheidungen je nach Auswirkung eines Systemfehlers. Beispielsweise wird ab Verletzungsgefahr von Passagieren eine 100 % Testabdeckung der Statements verlangt.

34.3.5 Messung der Codeabdeckung

Vielzahl an Produkten für die Sprachen Ada, C, C++, Cobol, C#, Java, Javascript, .NET, Perl, PHP, Python, Shellscript, Simulink, Tcl, VHDL, Verilog, Visual Basic uvam.

(aus https://de.wikipedia.org/wiki/Testabdeckung, 31Okt'22)

Teil IX
KI (AI)

KI

Da die Definition von "Intelligenz" eher *'nebulös'* ist, muss logischerweise auch *'künstliche Intelligenz'* verschwommen (also 'fuzzy') sein. . . :D

Künstliche **Intelligenz** (KI), auch artifizielle Intelligenz (AI bzw. A. I.), englisch artificial intelligence ist ein Teilgebiet der Informatik, das sich mit der Automatisierung intelligenten Verhaltens und dem maschinellen Lernen befasst. Der Begriff ist schwierig zu definieren, da es bereits an einer genauen Definition von "Intelligenz" mangelt. Dennoch wird er in Forschung und Entwicklung verwendet.

Meist bezeichnet künstliche Intelligenz den Versuch, bestimmte Entscheidungsstrukturen des Menschen nachzubilden, indem z. B. ein Computer so gebaut und programmiert wird, dass er relativ eigenständig Probleme bearbeiten kann. Oftmals wird damit aber auch eine nachgeahmte Intelligenz bezeichnet, wobei durch meist einfache Algorithmen ein "intelligentes Verhalten" simuliert werden soll, etwa bei Computergegnern in Computerspielen. [. . .]

Die Fähigkeit zu **lernen** ist eine Hauptanforderung an KI-Systeme und muss ein integraler Bestandteil sein, der nicht erst nachträglich hinzugefügt werden darf. [. . .]

Neben den Forschungsergebnissen der Kerninformatik selbst sind in die Erforschung der KI Ergebnisse der Psychologie, Neurologie und Neurowissenschaften, der Mathematik und Logik, Kommunikationswissenschaft, Philosophie und Linguistik eingeflossen. Umgekehrt nahm die Erforschung der KI auch ihrerseits Einfluss auf andere Gebiete, vor allem auf die Neurowissenschaften. Dies zeigt sich in der Ausbildung des Bereichs der Neuroinformatik, der der biologieorientierten Informatik zugeordnet ist, sowie der Computational Neuroscience.

Bei **künstlichen neuronalen Netzen (ANN)** handelt es sich um Techniken, die ab Mitte des 20. Jahrhunderts entwickelt wurden und auf der Neurophysiologie aufbauen.

KI stellt somit kein geschlossenes Forschungsgebiet dar. Vielmehr werden Techniken aus verschiedenen Disziplinen verwendet, ohne dass diese eine Verbindung miteinander haben müssen. [. . .]

Wissensbasierte Systeme modellieren eine Form rationaler Intelligenz für sogenannte Expertensysteme. Diese sind in der Lage, auf eine Frage des Anwenders auf Grundlage formalisierten Fachwissens und daraus gezogener logischer Schlüsse Antworten zu liefern. Beispielhafte Anwendungen finden sich in der Diagnose von Krankheiten oder der Suche und Beseitigung von Fehlern in technischen Systemen. [. . .]

Visuelle Intelligenz ermöglicht es, Bilder beziehungsweise Formen zu erkennen und zu analysieren. Als Anwendungsbeispiele seien hier Handschrifterkennung, Identifikation von Personen durch Gesichtserkennung, Abgleich der Fingerabdrücke oder der Iris, industrielle Qualitätskontrolle und Fertigungsautomation (letzteres in Kombination mit Erkenntnissen der Robotik) genannt.

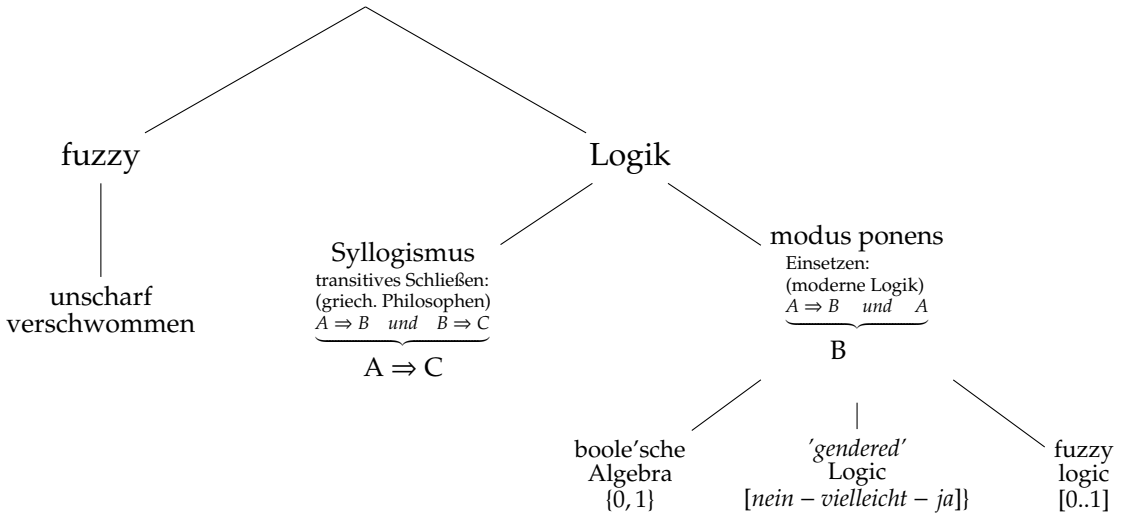
Mittels sprachlicher Intelligenz ist es beispielsweise möglich, einen geschriebenen Text in Sprache umzuwandeln (Sprachsynthese) und umgekehrt einen gesprochenen Text zu verschriftlichen (Spracherkennung). Diese automatische Sprachverarbeitung lässt sich ausbauen, so dass etwa durch latente semantische Analyse (kurz LSI) Wörtern und Texten Bedeutung beigemessen werden kann.

(aus https://de.wikipedia.org/wiki/Künstliche_Intelligenz, 17Feb'22)

Die Definitionen von *'Intelligenz'* und *'lernen'* in der Wikipedia sind 'wischi-waschi'.
"Intelligenz" ist das Gegenteil von "Dummheit"??
"Intelligenz" ist die Fähigkeit, "Intelligenz" zu definieren ??
"Intelligenz" ist Fehlervermeiden ??

(alls, was der XH sag, ischa Bledsinn! @17Feb22)

35 embedded Fuzzy Logic / Control



Mit *Logik] (von altgriechisch 'denkende Kunst', 'Vorgehensweise') oder* Folgerichtigkeit *wird Schlussfolgern, Schlussfolgerungslehre, Denklehre bezeichnet. In der* formalen *Logik wird die Gültigkeit von Argumenten untersucht (unabhängig vom Inhalt der Aussagen). Traditionell ist Logik Teil der Philosophie. Ursprünglich hat sich die traditionelle Logik in Nachbarschaft zur Rhetorik entwickelt. Seit dem 20. Jahrhundert versteht man unter Logik überwiegend symbolische Logik, die zB. innerhalb der Mathematik und der theoretischen Informatik als grundlegende Strukturwissenschaft behandelt wird.*
Die moderne symbolische Logik
➢ *verwendet eine künstliche Sprache*

➢ *Aussagenlogik ≡ Prädikatenlogik*

➢ `'Der Apfel ist rot'` $\xrightarrow{\text{Prädikatenlogik}}$ `'rot(Apfel)'`

➢ *streng definierte Schlussregeln*
zB. Aussagenlogik: atomare Aussagen → Buchstaben

➢ *symbolische Logik ≡ mathematische Logik ≡ formale Logik*

Die Syllogismen (von altgriechisch "Zusammenrechnen", "logischer Schluss") sind ein Katalog bestimmter Typen logischer Schlüsse. Sie bilden den Kern der im vierten Jahrhundert vor unserer Zeitrechnung entstandenen antiken Logik des Aristoteles und der traditionellen Logik bis ins 19. Jahrhundert. Der syllogistische Ansatz wurde erst durch die Integration der Logik in die Mathematik im Gefolge der Arbeiten von George Boole und Gottlob Frege im 19. und frühen 20. Jahrhundert als Haupttechnik der Logik abgelöst. Der Ausdruck "Logik" steht sowohl in der älteren Stoa wie im älteren Peripatos für eine Lehre vom Argumentieren bzw. Schließen, ist in dieser Bedeutung jedoch nicht vor dem 1. Jahrhundert v. Chr. belegt. Der Begriff wurde bereits von dem antiken Stoiker Zenon von Kition geprägt. Im Deutschen wird das Wort "Logik" im 19. Jahrhundert vielfach (etwa bei Immanuel Kant oder Georg Wilhelm Friedrich Hegel) auch im Sinne einer Erkenntnistheorie, Ontologie oder einer allgemeinen Dialektik verwendet. Die Logik im modernen Sinne wurde ua. als Analytik, Dialektik oder Logistik bezeichnet. Auch heute noch sind z. B. in der Soziologie Formulierungen wie Logik des Handelns oder der Literaturwissenschaft wie Logik der Dichtung u. Ä. verbreitet, bei denen unter "Logik" keine Theorie des Folgerns verstanden wird, sondern eine Lehre allgemeiner "Gesetze" oder Verfahrensweisen, die in einem bestimmten Bereich gelten. Insbesondere in der Tradition der Philosophie der normalen Sprache wurde unter einer logischen *Analyse vielfach eine Analyse begrifflicher Zusammenhänge verstanden. Unter dem Titel: "Logik der Forschung" (Karl Popper, 1935) sind alle oben genannten Verwendungsweisen des Wortes impliziert: die angemessenen methodischen Verfahrensweisen einer jeglichen Wissenschaft, welche wahrhaftige Erkenntnisse zur Folge haben sollen. Die einleitend dargestellte Verwendungsweise des Ausdrucks* Logik *ist dagegen seit Beginn des 20. Jahrhunderts üblich.*
(frei aus https://de.wikipedia.org/wiki/Logik, https://de.wikipedia.org/wiki/Syllogismus 05Feb'22)

s. `http://10.10.63.61` → "FuzzyCtl"

Fuzzylogic erlaubt Leuten ohne Kenntnis klassischer Regelkreismathematik (Laplace, Stabilitätskriterien, Einstellregeln, Hurwitz, Ziegler-Nichols, Chien-Hrones-Reswick, T-Summen-Regel, . . .) Steuerungen und Regelungen auf Basis primitiver Aussagen wie

'wenn kalt, dann heiz!'

zu erstellen.

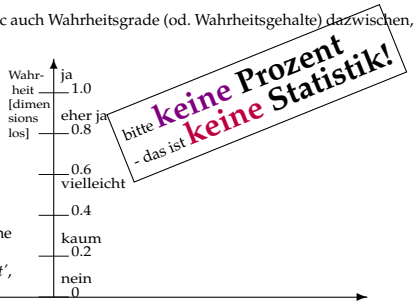
'Fuzzy Logic Controller' sind eine Anwendung der *'Frrrrtsch'* MAC-Operation und werden vorwiegend in Regelungs- und Steuerungsaufgaben (control technology) eingesetzt. Sie sind gut auch ohne Regelschleife (control loop) für *'forward predicton control'* einsetzbar, also vorteilhaft bei Stabilitäts- und Totzeitproblematiken.

35.1 Werte

Gegenüber der klassischen Aussagenlogik mit den Wahrheitswerten *'ja'* und *'nein'* verwendet Fuzzylogic auch Wahrheitsgrade (od. Wahrheitsgehalte) dazwischen, wie *'eher ja'*, *'kaum'*, *'ziemlich nein'* udgl., ausgedrückt in einer Zahl zwischen 0.0 (nein) und 1.0 (ja), zB.:

Wort (linguist.Term)	Zahl (Wahrheitsgrad)
nein	0.0
kaum	0.05
ziemlich nein	0.1
vielleicht	0.5
eher ja	0.67
ja	1.0

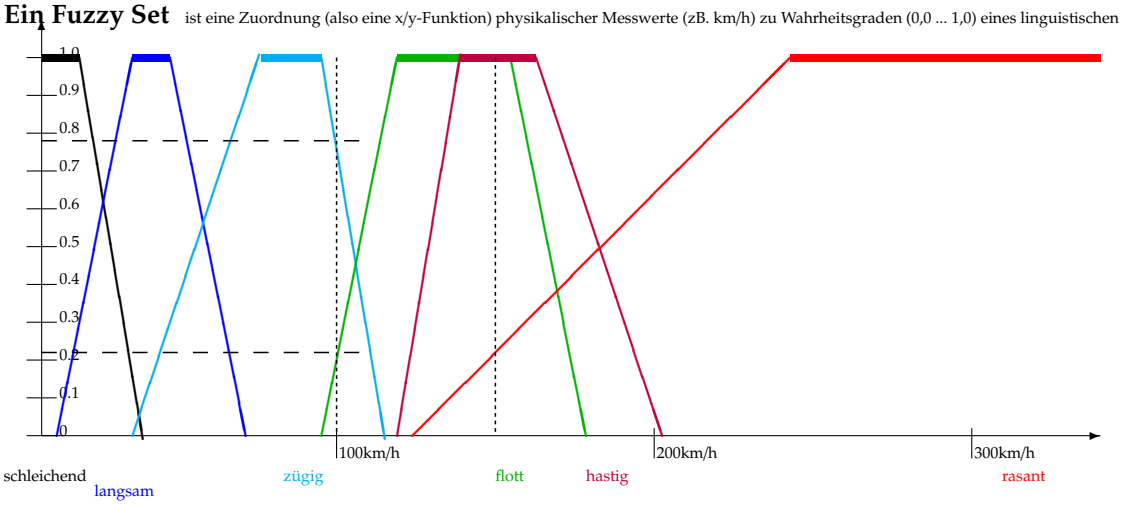
(willkürliche Zuordnung, nichtwissenschaftlich)



Eine linguistische Variable ist eine Größe, die *'linguistische Terme'* als Werte annimmt. So eine linguistische Variable wäre zB. *Geschwindigkeit*

linguistische Terme sind mögliche Werte einer linguistischen Variablen, zB. *'langsam'*, *'gemächlich'*, *'flott'*, *'rasch'*, *'hastig'*, *'schnell'*, *'rasant'* als Geschwindigkeitswerte.

Ein Fuzzy Set ist eine Zuordnung (also eine x/y-Funktion) physikalischer Messwerte (zB. km/h) zu Wahrheitsgraden (0,0 ... 1,0) eines linguistischen Terms



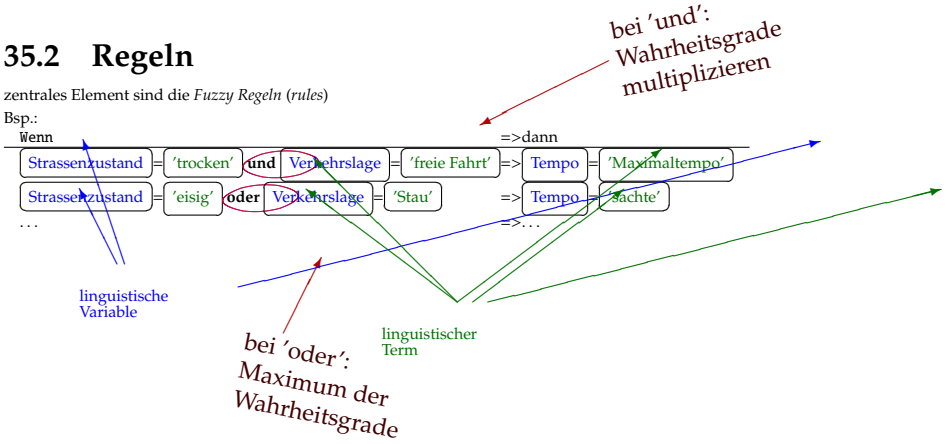
Hieraus ist zB. abzulesen, dass die Aussage 'die Geschwindigkeit ist ***flott'*** bei 100km/h nur einen Wahrheitsgrad von 0.21 besitzt, die Aussage 'die Geschwindigkeit ist ***zügig'*** hingegen 0.79 (also *'wahrer'* ist).

150 km/h bezeichnen wir zugleich als *'flott'*, *'hastig'* und *'rasant'* (mit unterschiedlichem Wahrheitsgehalt).

35.2 Regeln

zentrales Element sind die *Fuzzy Regeln* (*rules*)

Bsp.:



35.3 Defuzzyfication

Typischerweise sind mehrere Regeln zugleich mehr oder weniger zutreffend aufgrund der (and/or-kombinierten) Wahrheitsgrade der 'wenn' - Klauseln. Es ist dann nach folgenden Methoden aus den zugeordneten Ausgangs-Fuzzysets der 'dann' - Forderungen wieder ein Realwert zu gewinnen:

Vereinfachte Methode (nach Sugeno)

Bei zugleich mehrfachen 'dann' - Forderungen werden die Realwerte der linguistischen Terme mit den Erfüllungsgraden der Regeln gewichtet und die Produkte gemittelt (wie gewichtetes Mittel $\sum$ *Gewichte* = 1)

Eingangswert (X-Achse)

→ Erfüllungsgrade$\in$[0..1] aus Regel-Fuzzyset ablesen (Y-Achse)

→ Forderung

→ Gewicht

Konsistente Methode (nach Mamdani)

Die Ausgangs- (Forderungs-) Fuzzysets werden kombiniert (Begrenzung (Max-Min) oder Produkt (Max-Prod) mit Erfüllungsgraden) und die sich ergebende Menge wie ein Flächenschwerpunkt gemittelt

35.3.1 s. auch...

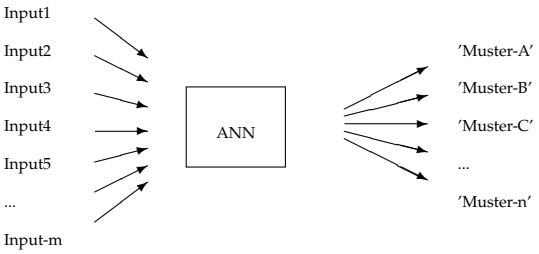
FuzzPrpe/FuzzPrpeXH08-Ie42b-9.pdf

36 embedded ANN

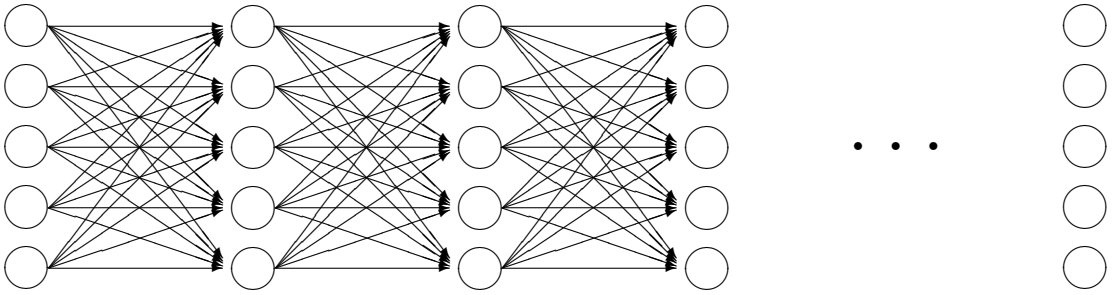
Was ist *Intelligenz*::=?? ist die kognitive bzw. geistige Leistungsfähigkeit speziell im Problemlösen; die Fähigkeit, sich in neuen Situationen zurechtzufinden und Aufgaben durch Denken zu lösen.
(aus https://de.wikipedia.org/wiki/Intelligenz)
KI ('künstliche Intelligenz, deut. 'artificial Intelligence' (AI))
(aus https://de.wikipedia.org/wiki/K%C3%BCnstliche_Intelligenz)
Die klassischen künstlichen neuronalen Netzwerke (ANN) ordnen jeden Input einer →"Aquivalenzklasse zu.
ANN *'artificial neural network'* verarbeiten m Inputs durch Musterklassifizierung unter Verwendung von k *'Neuronen'* zu n Outputs. Sie stellen damit — vor allem hinkünftig — ein fähiges (embedded) Steuerungssystem dar.

ANN simulieren Gehirnstrukturen als Vernetzung aus Verarbeitungseinheiten, die *'Neuronen'* genannt werden. Ein Neuron hat viele Inputs und *einen* Output, dessen Weiterleitung zu anderen Neuronen *'Synapsen'* heissen. Gewöhnlich werden sie *schichtenweise* vernetzt.

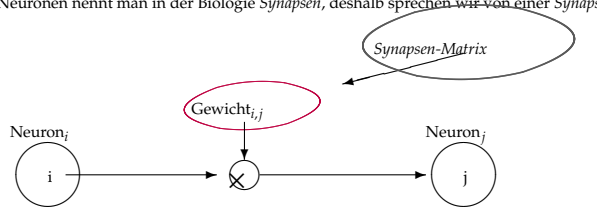
s. `http://10.10.63.61/Ebin2000/DOCs/ANN/neuro/ANN.pdf`
s. `http://10.10.63.61/SMUE/W02B/c/ann/ann03b.c`



ANN gruppieren die Inputkombinationsmöglichkeiten (m-Tupel) in Äquivalenzklassen; in der primitivsten Form wird jeder Äquivalenzklasse ein Output zugeordnet. Für anspruchsvollere (zB Steuerungs-) Aufgaben entspricht jeder Input-Äquivalenzklasse ein Output-Tupel.



Von jedem *'Neuron'*(=Knoten) - Ausgang zu jedem Eingang (auch zu sich selbst) gibt es eine Verbindungsstärke (=Multiplikationsfaktor, 'Gewicht'). Die Verbindung zwischen Neuronen nennt man in der Biologie *Synapsen*, deshalb sprechen wir von einer *Synapsen-Matrix der Verbindungsgewichte* (s.u.)



Jedes Neuron sumiert alle seine Inputs (=Output des Vorgängerneurons mal Verbindungsgewicht) und vergleicht die Summe mit einem Schwellwert th:

$$\sum Output_i * Gewicht_i > th$$

36.1 Synapsenmatrix

Das ANN ist (bes. in seiner Form als *'Synapsenmatrix'* - s.*'ann.c'*) eine Anwendung der MAC-Operation (*'Frrrrtsch'*).

Spalte-1 sind die Inputs, Zeile-1 die Outputs (oder umgedreht) und die Zellen sind die Synapsen (Weiterleitungs-Intensität). Bei k Neuronen, m Inputs und n Outputs entsteht eine (k+m, k+n) große Matrix (die überwiegend Nullen enthält ≡ *'dünn besetzte Matrix'*)

Output→ Input ↓	N1	N2	...	O1	O2	...
N1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,k+1}$	$w_{1,k+2}$	...
N2	$w_{1,1}$	$w_{1,2}$	$w_{2,3}$	$w_{2,k+1}$	$w_{2,k+2}$	...
...	$w_{1,1}$	$w_{1,2}$	$w_{3,3}$	...	...	...
I1	$w_{1,1}$	$w_{k+1,2}$	$w_{k+1,3}$	$w_{k+1,k+1}$	$w_{k+1,k+2}$	...
I2	$w_{1,1}$	$w_{k+2,2}$	$w_{k+2,3}$	$w_{k+2,k+1}$	$w_{k+2,k+2}$	...
...	...	...	...	...	...	...

I_m ... Netz-Inputs, N_k ... Neuronen, O_n ... Netz-Outputs

36.2 ANN-Synapsenmatrix-Demo

Listing 65: ANN Demo

```
/* 'carray01.c', 11Apr18, @XH, SdH,SdJ,SdL,SdM
 * ANN Synapsenmatrix Demo und Uebung
 * compile: 'gcc -o sm MatxAnn01.c -lm -pedantic'
 * run:     './sm'
```

```
 * stop     von selbst
 */

#include <stdio.h>      /*fuer printf-Ausgabe*/
#include "argcargv.h"


  /* FUNCTION PROTOTYPES: */
double MACop( double [], int, int, int );


int main( int argc, char *argv[] ){

  argcargv( argc, argv );       /* aus 'argcargv.h' */


  /* Synapsen-Matrix (SM): */
  int cols,input,muster;
  double SM [3] [9]={
/*input:*/      {  1,    0,    1,
                   0,    1,    0,
                   1,    0,    1  },

/*'X'*/         { .1,  -.3,   .1,
                 -.3,   .6,  -.3,
                  .1,  -.3,   .1  },

/*'O'*/         { .1,   .3,   .1,
                  .1, -1.4,   .1,
                  .1,   .3,   .1 }
        };

  printf("'X' erkannt: %+.2lf\t",
    MACop( (double*) SM, cols=9, input=0, muster=1 )
  );
  printf("'O' erkannt: %+.2lf\n",
    MACop( (double*) SM, cols=9, input=0, muster=2 )
  );

  return(0);
}

/*============================================================
 */double MACop( double M[], int cols, int sp1, int sp2){/*
      "MAC" = Multiply and ACumulate
            (multiplizieren + zammzaehlen)
============================================================*/
  int j;
  double sum=0.;
  for(j=0;j<cols;j++){
    sum += (M[sp1*cols+j] * M[sp2*cols+j]);
  }
  return(sum);
}
```

36.3 ANN-Synapsenmatrix-BspPGM

Listing 66: ANN Bsp

```
/* ann03.c
 * artificial neural network processor
 * with synapse matrix
 * HW03-5B
```

```
 * Stand 29Mar, 23Mar, 22Mar, 21Mar, 13-Mar-2008
 *
 * compile: gcc -o ann03 ann03.c
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <math.h>
#include <getopt.h>

typedef double wtyp;

/*Data Files: */
#define TRAINING_DATA_FILE      "anntdat.txt"
#define ANN_RESULT_FILE         "annresult.txt"
#define ANN_RUN_DATA_FILE       "anninput.txt"

/*ANNetz:*/
#define ANZ_SCHICHTEN           5
#define ANZ_NEURO_PRO_SCHICHT   20
#define ANZ_NEURONEN            (ANZ_SCHICHTEN*ANZ_NEURO_PRO_SCHICHT)
#define ANZ_NET_INPUTS          20
#define ANZ_NET_OUTPUTS         3
#define LERNRATE                0.125
#define ANZ_TRAINING_STEPS      10000

/*Neuronen-Transfer-Funktion: */
#define TRANSFER_XMIN    ((wtyp) -5.0 )
#define TRANSFER_XMAX    ((wtyp) +5.0 )
#define TRANSFER_YMIN    ((wtyp) -1.0 )
#define TRANSFER_YMAX    ((wtyp) +1.0 )
wtyp transfer_XMIN=TRANSFER_XMIN,
     transfer_XMAX=TRANSFER_XMAX,
     transfer_YMIN=TRANSFER_YMIN,
     transfer_YMAX=TRANSFER_YMAX;

/*Netz-Repraesentation als Matrix: */
#define WGT_ROW_DIM     (ANZ_NEURONEN + ANZ_NET_INPUTS)
#define WGT_COL_DIM     (ANZ_NEURONEN + ANZ_NET_OUTPUTS)
wtyp    net_input       [ ANZ_NET_INPUTS ];
wtyp    net_output      [ ANZ_NET_OUTPUTS ];
wtyp    weight          [ WGT_ROW_DIM ] [ WGT_COL_DIM ];        //==
   SynapsenMatrix
wtyp    neuron_o        [ WGT_COL_DIM ];
wtyp    neuron_i        [ WGT_ROW_DIM ];
wtyp    threshld        [ WGT_COL_DIM ];
wtyp    training_output [ ANZ_TRAINING_STEPS ] [ ANZ_NET_OUTPUTS ];
wtyp    training_input  [ ANZ_TRAINING_STEPS ] [ ANZ_NET_INPUTS ];
wtyp    wunsch_output   [ ANZ_NET_OUTPUTS ];


int  run_one_ann_propagation();
wtyp aktivierungs_funktion(wtyp inpt_summe, int neuron_nr);
wtyp transfer_funktion(wtyp inpt_summe);
int  ann_struktur_und_zufzi_init();
wtyp zufzi(wtyp, wtyp);
int  ann_training();
int  ann_LernSchritt(int lernschritt);
int  back_propagation();
int  ann_Production_Run();
int  read_training_data();
int  read_input_line(wtyp feld[], int anzahl, FILE*);
```

```
int  test_for_zeroes();
enum {msg_learn=1, msg_run=2, msg_dump=4};




/****ANN-einmal-laufen-lassen****
 *(aus 1 Input 1 Output brechnen)
 */
int run_one_ann_propagation(){
 int m, k, s, neuron_nr;
 wtyp summe, eingang, gewicht;

 for(s=0; s < ANZ_SCHICHTEN; s++){
  for(m=0; m < WGT_COL_DIM; m++){
    summe=0;
    //Summe aller Eingangswerte*Gewichte:
    for(k=0; k < WGT_ROW_DIM; k++){
      eingang= (k < ANZ_NEURONEN) ? neuron_i[k] : net_input[k];
      gewicht= weight[m][k];
      summe += ( eingang * gewicht );
    }
    //Schwellwert ueberschritten? => Ausgabe: +1
    if(m < ANZ_NEURONEN)
         neuron_o[m] = aktivierungs_funktion(summe, neuron_nr=m );
    else net_output[m]=neuron_o[m];
  }
  /* outputs wieder auf input-Spalte kopieren */
  for(m=0; m < ANZ_NEURONEN ; m++) neuron_i[m] = neuron_o[m];
 }
 return(0);
}









/****Neoronen-Transfer-Funktion ("sigmoid"?)****
 */
wtyp aktivierungs_funktion(wtyp inpt_summe, int neuron_nr){
/*
 * unsere etwas starke Vereinfachung:
 *
 *                   ^       _________________ .... YMAX
 *                   | <== /
 *                   |   / ==> (mit Schwellwert verschieben)
 *                   | /
 *-------------------/------------------------->
 *                 / |
 *              /    |
 *_____________/     |                          .... YMIN
 *
 *            |             |
 *          XMIN           XMAX
 */
 inpt_summe -= threshld[neuron_nr];
 return(
   (inpt_summe < transfer_XMIN)
   ?
   transfer_YMIN
```

```
    :
    ( (inpt_summe > transfer_XMAX)
      ?
      transfer_YMAX
      :
      (
        transfer_YMIN +
          (inpt_summe-transfer_XMIN)
        * (transfer_YMAX - transfer_YMIN) / (transfer_XMAX-transfer_XMIN)
      )
    )
 );
}




/****Neoronen-Transfer-Funktion ("sigmoid"?)****
 */
wtyp transfer_funktion(wtyp inpt_summe){
/*
 * unsere etwas starke Vereinfachung:
 *
 *                   ^       _________________ .... YMAX
 *                   | <== /
 *                   |   / ==> (mit Schwellwert verschieben)
 *                   | /
 *-------------------/------------------------->
 *                 / |
 *               /   |
 *_____________/     |                          .... YMIN
 *
 *            |             |
 *          XMIN          XMAX
 */
 return(
    (inpt_summe < transfer_XMIN)
    ?
    transfer_YMIN
    :
    ( (inpt_summe > transfer_XMAX)
      ?
      transfer_YMAX
      :
      (
        transfer_YMIN +
          (inpt_summe-transfer_XMIN)
        * (transfer_YMAX - transfer_YMIN) / (transfer_XMAX-transfer_XMIN)
      )
    )
 );
 /*schaut nit glei wia "C"="zaeh"
   aus, ha? ;-) */
}
```

```
/****ANN-initialisieren--(Zufall-ZufZi)****
 *   Struktur ergibt sich aus den Nullen
 *   und nicht-Nullen in der Verbindungs-Gewichts-Matrix
 */
int ann_struktur_und_zufzi_init(){
/**/
 #define VON (wtyp)
 #define BIS (wtyp)
 int k, n, anzs, rowvon, rowbis, colvon, colbis;
 printf("ANN init:\n");
 /*(1) alles auf null: */
 for(k=0;k < WGT_ROW_DIM; k++){
  for(n=0;n < WGT_COL_DIM; n++) weight[k][n]=0.0;
 }
 for(k=0;k < ANZ_TRAINING_STEPS; k++)
   for(n=0;n < ANZ_NET_OUTPUTS; n++) training_output [k] [n]=0.0;
 for(k=0;k < ANZ_TRAINING_STEPS; k++)
  for(n=0;n < ANZ_NET_INPUTS; n++) training_input [k] [n]=0.0;
 /*(2) inputs auf erste hidden-Schicht: */
 for(k=ANZ_NEURONEN; k< WGT_ROW_DIM; k++){
  for(n=0; n< ANZ_NEURO_PRO_SCHICHT; n++)
    weight[k][n] = zufzi( VON -0.8, BIS 0.8 );
 }
 /*(3) letzte hidden-Schicht auf outputs: */
 rowvon=ANZ_NEURONEN-ANZ_NEURO_PRO_SCHICHT;
 for(k=rowvon; k< ANZ_NEURONEN; k++){
  for(n=ANZ_NEURONEN; n< WGT_COL_DIM; n++)
    weight[k][n] = zufzi( VON transfer_YMIN, BIS transfer_YMAX );
 }
 /*(4) hidden-Schicht zu next hidden-Schicht: */
 for(anzs=0; anzs < ANZ_SCHICHTEN-1 ; anzs++){
  rowvon=anzs*ANZ_NEURO_PRO_SCHICHT;
  rowbis=(1+anzs)*ANZ_NEURO_PRO_SCHICHT;
  for(k=rowvon; k< rowbis; k++){
   colvon=k+ANZ_NEURO_PRO_SCHICHT;
   colbis=k+2*ANZ_NEURO_PRO_SCHICHT;
   for(n=colvon; n< colbis; n++)
     weight [k][n] = zufzi( VON -1.0, BIS 1.0 );
  }
 }
 /*(5) Neuronen-Schwellwerte: */
 for(k=0; k< WGT_COL_DIM ; k++){
   threshld[k] = zufzi(
       VON -0.4*(transfer_XMAX-transfer_XMIN),
       BIS  0.4*(transfer_XMAX-transfer_XMIN)
     );
 }
}


/****Zufallszahl ("ZufZi") erzeugen:
 */
wtyp zufzi(wtyp von, wtyp bis){
 static int zufzi_initialisiert=0;
 if(!zufzi_initialisiert) srand(997);
 return(
  von
  + (bis-von) * ( rand() / (wtyp)  RAND_MAX )
```

```
 );
}









/****ANN trainieren:****
 */
int ann_training(){
 int tschritt, zehntel, hundertstel;
 read_training_data();
 test_for_zeroes();
 printf("ANN trainieren:");
 zehntel=ANZ_TRAINING_STEPS/10; hundertstel=zehntel/10;
 for( tschritt=0 ; tschritt < ANZ_TRAINING_STEPS ; tschritt++ ){
  if(!(tschritt % zehntel)) printf("\n..... Schritt %6d   ",tschritt);
  if(!(tschritt % hundertstel)){ putchar('#'); fflush(stdout); }
  ann_LernSchritt(tschritt);
 }
 putchar('\n');
}


/****ANN-Lernschritt--(mit BackPropagation)****
 */
int ann_LernSchritt(int lernschritt){
 int i;
input_generieren:
 for(i=0 ; i< ANZ_NET_INPUTS ; i++) net_input[i]=training_input [lernschritt]
      [i];
netz_laufen_lassen:
 run_one_ann_propagation();
soll_output_hernehmen:
 for(i=0 ; i< ANZ_NET_OUTPUTS ; i++) wunsch_output[i]=training_output [
     lernschritt] [i];
ist_soll_vergleich_fuer_alle_outputs:
 {
  int k, neuron_nr, recursionLevel;
  wtyp delta;
  for( k=0; k < ANZ_NET_OUTPUTS ; k++){
   if(fabs(delta=wunsch_output[k] - net_output[k]) > 0.01){
    neuron_nr= ANZ_NEURONEN + k;
    back_propagation( neuron_nr, delta, recursionLevel=0 );
   }
  }
 }
}


/****back-propagation-step:
 //alle Eingangs-Gewichte anpassen
 //Schwellwerte anpassen
 */
int back_propagation( int neuron_nr, wtyp delta, int recursionLevel){
 int k,m;
 wtyp fehler;
//printf("41 "); fflush(stdout);
 if(recursionLevel > ANZ_SCHICHTEN) return(0);
 /* ganze Spalte nach Verursachern absuchen: */
 for( k=0; k< WGT_ROW_DIM; k++){
```

```
   if( fabs(weight[k][neuron_nr]) > 0.01 ){
    //korrigieren
    fehler= delta*weight[k][neuron_nr];
    if(k< ANZ_NEURONEN) back_propagation( k, fehler, recursionLevel+1 );
    weight[k][neuron_nr] += fehler * LERNRATE;
   }
  }
}









/****read one line from training-file into feld****
 */
int read_input_line( wtyp feld[], int anzahl, FILE *fp){
 /**/
 static char s1[ 16*(ANZ_NET_INPUTS+ANZ_NET_OUTPUTS) ],
      *s2;
 int k;
 fgets(s1, 16*anzahl, fp);
 s2=strtok(s1, " ");
 for(k=0; k< anzahl ; k++){
  feld[k]= (s2) ? atof(s2) : 0.0;
  s2=strtok(NULL, " ");
 }
}


/****read-training-input/output from file****
 */
int read_training_data(){
 FILE *fp;
 int  tschritt;

 if(!(fp=fopen(TRAINING_DATA_FILE,"r"))){
  perror(TRAINING_DATA_FILE);
 }else{
  for(tschritt=0 ; tschritt < ANZ_TRAINING_STEPS ; tschritt++ ){
   //input-Werte lesen
   read_input_line( training_input[tschritt], ANZ_NET_INPUTS, fp );
   //output-Werte lesen
   read_input_line( training_output[tschritt], ANZ_NET_OUTPUTS, fp );
  }
  fclose(fp);
 }
}



int save_ann_output(){
 FILE *fp;
 int  i;
 if(!(fp=fopen(ANN_RESULT_FILE,"a"))){
  perror(ANN_RUN_DATA_FILE);
 }else{
  fprintf(fp, "new_ANN_run: ");
  for(i=0; i< ANZ_NET_OUTPUTS ; i++){
   fprintf(fp, "%f ", net_output[i] );
```

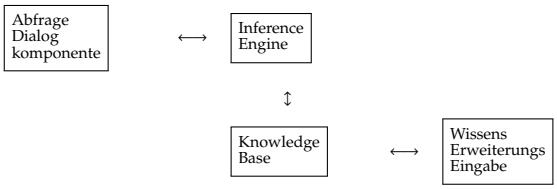
```
  }
  fprintf(fp, "\n");
  fclose(fp);
 }
}


int test_for_zeroes(){
 int k,m;
 long zeroctr=0L, lowctr=0;
 for(k=0; k< WGT_ROW_DIM; k++){
  for(m=0; m< WGT_COL_DIM; m++){
   if(weight[k][m] == 0.0) zeroctr++;
   if(fabs(weight[k][m]) < 0.01) lowctr++;
  }
 }
 printf("ANN Matrix has %ld elem, %ld zeroes, %ld lows/0.01\n",
   (0L+WGT_ROW_DIM)*WGT_COL_DIM, zeroctr, lowctr);
}









int ann_dump(){
}









int helptext(char **argv){
  printf("usage:\n");
  printf("%s --help        this helptext\n", argv[0]);
  printf("%s -t, --test    test transfer_funktion()\n", argv[0]);
  printf("%s -r, --run     run ANN\n", argv[0]);
  printf("%s --learndump   learn ANN, then dump\n", argv[0]);
  printf("%s --rundump     run ANN, then dump\n", argv[0]);
}




int main_parameter_auswerten(int argc, char **argv){
 int  r,opt_rv;
 wtyp q;
 static struct option long_options[] ={
  {"help",0,0,'h'},
  {"test",2,0,'t'},
  {"run",2,0,'r'},
  {"learndump",0,0,'l'},
  {"rundump",0,0,'d'},
  {0,0,0,0}
 };
 if(argc<2){ helptext(argv); _exit(0); }

 while( (r=getopt_long(argc, argv, "t::y::r",long_options,&opt_rv)) != -1){
  printf("rv=[%d/%c], opt_rv[%d], optarg=[%s], optopt=[%d], argv[optind]=[%s
      ]\n",
```

```
    r,r, opt_rv, optarg, optopt, argv[optind]);
  switch(r){
  case 'h': helptext(argv); break;
  case 't': for(q=-5.0; q<5.0; q+=0.5){
    printf("Transfer(%f)==%f\n", q, transfer_funktion(q));
   }
   break;
  case 'r': return(msg_learn+msg_run);
  case 'd': return(msg_learn+msg_run+msg_dump);
  case 'l': return(msg_learn+msg_dump);
  default: ;
  }
 }
 _exit(0);
}









int ann_Production_Run(){
 FILE *fp;
 int runctr=1;
 printf("ANN Production Phase:\n");
 if(!(fp=fopen(ANN_RUN_DATA_FILE,"r"))){
  perror(ANN_RUN_DATA_FILE);
 }else{
  while(!feof(fp)){
   printf("...ANN production turn Nr.%6d:\n", runctr++);
   /*******************************************************
   /*                                                     */
   /*/ read_input_line( net_input, ANZ_NET_INPUTS, fp ); /*
   /*/ run_one_ann_propagation();                         */
   /*/ save_ann_output();                                 */
   /*                                                     */
   /*******************************************************/
  }
  fclose(fp);
 }
}


/****main****main****main****main****main****main****main****
 *                          main                             *
 ****main****main****main****main****main****main****main****/

int main(int argc, char **argv){
 int rv=0;
 rv=main_parameter_auswerten(argc, argv);
 test_for_zeroes();
 if(rv & msg_learn){
ANN_Programm_HW03_5B:
 /*******************************************************
 /*                                                     */
 /*/ erstens:  ann_struktur_und_zufzi_init();          */
 */test_for_zeroes();/*
 /*/ zweitens: ann_training();                          */
 */test_for_zeroes();/*
 /*/ if(rv & msg_run)                                  */
 /*/ drittens: ann_Production_Run();                    */
 /*                                                     */
```

```
  /*/                                                    /*
  /*****************************************************/
  }
  if(rv & msg_dump) ann_dump();
}
```

37 embedded Expertensystem

37.1 Aufbau



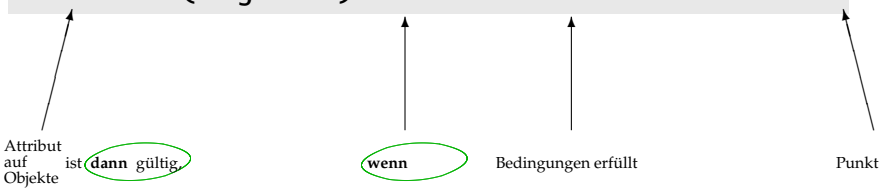
Expertensysteme sind meist keine *'Frrrrtsch'* MAC-Anwendungen, sondern von *'Symbolic Computation'*

37.2 regelbasierte Systeme

37.2.1 PROLOG

Regel::=

```
attribut(objekte)  :-  list-of-conditions.
```




```
list-of-conditions  ::=  condition ; list-of-conditions.
```

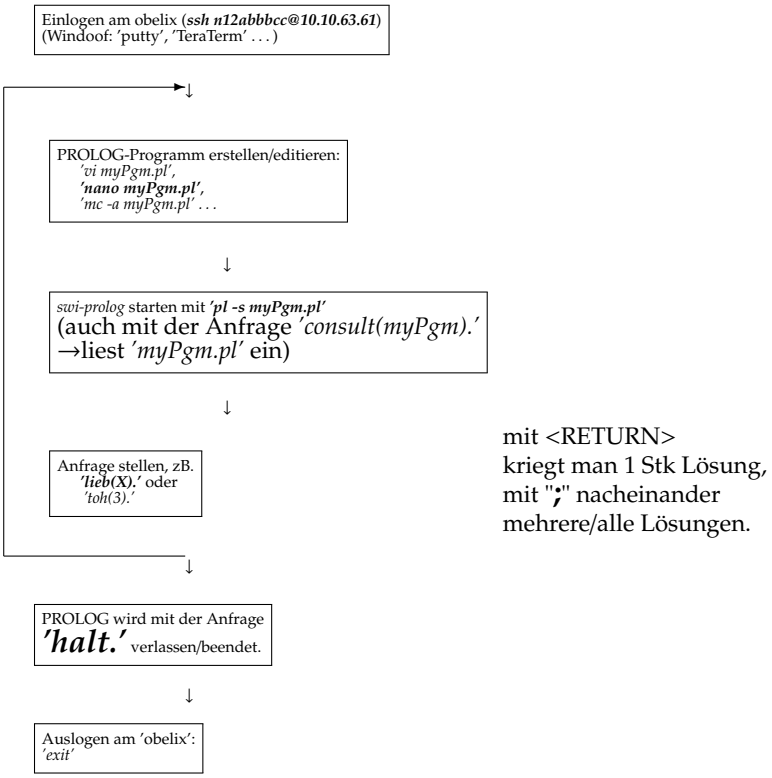
";"== ODER

das 'ODER' kann man auch als mehrere Regeln realisieren

```
list-of-conditions  ::=  condition , list-of-conditions.
```

","== UND

37.2.2 SWI-PROLOG am OBELIX



s. auch web-online-Interpreter (ohne Installation)
→ https://www.onlinegdb.com/online_prolog_compiler
→ https://www.tutorialspoint.com/execute_prolog_online.php

Sprache "PROLOG"	
Konstante Variable	kleingeschrieben zB. *joe* beginnt mit GrossBuchstabe oder "_" "_" (Unterstrich allein) ... "anonyme" Variable
Faktum	`attribut(konstante).` zB.: ist_mensch(klaus). ist_mensch(joe). ist_mensch(heli). ist_mensch(fratz). ist_viech(robin). ist_viech(indy). ist_viech(donald_duck). is_mine(robin). is_mine(indy). is_mine(fratz).
Fragen	"?-" + Faktum zB.: ?-ist_mensch(donald_duck). no ?-ist_viech(beethoven). no ?-ist_mensch(joe). yes
Fragen mit Variablen	?-ist_viech(X). X=robin; X=indy; X=donald_duck; no
Regeln	ist_wesen(X) :- ist_mensch(X) ; ist_viech(X). ";" ... "oder" inmyfamily(X) :- ist_wesen(X) , is_mine(X). "," ... "und"

37.2.3 Fakultät numerisch und symbolisch

```
% fuer 'swipl' interaktiv
% fac(X,N) XH@EaR
% facsym(X,N), XH/Fc5
%
fac(1,R) :- R < 2 .
fac(X,N) :- M is N-1, fac(Y,M), X is N*Y, !.

facsym(1,0).
facsym(X*M,M) :- N is M-1, facsym(X,N), !.
```

in der 'onlinegdb'-Webseite:

```
%https://www.onlinegdb.com/online_prolog_compiler
%numeric fac(n)
fac(1,R) :- R < 2 .
```

```
fac(X,N) :- M is N-1, fac(Y,M), X is N*Y, !.
facw(N):-fac(X,N),write('fac('),write(N),write(')='),write(X),halt.
:-facw(5).
```

```
%https://www.onlinegdb.com/online_prolog_compiler
%symbolic fac(n)
facsym(1,0).
facsym(X*M,M) :- N is M-1, facsym(X,N), !.
facs(N):-facsym(Q,N),write('fac('),write(N),
    write(')='),write(Q),halt.
:-facs(5).
```

in der 'tutorialspoint'-Webseite:

```
%https://www.tutorialspoint.com/execute_prolog_online.php
:- initialization(main).
main :- facw(6).
%numeric fac(n)
fac(1,R) :- R < 2 .
fac(X,N) :- M is N-1, fac(Y,M), X is N*Y, !.
facw(N):-fac(X,N),write('fac('),write(N),
    write(')='),write(X),halt.
```

```
%https://www.tutorialspoint.com/execute_prolog_online.php
:- initialization(main).
main :- facs(4).
%symbolic fac(n)
facsym(1,0).
facsym(X*M,M) :- N is M-1, facsym(X,N), !.
facs(N):-facsym(Q,N),write('fac('),write(N),
    write(')='),write(Q),halt.
```

37.2.4 Towers of Hanoi Spiel

```
%% (KI) Demo TOH - TowersOfHanoi
%% mit 'swi-pl' XH Xa6, XH EaR
%% interaktives Abfragen mit 'swipl': prima
toh(N) :- moveStack(N,t1,t2,t3),halt.

moveStack(0,_,_,_) :- !.
moveStack(N,A,B,C) :-
        M is N-1,
        moveStack(M,A,C,B), takeDisc(A,B), moveStack(M,C,B,A).

takeDisc(X,Y) :-
        write(X),
        write('->'),
        write(Y),
        nl.
```

in der 'onlinegdb'-Webseite:

```
%%fuer https://www.onlinegdb.com/online_prolog_compiler
%%interaktiv arbeiten: no nit gfunden (Xa6)
toh(N):-moveStack(N,t1 ,t2 ,t3),halt.
moveStack(0,_,_,_):-!.
moveStack(N,A,B,C):-
  M is N-1,
  moveStack(M,A,C,B),
  takeDisc(A,B),
  moveStack(M,C,B,A).
takeDisc(X,Y):-
  write(X),
  write('->'),
  write(Y),
  nl.
:- toh(5).
```

in der 'tutorialspoint'-Webseite:

```
%%fuer https://www.tutorialspoint.com/execute_prolog_online.php
%%interaktiv arbeiten: no nit gfunden (Xa6)
:- initialization(main).
main :- write('Hello to XHs World!\n'), toh(3).
toh(N):-moveStack(N,t1 ,t2 ,t3),halt.
moveStack(0,_,_,_):-!.
moveStack(N,A,B,C):-
  M is N-1,
  moveStack(M,A,C,B),
  takeDisc(A,B),
  moveStack(M,C,B,A).
takeDisc(X,Y):-
  write(X),
  write('->'),
  write(Y),
  nl.
```

37.2.5 symbolisch Differenzieren in 12 Zeilen!

```
%% (KI) Demo dx - symbolic differentiation
%%
%% XH, EaR
%%
d(X,X,1)        :- !.
d(C,X,0)        :- atomic(C).
d(-U,X,-A)      :- d(U,X,A).
d(U+V,X,A+B)    :- d(U,X,A), d(V,X,B).
d(U-V,X,A-B)    :- d(U,X,A), d(V,X,B).
d(C*U,X,C*A)    :- atomic(C),C \= X, d(U,X,A), !.
d(U*V,X,B*U+A*V):- write('Produktregel'),nl, d(U,X,A), d(V,X,B).
d(U/V,X,A)      :- write('Quotientenregel'),nl, d(U*V^(-1),X,A).
d(U^C,X,C*U^(C-1)*W) :- atomic(C), C \= X, d(U,X,W).
d(log(U),X,A*U^(-1)) :- d(U,X,A).
d(sin(U),X,A*cos(U)) :- d(U,X,A).
d(cos(U),X,(-A)*sin(U)) :- d(U,X,A).
```

in der 'onlinegdb'-Webseite:

```
%% fuer https://www.onlinegdb.com/online_prolog_compiler
%% interaktiv arbeiten: no nit gfunden (Xa6)
%% (KI) Demo dx - symbolic differentiation
%% XH-Xa6, XH-EaR
d(X,X,1):- !.
d(C,_,0):- atomic(C).
d(-U,X,-A):- d(U,X,A).
d(U+V,X,A+B):-d(U,X,A), d(V,X,B).
d(U-V,X,A-B):-d(U,X,A), d(V,X,B).
d(C*U,X,C*A):-atomic(C),C\=X, d(U,X,A), !.
d(U*V,X,B*U+A*V):-write('Produktregel'),nl, d(U,X,A), d(V,X,B).
d(U/V,X,A):-write('Quotientenregel'),nl, d(U*V^(-1),X,A).
d(U^C,X,C*U^(C-1)*W):- atomic(C), C \= X, d(U,X,W).
d(log(U),X,A*U^(-1)):- d(U,X,A).
d(sin(U),X,A*cos(U)):- d(U,X,A).
d(cos(U),X,(-A)*sin(U)):- d(U,X,A).

dw(N):-d(N,x,Q),write('d/dx('),write(N),
    write(')='),write(Q),halt.
:-dw(sin(x*x)/cos(1/x)).
```

```
Output:
Quotientenregel
Produktregel
Produktregel
Quotientenregel
d/dx(sin(x*x)/cos(1/x))=-1*cos(1/x)^(-1-1)*(-(1*(-1*x^(-1-1)*1))*sin(1/x))*sin(x*x)
        +(1*x+1*x)*cos(x*x)*cos(1/x)^-1

...Program finished with exit code 0
```

in der 'tutorialspoint'-Webseite:

```
%% fuer https://www.tutorialspoint.com/execute_prolog_online.php
%% interaktiv arbeiten: no nit gfunden (Xa6)
%% (KI) Demo dx - symbolic differentiation
%% XH-Xa6, XH-EaR
:- initialization(main).
main :- dw(sin(x*x)/cos(1/x)).
d(X,X,1):- !.
d(C,_,0):- atomic(C).
d(-U,X,-A):- d(U,X,A).
d(U+V,X,A+B):-d(U,X,A), d(V,X,B).
d(U-V,X,A-B):-d(U,X,A), d(V,X,B).
d(C*U,X,C*A):-atomic(C),C\=X, d(U,X,A), !.
d(U*V,X,B*U+A*V):-write('Produktregel'),nl, d(U,X,A), d(V,X,B).
d(U/V,X,A):-write('Quotientenregel'),nl, d(U*V^(-1),X,A).
d(U^C,X,C*U^(C-1)*W):- atomic(C), C \= X, d(U,X,W).
d(log(U),X,A*U^(-1)):- d(U,X,A).
d(sin(U),X,A*cos(U)):- d(U,X,A).
d(cos(U),X,(-A)*sin(U)):- d(U,X,A).

dw(N):-d(N,x,Q),write('d/dx('),write(N),
    write(')='),write(Q),halt.
```

```
Output:
Quotientenregel
Produktregel
Produktregel
Quotientenregel
d/dx(sin(x*x)/cos(1/x))=-1*cos(1/x)^(-1-1)*(- (1*(-1*x^(-1-1)*1))*sin(1/x))*sin(x*x)
        +(1*x+1*x)*cos(x*x)*cos(1/x)^-1
```

38 Grammatik, formale Sprache, Automatentheorie, embedded Control

Jede Art der Kommunikation ≡ *Sprache.*

Eine *Sprache* besteht aus der Menge all ihrer (korrekten) Sätze (ein Computerprogramm ist ein Satz).

↓

Auch jede Art der Mitteilung von Messwerten oder Steuerkommandos ist eine *Kommunikation.*

↓

Die Kommunikation zwischen Automaten ist meist stark formal schematisiert und fällt somit zentral in das Fachgebiet der *formalen Sprachen.*

Formale Sprachen und **Automaten (Schrittschaltwerke)** sind - je nach Klassifizierung - zudem **gleichmächtig** (Beweisführung: A. Touring) und folglich im Fokus digitaltechnischer Disziplinen.

38.0.1 formale Sprachen - Theorie

Eine **Sprache** besteht aus der **Menge** all ihrer korrekten Sätze

Eine **Grammatik** ist ein **Regelwerk** zur (verkürzten) Beschreibung der Sprache

betrifft:
- Syntax von Programmier- u. Kommandosprachen (zB. *'expressions'*)
- Struktur von Zeichenketten

Einteilung (Chomsky-Hierarchie):
- **reguläre** Sprachen
  Erkennung durch endliche Automaten bei *einmaligem* Lesen
- **kontextfreie** Sprachen
  die Bedeutung ist nicht vom Kontext abhängig
  Erkennung durch unendliche *Keller*-Automaten bei *einmaligem* Lesen
- **kontex-sensitive** Sprachen
  die Bedeutung hängt vom Kontext ab
  Erkennung durch linear beschränkte Automaten
- **unbeschränkte** Sprachen
  halten sich nicht an obige Schemata. (Fast alle "natürlichen" Sprachen)

- Grundelement: Der **Satz**[10]
- **Alfabet**: Menge von Sprachsymbolen (zB Buchstaben, Laute, 0/1-Bits, Morsezeichen)
- Beschreibung (aufgrund des Umfanges) meist in *'BNF'* oder *'EBNF'*[11] durch eine **Grammatik ::=** Menge von Regeln der Art

{ [Vn | Vt] }$^+$ ::= { [Vn | Vt] }$^+$

Vn ... Non-TerminalSymbol ("NT") (üblicherweise Grossbuchstaben)
Vt ... TerminalSymbol ("TS") (üblicherweise Kleinbuchstaben oder Sequenzen in Anführungszeichen)

zum Beispiel:

```
G(S)   =   {
S ::=  A;
A ::=  aB
A ::=  BBb
B ::=  b
B ::=  ab
}
```

hier:
Vt = { "a", "b", ";" }
Vn = { "S", "A", "B" }
wie lauten die gültigen Sätze dieser Sprache L ?
L (G(s)) = { "ab;", "aab;", "bbb;", "babb;", "abbb;", "ababb;" }

Begriffe:
- Phrase ::= Teilkette, die aus NT (NonTerminalSymbol) ableitbar ist
- einfache Phrase ::= Teilkette, die aus NT direkt ableitbar ist
- Ansatz := linkeste, einfache Phrase

zB:

```
a b a b b ;
---
 !
A ... Phrase, einfache Phrase, linkeste, also Ansatz
```

- Alfabet ::= { Vn ∪ Vt }
- Satzform ::= enthält auch NT (NonTerminalSymbole)
- Satz ::= enthält nur TS (TerminalSymbole)

Rekursivität:
- $G(A) = \{A ::= a|Ab\}$ ... *'linksrekursiv'*
  $L(G(A)) = ab^*$
- $G(A) = \{A ::= a|(A)\}$ ... *'zentralrekursiv'*
  $L(G(A)) = (^*a)^*$
- G(A) = { A ::= a | bA } ... *'rechtsrekursiv'*

Hierarchie der Phrasenstruktur-Grammatiken:
- $G := (V_N, V_T, R, S)$
  S ... Satz-Symbol
  R ... Regeln,
  V_T ... TerminalSymbole
  V_N ... NonTerminalSymbole
  V ... $V_N \cup V_T$
- $R := \{\alpha \rightarrow \beta : (\alpha = V^* V_N V^*)$ und $(\beta \epsilon V^*)\}$

[10] Verwechslungsgefahr: Manchmal auch *'Wort'* genannt
[11] BNF ... Backus-Naur-Form, EBNF ... Extendend BNF, https://de.wikipedia.org/wiki/Backus-Naur-Form

Klassifizierung:

- o $L_3 : A \to a, A \to aB, \ldots ...$ (beliebig viele Alternativen)
  reguläre Sprachen (gleichmächtig mit endlichen Automaten, Daten nur 1 x lesen)
- o $L_2 : A \to \beta$:
  kontextfreie Sprachen (Keller-Automaten (unendlich), Daten nur 1 x lesen))
- o $L_1 : \alpha \to \beta, |\alpha| \leq |\beta|$:
  context-sensitive Sprachen (linear beschränkte Automaten)
- o $L_0 : \alpha \to \beta$:
  unbeschränkte Sprachen (Turing-Maschine (Alan TURING, 1936))

was sind Automaten/StateMachines?

- o $A = (I, Z, O, f_z, f_o)$
  I...Input-Alfabet (Menge),
  O...Output-Alfabet (Menge),
  Z...innere Zustände (Menge),
  f_z ... Zustandsfunktion: $z_n := f_z(i, z_{n-1})$,
  f_o ... Outputfunktion: $o_n := f_o(i, z_n - 1)$

Übung:

- o (a) Welche Sprache L erzeugt die Grammatik
  $G := (S \to 0A, A \to 1|1A|0A)$?
- o (b) Welche Sprache L erzeugt die Grammatik
  $G := (S \to B, B \to b|bR, R \to b|z|bR|zR)$?
  $be\{Buchstaben\}, z\epsilon; \{Ziffern\}$

Quelle: Vorlesung "Formale Sprachen", Prof.Rechenberg, Linz

38.0.2 Bsp. 'arithmetic expression' syntax

```
ArExpr    ::=   "-" TERM | TERM "+" TERM | TERM "-" TERM | TERM .
TERM      ::=   OP "*" OP | OP "/" OP | OP .
OP        ::=   Konstante | Variable | "(" ArExpr ")" .
```

38.0.3 attributierte Grammatik f. Compilerbau

Arbeitsweise des Syntaxanalysators (Dr.W.Pree, 2002)

- Jede Ableitungsregel wird von links nach rechts durchlaufen
- Zur Alternativenauswahl wird das nächste Eingabesymbol verwendet
- Bei jedem Terminalsymbol fordert der Syntaxanalysator vom lexikalischen Analysator das nächste Eingabesymbol an
- Bei jedem Nonterminalsymbol
  - unterbricht der Syntaxanalysator die Abarbeitung der laufenden Regel
  - arbeitet das angetroffene Nonterminalsymbol ab
  - setzt dann an der Abbruchstelle fort.

Eine Attributgrammatik ist eine kontextfreie Grammatik, die um Attribute sowie Regeln und Bedingungen für diese Attribute erweitert ist. Angewandt wird das Konzept im Compilerbau, um die Einhaltung von Regeln zu überprüfen, die mit kontextfreien Grammatiken nicht formuliert werden können. Solche Regeln sind z. B. die, dass jede Variable deklariert sein muss und ihrem Datentyp entsprechend verwendet wird.

Ein Compiler überprüft die Einhaltung dieser Regeln während der semantischen Analyse. Dabei hat er nur die Informationen zur Verfügung, die im Syntaxbaum des Programms enthalten sind. Zusätzliche Informationen, die die semantische Analyse erleichtern, kann man als Attribute in den Syntaxbaum integrieren.

Zum Beispiel kann der Typ eines Ausdrucks als Attribut an den entsprechenden Knoten im Syntaxbaum annotiert werden. Durch Attributregeln und -bedingungen können zusätzlich Abhängigkeiten von anderen Attributen (auch anderer Knoten im Syntaxbaum) angegeben werden.

Die Programmierung der betreffenden Teile des Compilers vereinfacht sich, wenn die Produktionen der Grammatik selbst mit entsprechenden Attributen versehen werden.

S-Attributgrammatiken (SAG) sind Attributgrammatiken, die nur auf synthetischen Attributen arbeiten. So können sie direkt bei den Reduce-Schritten des Parse-Vorgangs eines LR(k)- Parsers berechnet werden. Implementiert in yacc.

L-Attributgrammatiken (LAG) können in einem Top-down-Durchgang von links nach rechts durch den abstrakten Syntaxbaum ausgewertet werden. Sie können für jede LL-Grammatik ausgewertet werden und somit für Pascal-ähnliche Programmiersprachen verwendet werden. Bei diesen dürfen nur abgeleitete und nachstehende Baumteile auf aktuelle Attribute zugreifen.

LR-Attributgrammatiken Eine Teilklasse der L-Attributgrammatiken, und zwar gerade diejenigen, die sich in einem Durchgang von links nach rechts während des LR-Parsens auswerten lassen. Implementierung: zyacc; in yacc von Hand über globale Variablen realisierbar. Der Vorteil der größeren Mächtigkeit des LR-Parsens gegenüber dem LL-Parsen manifestiert sich somit spiegelbildlich im Nachteil der geringeren Mächtigkeit der LR-Attributgrammatiken gegenüber den L-Attributgrammatiken.

ECLR-Attributgrammatiken Eine Variante der LR-Attributgrammatiken; sie benutzt eine Äquivalenzrelation, um die Attributauswertung zu optimieren.

(aus https://de.wikipedia.org/w/index.php?title=Attributgrammatik&oldid=116173443 01-Nov-16)

s. https://de.wikipedia.org/wiki/Kategorie:Compilerbau

38.0.4 *'Kleinbuchstaben'* schreibt man groß!

Paradoxerweise fordert die deutsche Rechtschreibung (www.duden.de), "Kleinbuchstaben" mit großem Anfangsbuchstaben zu schreiben:

`Klein-buch-sta-be`

Substantiv, maskulin - Buchstabe aus der Reihe der kleinen Buchstaben eines Alphabets
(01-Nov-16 aus http://www.duden.de/suchen/dudenonline/Kleinbuchstabe)

39 genetische Programmierung - evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) sind eine Klasse von stochastischen Optimierungsverfahren, die sich an biologischen Prozessen orientiert. In Anlehnung an die Natur werden Lösungskandidaten für ein bestimmtes Problem künstlich evolviert, EA sind also naturanaloge Optimierungsverfahren. Die Zuordnung zu den stochastischen und metaheuristischen Algorithmen bedeutet vor allem, dass EA meist nicht die beste Lösung für ein Problem finden, aber bei Erfolg eine hinreichend gute, was in der Praxis vor allem bei NP-vollständigen Problemen bereits wünschenswert ist. Die Verfahren verschiedener EA unterscheiden sich untereinander in erster Linie durch die genutzten Selektions-, Rekombinations- und Mutationsoperatoren, das Genotyp-Phänotyp-Mapping sowie die Problemrepräsentation. Die ersten praktischen Implementierungen evolutionärer Algorithmen wurden Ende der 1950er Jahre veröffentlicht, allerdings äußerten sich bereits in den vorhergehenden Jahrzehnten Wissenschaftler zum Potenzial der Evolution für maschinelles Lernen. Es gibt vier Hauptströmungen, deren Konzepte zumindest historisch voneinander zu unterscheiden sind:

- genetische Algorithmen
- Evolutionsstrategien
- genetische Programmierung und
- evolutionäre Programmierung

Heute verschwimmen diese Abgrenzungen zunehmend. Für eine bestimmte Anwendung wird ein EA geeignet entworfen, wobei in den letzten Jahrzehnten viele verschiedene Algorithmen und einzelne Operatoren entwickelt wurden, die heute benutzt werden können. Die Anwendungen von EA gehen über Optimierung und Suche hinaus und finden sich auch in Kunst, Modellierung und Simulation, insbesondere auch bei der Untersuchung evolutionsbiologischer Fragestellungen. Evolutionäre Algorithmen werden vorrangig zur Optimierung oder Suche eingesetzt. Konkrete Probleme, die mit EA gelöst werden, sind äußerst divers: z. B. Die Entwicklung von Sensornetzen, Aktienmarktanalyse, RNA-Strukturvorhersage, Schedulingprobleme, Designoptimierung (siehe auch obiges Bild der Satellitenantenne)

schaut verdammt nach Büroklammer aus, nit? Link s.u. (Anm.XH)

oder Roboterbahnplanung. Auch bei Problemen, über deren Beschaffenheit nur wenig Wissen vorliegt, können sie zufriedenstellende Lösungen finden. Dies ist auf die Eigenschaften ihres natürlichen Vorbildes zurückzuführen. In der biologischen Evolution sind die Gene von Organismen natürlich vorkommenden Mutationen ausgesetzt, wodurch genetische Variabilität entsteht. Mutationen können sich positiv, negativ oder gar nicht auf Erben auswirken. Da es zwischen erfolgreichen Individuen zur Fortpflanzung (Rekombination) kommt, können sich Arten über lange Zeiträume an einen vorliegenden Selektionsdruck anpassen (z. B. Klimaveränderungen oder die Erschließung einer ökologischen Nische). Diese vereinfachte Vorstellung wird in der Informatik idealisiert und künstlich im Computer nachgebildet. Dabei wird die Güte eines Lösungskandidaten explizit mit einer Fitnessfunktion berechnet, sodass verschiedene Kandidaten vergleichbar sind. Entsprechend dem natürlichen Vorbild gibt es bei den EAs Individuen, die aus einem Genom bestehen, welches die zu bestimmenden Eigenschaften der gesuchten Lösung in geeigneter Weise enthält. Ein Individuum entspricht einem Lösungskanidaten. Die durch die genetischen Operatoren erzeugten Individuen werden Nachkommen oder Kinder genannt. Eine Iteration des Verfahrens heißt entsprechend dem biologischen Vorbild Generation. Weitere Begriffsdefinitionen können in der VDI-Richtlinie VDI/VDE 3550 gefunden werden. In der Praxis könnte z.B. die Form einer Autotür so optimiert werden, dass der aerodynamische Widerstand minimal wird. Die Eigenschaften einer potenziellen Lösung werden dabei im Rechner als Genom gespeichert. Häufige Problemrepräsentationen sind Genome aus binären oder reellen Zahlen oder eine Reihenfolge bekannter Elemente (bei kombinatorischen Problemen, z. B. Travelling Salesman). Die starken Vereinfachungen, die im Vergleich zur Evolution getroffen werden, stellen ein Problem in Bezug auf die Erforschung evolutionsbiologischer Fragestellungen mit EA dar. Ergebnisse können nicht einfach auf die komplexere Natur übertragen werden. Das grobe Verfahren eines Evolutionären Algorithmus lässt sich wie folgt beschreiben:

- Initialisierung: Die erste Generation von Lösungskandidaten wird (meist zufällig) erzeugt.
- Evaluation: Jedem Lösungskandidaten der Generation wird entsprechend seiner Güte ein Wert der Fitnessfunktion zugewiesen.
- Durchlaufe die folgenden Schritte, bis ein Abbruchkriterium erfüllt ist:
- Selektion: Auswahl von Individuen meist basierend auf ihrer Fitness, die die Eltern für die Rekombination bilden
- Rekombination: Erzeugung von Nachkommen durch zufällige Kombination der Genome der Eltern
- Mutation: Zufällige Veränderung aller oder eines Teils der Nachkommen
- Evaluation: Jedem Nachkommen wird entsprechend seiner Güte ein Wert der Fitnessfunktion zugewiesen.
- Selektion: Bestimmung einer neuen Generation aus der alten und/oder den in dieser Generation gebildeten Nachkommen

Die gebräuchlichen EA unterscheiden sich in der Auswahl der Operatoren (Rekombination, Selektion, . . .) voneinander ab. So kann man beispielsweise die Schritte Rekombination - Mutation - Evaluation pro Elternpaar mehrmals ausführen, um pro Paarung mehr als zwei Nachkommen zu erzeugen. Oder es werden in Abweichung vom biologischen Vorbild Klone eines Elters erzeugt und mutiert. EAs unterscheiden sich außerdem in verschiedenen Problemrepräsentationen, der entsprechenden Fitnessfunktion oder zusätzlichen Schritten. Rekombination muss dabei nicht zwangsläufig stattfinden, da die Individuen sich auch asexuell fortpflanzen können. EA werden oft mit künstlichen neuronalen Netzen oder lokaler Suche kombiniert, was dann häufig zu memetischen Algorithmen führt. Je nach vorliegender Anwendung ergeben sich Vor- und Nachteile in Bezug auf spezielle Operatoren oder Konzepte. Evolutionäre Algorithmen unterscheiden sich untereinander vor allem in der jeweiligen genetischen Repräsentation, der Fitnessfunktion und den genutzten genetischen Operatoren: Mutation, Rekombination und Selektion. Mutation und Rekombination sind die Suchoperatoren evolutionärer Algorithmen, mit denen der Suchraum erkundet wird. Ihre Anwendung auf Lösungskandidaten kann keine Verbesserung garantieren, allerdings erhält der Suchprozess durch die Selektion eine Richtung, die bei erfolgreicher Konzeption zum globalen Optimum führt. Während mit dem Mutationsoperator völlig neue Bereiche des Suchraums erschlossen werden können, ermöglicht die Rekombination vor allem die Zusammenführung erfolgreicher Teillösungen oder Schemata bei den klassischen genetischen Algorithmen (Building-Block-Hypothese). Eine erfolgreiche Suche basiert also auf der Kombination beider Suchoperatoren. Der Erfolg eines Rekombinationsoperators hängt von der Beschaffenheit der Fitnesslandschaft ab. Je mehr lokale Optima die Fitnesslandschaft aufweist, desto wahrscheinlicher erzeugt die Rekombination aus zwei Individuen, die sich auf einem lokalen Optimum befinden, einen Nachfahren im Tal dazwischen. Mutationen sind von dieser Eigenschaft der Fitnesslandschaft nahezu unabhängig. Der Entwurf der verschiedenen Komponenten bestimmt, wie sich der evolutionäre Algorithmus bei der Optimierung des gegebenen Problems in Bezug auf Konvergenzverhalten, benötigte Rechenzeit und die Erschließung des Problemraums verhält. Insbesondere müssen die genetischen Operatoren sorgfältig auf die zugrunde liegende Repräsentation abgestimmt sein, sodass sowohl die bekannten, guten Regionen des Problemraums genutzt, als auch die unbekannten Regionen erkundet werden können. Dabei spielen die Beziehungen zwischen Such- und Problemraum eine Rolle. Im einfachsten Fall entspricht der Suchraum dem Problemraum (direkte Problemrepräsentation).

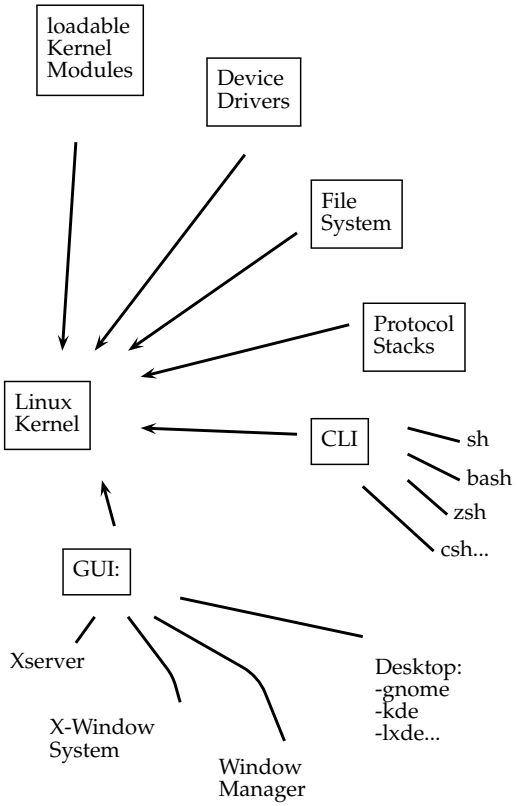
(aus https://de.wikipedia.org/wiki/Evolution%C3%A4rer_Algorithmus, 01Sep'22)

40 OS4ES: Betriebssysteme f. Embedded Systems

CLI ..... CommandLineInterpreter
ES ..... EmbeddedSystem
GUI ..... GraphicalUserInterface
MMU ..... MemoryManagementUnit
OS ..... OperatingSystem
RT ..... RealTime

UI ..... UserInterface
VFS ..... VirtualFileSystem
VMM ..... VirtualMemoryManagement
= MMU+Auslagerung(swapping)

40.1 BlockDG



Legende:
CLI command line interpreter, Kommandozeilen-Eingabe, 'cmd'
GUI graphical user interface, grafische Bedienungsoberfläche, das '*Desktop*'

40.2 security issues

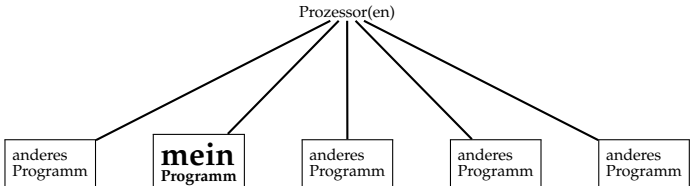
Sicherheitskritisch ist der Konflikt aus

a) Kernelmodule brauchen den *supervisor mode* für Hardwarezugriffe
b) aus Sicherheitsgründen sollten auch Kernelmodule nur im *user mode* laufen (*vergiftete* Module könnten zu Admin-Rechten kommen)

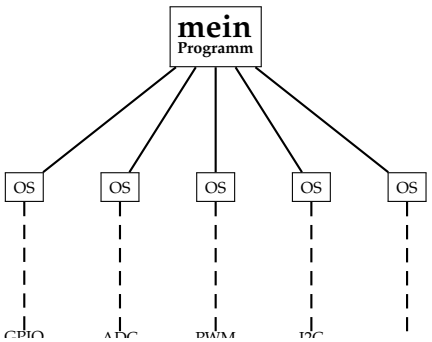
40.3 ComputerModul als EmbeddedSystem: Application auf OS

Problem:

- Mein Steuerungsprogramm muss sich die *Ressourcen* — besonders den/die Prozessoren — mit anderen Programmen teilen, dh. es wird immer wieder unterbrochen



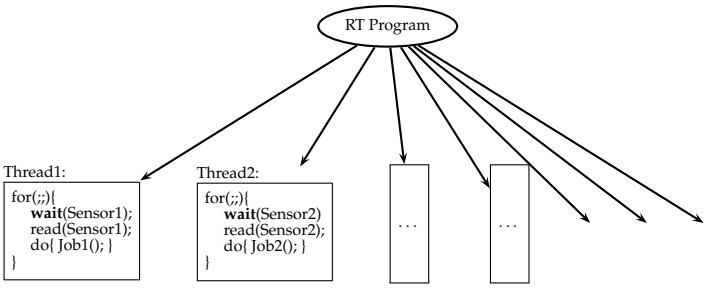
- Mein Steuerungsprogramm hat keinen Zugriff auf die Hardware (nix 'digitalWrite()' und vor allem: **nix 'Interrupt'**) und muss dazu OS-Funktionen aufrufen.



⟶
Mein Steuerungsprogramm ist ganz anders zu strukturieren:

- jede Einzelaufgabe wird als **Parallelprozess (Thread)** programmiert.
- die Parallelthreads müssen Wartezustände einnehmen (mit blocking read(), nanosleep(), set_timer, select() odgl.)
- *Wichtigkeit* wird mit Prioritätenvergabe geregelt.
- I/O wird als read/write auf 'device files' realisiert.
- der Datenaustausch zwischen **Parallel**prozessen wird mit **Prozesskommunikation**smitteln realisiert (→pipe, →message queue, →shared memory).

dh. die Reihenfolgesteuerung der Einzelaufgaben wird dem
Taskmanagement ('Scheduling')
des OS überlassen.



oder

var1, var2, . . .

Thread1:
```
Control(
var1,var2,...){
. . .
}
```

Thread2:
```
for(;;){
    wait(Sensor1)
    var1=read(Sensor1)
}
```

Thread3:
```
for(;;){
    wait(Sensor2)
    var2=read(Sensor2)
}
```

40.3.1 Warte-Zustand

Wartezustände erreichen wir zB. mittels

- read() im *'blocking mode'*
- nanosleep(), clock_nanosleep(), usleep()
- per timer_settime() ausgelöste *pthreads*
- poll(), epoll(), select() (epoll() nicht POSIXkonform)
- sem_wait(), mq_receive()
- wait(), waitid(), waitpid()

Achtung:
Wenn ein *'file descriptor'* (per *'fcntl'* oder *'creat'*) in den *'non-blocking-mode* versetzt wurde, wartet *'read()'* **nicht mehr**!

40.3.2 realtime-Übung

Hardware out

```
fd=open( devicefile, O_WRONLY );
n=write( fd, value, length);
close(fd);
```

Hardware in

```
fd=open( devicefile, O_RDONLY );
n=read( fd, &value, length);
close(fd);
```

parallel thread

```
int lampon=0;
struct timespec w={0L, 500000000L};
  ...
pthread_create( &hdl, 0, myFunction, 0);
  ...
```

```
void *myfunction(void* a){
  for(;;){
   fd=open("/dev/led0", O_RDONLY );
   n=read(fd, &lampOn, 1);
   close(fd);
   lampOn = !lampOn;
   fd=open("/dev/led0", O_WRONLY);
   n=write(fd, lampOn, 1);
   close(fd);
   nanosleep(w,NULL));
  }
}
```

nanosleep

```
struct timespec w={0L, 500000000L};
  ...
nanosleep(w,NULL);
```

OS timer

```
struct timespec tstrt={5L,0L};
struct timespec tres={5L,0L};
timer_t tmrID;
const struct itimerspec
    tmrStopSpec={{0L,0L},{0L,0L}};
const struct itimerspec
    tmrSpec={{0L,500000000L},{0L,500000000L}};
  ...
struct sigevent sev;
sev.sigev_notify_function = timerServiceRoutine;
sev.sigev_notify = SIGEV_THREAD;
sev.sigev_value = (sigval_t)0;
sev.sigev_notify_attributes = NULL;

if(0>timer_create(CLOCK_REALTIME, &sev, &tmrID))
    perror("timer_create failed.");

if(0>timer_settime(tmrID, 0, &tmrSpec, NULL))
    perror("timer_settime failed.");
  ...
```

```
void timerServiceRoutine(union sigval v){
  fd=open("/dev/led0", O_RDONLY );
  n=read(fd, &lampOn, 1);
  close(fd);
  lampOn = !lampOn;
  fd=open("/dev/led0", O_WRONLY);
  n=write(fd, lampOn, 1);
  close(fd);
}
```

40.3.3 OS Timing

40.4 delaying + time wasting in general

40.4.1 busy waiting

Tue **niemals**(!) die Zeit mit *Warteschleifen*

```
for( i=0; i<N; i++)
```

verbringen (= *'busy waiting'*)!

Busy waiting beschäftigt die Prozessoren, **belastet** also das Computersystem und

→ das ist **no go**

Wenn man **warten muss**, möchte man die Zeit für etwas anderes **nutzen**. Auch im *embedded real time* Computermodul möchte man

a) entweder Tasks im *ready state* weiterrechnen
b) oder dem OS die Möglichkeit zur *Spar-Abschaltung* bieten

40.4.2 sleep() :no

ist die wohl bekannteste Unix/C Zeitschinderfunktion (→ *man 1 sleep, man 3 sleep*)(unistd.h), kann aber

- nur in Vielfachen ganzer Sekunden warten
- ist sehr ungenau → wartet mitunter wesentlich länger

40.4.3 usleep() / unistd.h :no

The usleep() function suspends execution of the calling thread for (at least) usec microseconds. The sleep may be lengthened slightly by any system activity or by the time spent processing the call or by the granularity of system timers.

POSIX.1-2001 declares this function obsolete; use nanosleep(2) instead. POSIX.1-2008 ***removes the specification of usleep()****.* (aus der *man page*)

40.4.4 nanosleep() / time.h :✓

```
#include <time.h>
int nanosleep(const struct timespec *req,
     struct timespec *rem);
```

*nanosleep() suspends the execution of the calling thread until either at least the time specified in *req has elapsed, or the delivery of a signal that triggers the invocation of a handler in the calling thread or that terminates the process.*

*If the call is interrupted by a signal handler, nanosleep() returns -1, sets errno to EINTR, and writes the remaining time into the structure pointed to by rem unless rem is NULL. The value of *rem can then be used to call nanosleep() again and complete the specified pause (but see NOTES).*

The structure timespec is used to specify intervals of time with nanosecond precision. It is defined as follows:

```
struct timespec {
    time_t tv_sec;   /*seconds*/
    long   tv_nsec;  /*nanoseconds*/
};
```

The value of the nanoseconds field must be in the range 0 to 999999999.

40.4.5 clock_nanosleep() / time.h :✓

Noch *realtimiger* schläft man mit:

clock_nanosleep - high-resolution sleep with specifiable clock

```
#include <time.h>
int clock_nanosleep(
        clockid_t clock_id,
        int flags,
        const struct timespec *request,
        struct timespec *remain);
```

Link with -lrt (only for glibc versions before 2.17).

Like nanosleep(2), clock_nanosleep() allows the calling thread to sleep for an interval specified with nanosecond precision. It differs in allowing the caller to select the clock against which the sleep interval is to be measured, and in allowing the sleep interval to be specified as either an absolute or a relative value.

The time values passed to and returned by this call are specified using timespec structures, defined as follows:

```
struct timespec {
    time_t tv_sec;   /*seconds*/
    long   tv_nsec;  /*nanoseconds*/
};
```

The clock_id argument specifies the clock against which the sleep interval is to be measured. This argument can have one of the following values:

CLOCK_REALTIME A settable system-wide real-time clock.

CLOCK_MONOTONIC A nonsettable, monotonically increasing clock that measures time since some unspecified point in the past that does not change after system startup.

CLOCK_PROCESS_CPUTIME_ID A settable per-process clock that measures CPU time consumed by all threads in the process. (aus der *man page*)

40.5 Betriebssystem - Timer; POSIX Standard konform:

40.5.1 mit 'timer_create()'

Listing 67: timer create Nr1

```
/* tcreatX01.c : experiment w. timer_create(), cfs@PbIj2
 *
 * +--starts 'timerID' as a periodic cylce SYSTEM TIMER
 *    +--which calls 'timerServiceRoutine' as a parallel thread each time
 *       +--which does timing statistics
 * =one task splitting up into sub-thread(s)
 *
 * advice: read 'man 7 time'
 * compile : gcc tcreatX01.c -o tcX01 -lrt -lpthread
 * run     : ./tcX01
 * stop    : (self terminating)
 */
#include <stdio.h>      //printf
#include <stdlib.h>
#include <string.h>     //memcpy
#include <stdint.h>     //uint64-t
#include <time.h>       //struct timespec
#include <signal.h>     //sigval_t
#include <limits.h>     //LONG_MAX
#define DebugModeXH 0

    //ZeitSpeicher:
```

```
#define ExpDim 200
struct timespec tMem[ExpDim]={{0L,0L}};
int             expCtr=1;
struct timespec t;
long long       nsecsDiff=0LL;
    //timer:
timer_t         timerID;
const struct itimerspec tmrStopSpec={{0L,0L},{0L,0L}};
```

```
void timerServiceRoutine(union sigval v){/*
-------------------------------------------------------------*/
  clock_gettime(CLOCK_REALTIME, &t);    //GET_EXACT_TIME
  #if(DebugModeXH)
    nsecsDiff= (t.tv_sec-tMem[expCtr-1].tv_sec)*1000000000LL + (t.tv_nsec-tMem[expCtr
        -1].tv_nsec);
    printf("timer: Nr=%5d t=%4ld.%06lds, dt=%10lldns\n", expCtr, t.tv_sec, t.tv_nsec
        /1000L, nsecsDiff);
  #endif  //DebugMode
  memcpy( &(tMem[expCtr++]), &t, sizeof(t));
  if(expCtr >= ExpDim) timer_settime(timerID, 0, &tmrStopSpec, NULL);
}
```

```
#define DO_INITIAL_KREMPEL(a)   {\
  struct timespec tstrt={5L,0L};        \
  struct timespec tres={5L,0L};         \
  if(0 > clock_gettime(CLOCK_REALTIME, &tstrt)) perror("clock_gettime failed.");
      \
  if(0 > clock_getres(CLOCK_REALTIME, &tres)) perror("clock_getres failed.");   \
  if(a) printf("timer_create ist POSIX.1-2001\n"        \
        "start @t=[%ld.%09lds],\ttimerInterval=%ld\n"   \
        "CLOCK_REALTIME Granularity=%lds+%ldns,\tCLOCKS_PER_SEC =(%ld)\n",      \
        tstrt.tv_sec, tstrt.tv_nsec, tINTVL, tres.tv_sec, tres.tv_nsec, CLOCKS_PER_SEC);
                  \
  }


#define START_INTERVAL_TIMER()  {\
  const struct itimerspec tmrSpec={{0L,tINTVL},{0L,tINTVL}};    \
  struct sigevent sev;          \
  sev.sigev_notify_function     = timerServiceRoutine;  \
  sev.sigev_notify              = SIGEV_THREAD; \
  sev.sigev_value               = (sigval_t)0;  \
  sev.sigev_notify_attributes   = NULL; \
  if(0> timer_create(CLOCK_REALTIME, &sev, &timerID)) perror("timer_create failed.");
      \
  if(0> timer_settime(timerID, 0, &tmrSpec, NULL)) perror("timer_settime failed.");
      \
  }


#define WAIT_some_Time()        {\
  struct timespec wm={5L,0L};   \
  nanosleep( &wm, NULL);        \
  }

#define KILL_Timer()    { timer_delete(timerID);}



/*
----------------------------------------*/
int main(int argc, char *argv[]){/*
----------------------------------------*/

  #define tINTVL 20000000L

  DO_INITIAL_KREMPEL(0);
  START_INTERVAL_TIMER();
  WAIT_some_Time();
  KILL_Timer();


    //ABSCHLUSS_AUSWERTUNG:
  long long nsecsDiffMax=0LL, nsecsDiffMin=LONG_MAX;
  int i;
  for(i=2;i<expCtr;i++){
    nsecsDiff   = (tMem[i].tv_sec  - tMem[i-1].tv_sec )*1000000000L
                + (tMem[i].tv_nsec - tMem[i-1].tv_nsec);
    if( nsecsDiff > nsecsDiffMax ) nsecsDiffMax=nsecsDiff;
    if( nsecsDiff < nsecsDiffMin ) nsecsDiffMin=nsecsDiff;
  }
  printf("Min-Max: %lld..%lld -> b=(%lld)==(%lf\%)\n",
        nsecsDiffMin, nsecsDiffMax, nsecsDiffMax-nsecsDiffMin, (1.0+nsecsDiffMax-
            nsecsDiffMin)/(1.0+tINTVL) );
}
```

40.5.2 mit 'timerfd_create()', poll() und read()

Ein *timer*-File

```
timerFD=timerfd_create(
        CLOCK_REALTIME,         //mode
        0);
```

wird im Timertakt

```
const struct itimerspec newTS=
        {{0L,312345678L},{2L,10L}};
        //{{interval},{starttime}}
timerfd_settime(
        timerFD,        //file handle
```

```
        0,
        &newTS,         //timing
        NULL);
```

geschrieben (vom OS).
Ein *parallel thread*

```
rv=pthread_create(
        &thTimer,       //thread-handle
        NULL,
        &timerReact,    //ISR
        NULL);
```

wartet mit

```
poll( pollfds, 1, -1);
read( timerFD, tmrbuf, sizeof(uint64_t));
```

jeweils auf diese *Befüllungen*. Die Daten sind unbedeutend, denn wir sind nur auf die Zeit--Taktung scharf.

Listing 68: mit timerfd_create()

```
/* timerfdX01.c example cfs@PaD96
 *   DEMO: "timerfd_create()", "poll()"
 * ---
 * there are
 *   no "timerISR" interrupt service routines available
 *   under multi-user OSes,
 * plus,
 *   users wouldnt have (direct) access to any computer hardware
 *   (but through OS API function calls).
 * ---
 * This DEMO does:
 *   OS timer signals timer expirations by writing to a file "timerFile"
 *   a call to poll() is blocked, until the timer writes into it
 *   "timerReact"-thread repeatedly waits by poll() for timer expirations
 *     and displays elapsed time values
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>     //memcpy
#include <stdint.h>     //uint64-t
#include <poll.h>
//#include <time.h>
#include <sys/timerfd.h>
#define  LOOP_FOREVER /**/

struct timespec tstrt, tres;
int             timerFD;
pthread_t       thTimer;

int timerReact(){  //used_like_an_ISR
  struct timespec t,oldt;
  struct pollfd   pollfds[1]={{-1,POLLIN,0}};
  char    tmrbuf[sizeof(uint64_t)];
  pollfds[0].fd=timerFD;
  clock_gettime(CLOCK_REALTIME, &oldt);
  for(;LOOP_FOREVER;){
    //WAIT_FOR_CHANGES_ON_TIMER_FILE_:
      poll( pollfds, 1, -1);
      read( timerFD, tmrbuf, sizeof(uint64_t));
    //GET_EXACT_TIME_:
      clock_gettime(CLOCK_REALTIME, &t);
    //DISPLAY_REPORT_INFO_:
      printf("tReact @t=%4ld.%03lds d=%8ldus\n", t.tv_sec-tstrt.tv_sec, t.tv_nsec/1000000L,
           (t.tv_sec-oldt.tv_sec)*1000000L + (t.tv_nsec-oldt.tv_nsec)/1000L );
    //STORE_TIME_STAMP_FOR_LATER_USE_:
      memcpy(&oldt,&t,sizeof(t));
  }
}

/*
----------------------------------------*/
int main(int argc, char *argv[]){      /*
----------------------------------------*/
  const struct itimerspec newTS={{0L,312345678L},{2L,10L}};  //{{interval},{starttime}}
  int   rv;

  //DO_INITIAL_STUFF_:
    clock_gettime(CLOCK_REALTIME, &tstrt);
    clock_getres(CLOCK_REALTIME, &tres);
    printf("start @t=[%ld.%09ld], CLOCK_REALTIME Granularity=%ldns\n"
```

```
        "CLOCKS_PER_SEC=%ld.\n",
        tstrt.tv_sec,tstrt.tv_nsec,  tres.tv_nsec, CLOCKS_PER_SEC);

  //START_TIMER_INTERVAL_:
    timerFD=timerfd_create(CLOCK_REALTIME,0);   //timer_erzeugen
    timerfd_settime(timerFD, 0, &newTS, NULL);  //timer_starten

  //START_TIMER_REACTION_FUNCTION_THREADS_:
    rv=pthread_create( &thTimer, NULL, &timerReact, NULL);

  //n_WAIT_FOR_END_OF_PARALLEL_THREAD___WHICH_NEVER_ENDS_:
    pthread_join( thTimer, NULL);

  close(timerFD);
}
```

40.5.3 mit 'timerfd_create()', poll() und read() *extended*

Wie *'timerfdX01.c'* example s.o.

Listing 69: timerfdX02.c example

```
/* timerfdX02.c example cfs@PaD96
 *
 * Bissl a Zusatzwerte ausgeben
 *
 * OS timer writes to timerFile
 * thread (instead of ISR) waits (poll()) for timer message
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>     //memcpy
#include <stdint.h>     //uint64-t
#include <poll.h>
#include <time.h>
#include <sys/timerfd.h>
#include <errno.h>      //errno
#define  LOOP_FOREVER /**/

struct timespec tstrt, tres;
int             timerFD;
pthread_t       thTimer;

int timerReact(){
  struct timespec t,oldt;
  struct pollfd   pollfds[1]={{-1,POLLIN,0}};
  char   tmrbuf[10];
  pollfds[0].fd=timerFD;
  clock_gettime(CLOCK_REALTIME, &oldt);
  for(;LOOP_FOREVER;){
    //WAIT_FOR_CHANGES_ON_TIMER_HANDLE_:
      poll( pollfds, 1, -1);
      read( timerFD, tmrbuf, sizeof(uint64_t));

    //GET_EXACT_TIME_:
      clock_gettime(CLOCK_REALTIME, &t);

    //DISPLAY_REPORT_INFO_:
    printf("tReact @t=%4ld.%03lds d=%8ldus\n", t.tv_sec-tstrt.tv_sec, t.tv_nsec/1000000L
        ,
           (t.tv_sec-oldt.tv_sec)*1000000L + (t.tv_nsec-oldt.tv_nsec)/1000L );

    //STORE_TIME_STAMP_FOR_LATER_USE_:
    memcpy(&oldt,&t,sizeof(t));
  }
}


int main(int argc, char *argv[]){
  struct timespec wm={5L,0L};
  const struct itimerspec newTS={{0L,312345678L},{2L,10L}};
  int rv;
```

```
  //DO_INITIAL_STUFF_:
      if(0 > clock_gettime(CLOCK_REALTIME, &tstrt)){ perror("clock_gettime"); exit(
          EXIT_FAILURE);}
      printf("start @t=[%ld.%09ld]\n", tstrt.tv_sec, tstrt.tv_nsec);

      clock_getres(CLOCK_REALTIME, &tres);
      printf("CLOCK_REALTIME Granularity=%ldns - %ld/%ld\n", tres.tv_nsec, tres.tv_sec,
          tres.tv_nsec);

      clock_getres(CLOCK_MONOTONIC, &tres);
      printf("CLOCK_MONOTONIC Granularity=%ldns\n", tres.tv_nsec);

      clock_getres(CLOCK_PROCESS_CPUTIME_ID, &tres);
      printf("CLOCK_PROCESS_CPUTIME_ID Granularity=%ldns\n", tres.tv_nsec);
      printf("CLOCKS_PER_SEC =(%ld)\n", CLOCKS_PER_SEC);

  //START_CONSUMER_TIMER_INTERVAL_:
      timerFD=timerfd_create(CLOCK_REALTIME,0);
      timerfd_settime(timerFD, 0, &newTS, NULL);

  //START_PRODUCER_AND_CONSUMER_THREADS_:
      rv=pthread_create( &thTimer, NULL, &timerReact, NULL);

  //n_LET_THEM_WORK_IN_PEACE_:
      nanosleep( &wm, NULL);
      pthread_join( thTimer, NULL);

  //CLEAN_UP_:
  close(timerFD);
}
```

40.5.4 Scheduling + process synchronization

clock_nanosleep() + semaphore + scheduling:

Listing 70: clkNanoSleepX03.c example

```
/* clkNanoSleepX03.c example cfs@PbJbm
 *   Loesg. mit "clock_nanosleep( ... TIMER_ABSTIME ... )"
 *
 * a) Abfrage Systemzeit alle 100ms exakt
 * b) Statistik der Abweichungen von den 100ms
 *    Minimalwert, Maximalwert, Mittelwert, Varianz,
 *    Haufigkeit im Bereich +/-5% vom Mittelwert,
 *    Haufigkeit im Bereich >110% vom Mittelwert,
 *    Haufigkeit im Bereich < 90% vom Mittelwert
 * c) Systemzeit und Statistik in ansprechender HTML-Gestaltung
 *    jederzeit als Webseite abrufbar
 * d) per Kommandozeilenparameter die
 *    Auswahl der Scheduling-Strategie plus ggf. Prioritaet erlaubt:
 *    -- standart usermode Scheduling ("SCHED_OTHER")
 *    -- Round-Robin Scheduling ("SCHED_RR")
 *    -- FIFO Scheduling ("SCHED_FIFO")
 *
 * compile: gcc clkNanoSleepX03.c -o clkNSX03 -lrt -lpthread
 * run:               ./clkNSX03 <-r|-f> <IPportNr>
 * run m.SCHED_OTHER: ./clkNSX03 1234
 * run m.SCHED_RR   : ./clkNSX03 -r 1235
 * run m.SCHED_FIFO : ./clkNSX03 -f 1236
 * anbrowsen: http://<host>:1234, http://<host>:1235, http://<host>:1236
 */
#include <stdio.h>      //printf
#include <stdlib.h>
#include <string.h>     //memcpy
#include <stdint.h>     //uint64-t
#include <time.h>       //struct timespec
#include <signal.h>     //sigval_t
#include <semaphore.h>  //sem_init, sem_post, sem_wait
#include <limits.h>     //LONG_MAX
#include <unistd.h>     //pipe()
#include <pthread.h>    //pthread_create
#include <sys/prctl.h>  //PR_GET_TIMERSLACK
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define IpPort  (argv[argc-1])

#define DebugModeXH 0
#define tINTVL (100L*1000000L)

  //Statistik-Speicher:
struct statistik{
  double dtSum, sqSum, minVal, maxVal, avgVal, varVal;
  long   fq5Avg, fq10Minus, fq10Plus, valCnt;
} statsMem={0.0, 0.0, 1E30, 0.0, 0.0, 0.0, 0L, 0L, 0L, 0L};
  //ZeitSpeicher:
long            tickCtr=0;
struct timespec t;
  //timer:
timer_t         tickTmrID;
const struct itimerspec tmrStopSpec={{0L,0L},{0L,0L}};
pthread_t       tickTmrTH;
int             tickTmrSlackVal=0;
  //semaphore:
sem_t           sempho1;
int             semphoSharing=0;
  //scheduling:
int             schedPolicy=SCHED_OTHER, schedPol2, schedPrty2;
  //statistik-Prozess:
pthread_t       statsTH;
  //comm.pipe:
int             pipeFD[2];
#define pipeEINI  (pipeFD[1])
#define pipeAUSSA (pipeFD[0])

#define SCHEDULING_POLICY_AS_STRING(x) (\
        (SCHED_FIFO == (x)?"SCHED_FIFO":                \
          (SCHED_RR == (x)?"SCHED_RR":                  \
            (SCHED_OTHER == (x)?"SCHED_OTHER":"???")    \
          )     \
        )       \
  )

void *tickTmrSR(void *nix){ /*
-----------------------------------*/
  struct timespec       t1,t2;
  pthread_attr_t        attrR;
  struct sched_param    par1;
  pthread_getattr_np( tickTmrTH, &attrR );
  pthread_attr_getschedpolicy( &attrR, &schedPol2 );
  pthread_attr_getschedparam(  &attrR, &par1);
  schedPrty2=par1.sched_priority;
  printf("Policy:Wanted=[%s] Granted=[%s]/Prty=%d\n",
        SCHEDULING_POLICY_AS_STRING( schedPolicy ),
        SCHEDULING_POLICY_AS_STRING( schedPol2 ),
        schedPrty2 );
  for(;;){
    if(0!= clock_gettime(CLOCK_REALTIME, &t1)) perror("tickTmrSR clock_gettime failed.")
        ;
    write( pipeEINI, &t1, sizeof(t1) );
    if(0!= sem_post(&sempho1)) perror("tickTmrSR sem_post failed.");
    tickCtr++;
    t2.tv_nsec = (t1.tv_nsec+tINTVL)%1000000000L;
    t2.tv_sec  = t1.tv_sec+(t1.tv_nsec+tINTVL)/1000000000L;
    tickTmrSlackVal=prctl(PR_GET_TIMERSLACK,1,1,1,1);
    #if (DebugModeXH)
      printf("ticker:Nr=%4ld, t1=%ld.%09ld[s], t2=%ld.%09ld[s]\n",
                tickCtr, t1.tv_sec, t1.tv_nsec, t2.tv_sec, t2.tv_nsec);
    #endif
    clock_nanosleep(CLOCK_REALTIME, TIMER_ABSTIME, &t2, NULL);
  }
}


void *statsSR(void *nix){
```

```
  struct timespec tWert, tPrev;
  double        dt, dsq2;
  for(;;){
    if(0!= sem_wait(&sempho1)) perror("Stats sem_wait failed.");
    read( pipeAUSSA, &tWert, sizeof(tWert) );
    if(statsMem.valCnt++){
      dt= (tWert.tv_sec-tPrev.tv_sec) + (tWert.tv_nsec-tPrev.tv_nsec)/1000000000.0;
      statsMem.dtSum += dt;
      statsMem.avgVal = (statsMem.dtSum) /statsMem.valCnt;
      dsq2 = dt - statsMem.avgVal;
      statsMem.sqSum += (dsq2*dsq2);
      statsMem.varVal = statsMem.sqSum /statsMem.valCnt;
      if(dt > statsMem.maxVal) statsMem.maxVal = dt;
      if(dt < statsMem.minVal) statsMem.minVal = dt;
      if(0.95*statsMem.avgVal < dt  &&  dt < 1.05*statsMem.avgVal) statsMem.fq5Avg++;
      if( dt > 1.10*statsMem.avgVal ) statsMem.fq10Plus++;
      if( dt < 0.90*statsMem.avgVal ) statsMem.fq10Minus++;
      #if (DebugModeXH)
        printf("StatistikRechnung: tWert=%ld.%09ld", tWert.tv_sec, tWert.tv_nsec);
        printf(" dt=%lf\n", dt);
        printf("min=%g, max=%g, avg=%g, var=%g\n",
               statsMem.minVal, statsMem.maxVal, statsMem.avgVal, statsMem.varVal );
      #endif
    }
    memcpy(&tPrev, &tWert, sizeof(tWert));
  }
}


#define DO_INITIAL_KREMPEL(a)   {\
    if(argc<2){ printf("usage: %s <PortNumber> -r -f",*argv); exit(0); }        \
    schedPolicy=SCHED_OTHER;    \
    while ((opt = getopt(argc, argv, "rf")) != -1) {    \
      switch(opt){      \
      case 'f': schedPolicy=SCHED_FIFO; printf("FIFO %s\n",IpPort); break;      \
      case 'r': schedPolicy=SCHED_RR;   printf("RoundRobin %s\n",IpPort); break;
          \
      } \
    }   \
    struct timespec tstrt={5L,0L};      \
    struct timespec tres={5L,0L};               \
    if(0!= clock_gettime(CLOCK_REALTIME, &tstrt)) perror("init clock_gettime failed.");
        \
    if(0!= clock_getres(CLOCK_REALTIME, &tres))   perror("init clock_getres failed.");
        \
    if(0!= sem_init(&sempho1, semphoSharing, 0))  perror("init sem_init failed.");
        \
    if(0!= pipe(pipeFD)) perror("init pipe failed.");   \
    if(a) printf("start @t=[%ld.%09lds],\ttimerInterval=%ld\n"  \
        "CLOCK_REALTIME Granularity=%lds+%ldns,\tCLOCKS_PER_SEC =(%ld)\n",      \
        tstrt.tv_sec, tstrt.tv_nsec, tINTVL, tres.tv_sec, tres.tv_nsec, CLOCKS_PER_SEC);
                  \
  }
#define START_100ms_TIMER()     {\
    pthread_attr_t pat1;                \
    struct sched_param par1;            \
    pthread_attr_init(&pat1);           \
    pthread_attr_setdetachstate( &pat1, PTHREAD_CREATE_JOINABLE);       \
    pthread_attr_setinheritsched(&pat1, PTHREAD_EXPLICIT_SCHED);        \
    pthread_attr_setschedpolicy( &pat1, schedPolicy);   \
    par1.sched_priority=1;      \
    pthread_attr_setschedparam(&pat1, &par1);   \
    if(pthread_create( &tickTmrTH, &pat1, tickTmrSR, NULL)) perror("timer pthread_create
         failed.");     \
  }
#define START_STATISTIK()       {\
    if(0!= pthread_create( &statsTH, NULL, statsSR, NULL)) perror("stats pthread_create
        failed.");\
  }
#define MACH_LAN_SOCKET()       {\
    gethostname(hostName,255);  \
    fds=socket(AF_INET,SOCK_STREAM,0);  \
    s_address.sin_family=AF_INET;       \
```

```
    s_address.sin_addr.s_addr = htonl(INADDR_ANY);      \
    s_address.sin_port = htons(atoi(IpPort));   \
    if(bind(fds,(struct sockaddr *) &s_address, sizeof(s_address))){    \
      perror("bind failed"); abort();   \
    }   \
    listen(fds,5);\
  }
#define WAIT_some_Time()        {\
    struct timespec wm={8L,0L}; \
    nanosleep( &wm, NULL);      \
  }
#define KILL_Timer()    {\
    timer_settime(tickTmrID, 0, &tmrStopSpec, NULL);    \
    timer_delete(tickTmrID);    \
  }


  /*geordnetes Programm-Ende mit 'q' und Strg+C*/
int fds; //Achtung:GlobaleVariable
void exitCode(int snum, siginfo_t *sinfo, void *nix){
  shutdown(fds,SHUT_RDWR);
  close(fds);
  printf("*** ->server terminated on signal %i.\n", snum);
  exit(EXIT_SUCCESS);
}
void *exit_on_Q(void*nix){
  while( NULL == strchr("QqCc\003",getchar()) );
  exitCode(-12345, NULL, NULL);
}
#define INSTALL_EXIT_FUNCTION() {\
    struct sigaction sa;        \
    sa.sa_handler       = NULL; \
    sa.sa_sigaction     = exitCode;     \
    sa.sa_flags         = SA_SIGINFO;   \
    sigemptyset(&sa.sa_mask);   \
    if(sigaction(SIGABRT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGALRM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGHUP , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGINT , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGQUIT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTERM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTSTP, &sa, NULL)) perror("setting sigaction() failed.");    \
    pthread_t   THx;    \
    pthread_create( &THx, NULL, exit_on_Q, NULL);       \
    alarm(300); /*stop after 300[s]*/   \
  }




  //***************//
   //*** MAIN ***//
    //**********//
      //*****//
int main(int argc, char *argv[]){
  int           fdc, lenC,r1=0, opt;
  char          buf[4096], hostName[256];
  struct sockaddr_in    s_address, c_address;

  DO_INITIAL_KREMPEL(1);

  START_100ms_TIMER();
  START_STATISTIK();
  MACH_LAN_SOCKET();
  INSTALL_EXIT_FUNCTION();

  for(;;){
   printf("server \"%s\" @ \"%s\" waiting\n", argv[0], hostName);
   lenC=sizeof(c_address);
   fdc=accept(fds,(struct sockaddr *) &c_address, &lenC);
    if(1 > (r1=read(fdc, buf, sizeof(buf)-2)) ) break; //buf[r1]=0;
    sprintf(buf,
        "HTTP/1.0 200 OK\r\n"
```

```
        "Server: RWS\r\n"
        "Content-Type: text/html\r\n\r\n"
        "<html><head><title>RWS - ein kleiner Webserver</title></head>\n"
        "<body bgcolor=tan><h2>NWES3a</h2><hr>\n"
        "<div align=center bgcolor=silver>"
        "<table border=1 align=center bgcolor=wheat"
        "       style=\"font:500 14pt monospace;\">\n"
        " <tr><td colspan=2 style='font:900 14pt serif;'>"
        "    100ms Timer Statistik timer_create():  \n"
        " <tr><td bgcolor=silver>sched.policy<td bgcolor=silver>%s /%d\n"
        " <tr><td bgcolor=silver>timerSlack<td bgcolor=silver>%d\n"
        " <tr><td bgcolor=silver>values<td bgcolor=silver>%ld\n"
        " <tr><td bgcolor=yellow>range <td bgcolor=yellow align=center>[%g, %g]\n"
        " <tr><td bgcolor=red   >&Delta;<td bgcolor=red align=center>%g (%g \%)\n"
        " <tr><td bgcolor=orange>average<td bgcolor=orange align=center>%g\n"
        " <tr><td bgcolor=aqua  >variance<td bgcolor=aqua   align=center>%g\n"
        " <tr><td colspan=2 align=center>frequencies:"
        " <tr><td bgcolor='#FFD0C0'>&lt; 90%% avg<td>%ld"
        " <tr><td bgcolor='#FFD0C0'>95..105 %% avg<td>%ld"
        " <tr><td bgcolor='#FFD0C0'>&gt;110 %% avg<td>%ld"
        "</table></div><hr>\n"
        "<h9>aus (c)2005 richard weinberger</h9>"
        "<script>"
        "  window.setInterval('reloader()', 3000);"
        "  function reloader(){"
        "     window.location.reload();"
        "  }"
        "</script></body></html>\r\n\r\n",
        SCHEDULING_POLICY_AS_STRING( schedPol2 ),
        schedPrty2, tickTmrSlackVal,
        statsMem.valCnt,
        statsMem.minVal, statsMem.maxVal, statsMem.maxVal-statsMem.minVal,
        100.0*(statsMem.maxVal-statsMem.minVal)/statsMem.avgVal,
        statsMem.avgVal, statsMem.varVal,
        statsMem.fq10Minus, statsMem.fq5Avg, statsMem.fq10Plus
        );
    r1=write(fdc, buf, strlen(buf));
    close(fdc);
  }
  shutdown(fds,SHUT_RDWR);
  close(fds);
  printf("\nserver \"%s\" terminated.\n",*argv);

  KILL_Timer();
  printf("done.\n");
}
```

40.6 sleep(), usleep(), nanosleep(), clock_nanosleep()

Mit Prozess-Parallelisierung und Warten lässt sich Vieles code-mäßig sogar vereinfachen! Anstelle verschachtelter Schleifen mit mehreren *if(dies und jenes)* schreibt man etwa

```
parallelTask(){
  forever{
     tuDeinJob;
     warte;
  }
}
```

Der Warterei kommt im Multitasking eine zentrale Rolle zu, weil es jedesmal einen Taskwechsel erwirkt. → So kann das gesamte OS auch mit höchstpriorisierten Prozessen (s. SCHED_FIFO Scheduling) (mehr oder minder) normal weiterlaufen.

40.6.1 nanosleep()

ist das richtige Werkzeug. Es ist
threadsafe und
asynchron signal safe.
Auf Grund der feinen Auflösung ist die Wartezeit (*req* wie *requested*) in 2 struct-Komponenten aufgeteilt: Sekunden und Nanosekunden *(The value of the nanoseconds field must be in the range 0 to 999999999)*. Den zweiten Parameter (*rem* wie *remainig*) braucht man selten, den setzt man *NULL*. Aus der manpage *man 2 nanosleep*:

```
#include <time.h>
int nanosleep(const struct timespec *req, struct timespec *rem);

   struct timespec {
       time_t tv_sec;        /* seconds */
       long   tv_nsec;       /* nanoseconds */
   };

Compared to sleep(3) and usleep(3),
nanosleep() has the following advantages:
it provides a higher resolution for specifying the sleep interval;
POSIX.1 explicitly specifies that it does not interact with signals;
```

```
and it makes the task of resuming a sleep that has been interrupted
by a signal handler easier.
```

40.6.2 usleep()

Mikro-Sleep klingt zwar gut, aber der Haken ist (aus *man 3 usleep*):

```
#include <unistd.h>
int usleep(useconds_t usec);
       The usleep() function suspends execution of the calling thread for (at least)
       usec microseconds. The sleep may be lengthened slightly by any system activity
       or by the time spent processing the call or by the granularity of system timers.
ERRORS
       EINTR  Interrupted by a signal; see signal(7).
       EINVAL usec is not smaller than 1000000.
       (On systems where that is considered an error.)

       The  interaction  of this function with the SIGALRM signal,
       and with other timer functions such as alarm(2), sleep(3),
       nanosleep(2), setitimer(2), timer_create(2), timer_delete(2),
       timer_getoverrun(2), timer_gettime(2), timer_settime(2), ualarm(3)
       is unspecified.
```

Man sieht schon, das ist nicht so ganz das Richtige für präzises Timing und Multitasking.

40.6.3 sleep(sekunden)

(nicht zu verwechseln mit dem shell-Kommando *'sleep'*) ist der groben Auflösung wegen für Multitasking und damit hier kaum brauchbar.
Um zB. in nicht-parallelisierter Umgebung zwischen zwei Webseitenabrufen einige Sekunden zu verschlafen, ist es die schlichteste Variante.
sleep gibts auch als shell-Kommando (man 1 sleep), zB um wiederholte Abfragen zu bremsen, wie etwa in

```
while((1)); do df -k >> prot; sleep 5;
    done
```

um Speicherplatzveränderungen zu protokollieren.

40.6.4 Aufgabe 2x nanosleep()

Schreib ein C-Programm, welches

1. alle 500ms ein '+' Plus ausgibt
2. alle 2,8s "NeueZeile" schreibt und eine neue Zeile beginnt
3. den Zugriff auf *stdout* (also putchar(), printf()...) per *mutex* serialisiert.

Lösung *plusUndCR1.c*:

Listing 71: Bsp: plusUndCR1.c

```
/* plusUndCR1.c, cfs@PbEi6
 * Demo: alle 500ms '+' ausgeben; alle 2.8s "NeueZeile\n" ausgeben
 * Loesung mit Threads; (ungesichert = ohne Mutex)
 * compile: gcc plusUndCR1.c -o plusUndCR1 -lpthread
 * run    : ./plusUndCR1
 */
#include <stdio.h>      //printf
#include <time.h>       //struct timespec
#include <pthread.h>    //threads

pthread_t       thp1, thp2;

void *sayPlus(void *nix){
  struct timespec wt={ 0L, 500000000L };//[s], [ns]
  for(;;){      //endless
    putchar('+');
    fflush(stdout);
    nanosleep( &wt, NULL);
  }
}

void *sayCR(void *nix){
  struct timespec wt={ 2L, 800000000L };//[s], [ns]
  for(;;){      //endless
    printf("NeueZeile\n");
    nanosleep( &wt, NULL);
  }
}


main(){
  if(pthread_create( &thp1, NULL, &sayPlus , NULL)) perror("nix Plus");
  if(pthread_create( &thp2, NULL, &sayCR,   NULL)) perror("nix CR");
  pthread_join(thp1, NULL);
  pthread_join(thp2, NULL);
}
```

40.6.5 Aufgabe: Mit BBB-LEDs blinken

Das BBB hat 4 Stk blitzblauhelle SMD-LEDs auf dem Board neben der LAN-Buchse. Sie heißen USER0..USER3 und sie blinken normalerweise von selber *irgendwie* (=Hartbeat, Busy, eMMC, SD) herum. Ohne SD ist USER1 dauerfinster, sodass wir sie inanspruch nehmen können. Und zwar lassen wir sie nicht einfach nur doof blinken, sondern steigern ihre Helligkeit wiederholt in 1/10 Stufen mittels selbstgemachter *PWM* (mit Wartezeiten)

Listing 72: Bsp: L1pw.c

```
/* L1pw.c: toggle USER1 LED  digitalIOpin, cfs@PbD
 * compile: gcc L1pw.c -o L1pw.bin
 * run:     ./L1pw.bin
 */
#include <stdlib.h>     //f. atexit()
#include <stdio.h>      //f. printf()
#include <string.h>     //f. strcpy(), strlen()
#include <fcntl.h>      //f. O_RDONLY, O_WRONLY, open,read,write,close
#include <time.h>       //f. nanosleep()
#include <pthread.h>
#define P_USER1 "/sys/class/leds/beaglebone:green:usr1/brightness"
#define PERIODNS 10000000L

long    periodns= PERIODNS;
struct timespec thi={ 0L, PERIODNS/2L},
                tlo={ 0L, 10L};
char    text[10], hiwert[]="1", lowert[]="0";
int     fd;
pthread_t thp1;
void    gegentum(void);


void *heller(void *nix){
  struct timespec dawei={ 0L, 500000000L};
  int   pw=1;
  for(;;){
    //adjust duty cycle:
        if(pw++ > 10) pw=0;
        thi.tv_nsec = periodns*pw/10;
    //wait
        nanosleep( &dawei, NULL);
  }
}


main(){
  atexit(&gegentum);    //was tun am Schluss?

  if(pthread_create( &thp1, NULL, &heller, NULL)) perror("thp1 not started.");

  for(;;){  //endlos
    //digitalPin HI schreiben:
        fd=open(P_USER1, O_WRONLY);
        write(fd, hiwert, strlen(hiwert));
        close(fd);
        nanosleep( &thi, NULL);  //warten
    //digitalPin LO schreiben:
        fd=open(P_USER1, O_WRONLY);
        write(fd, lowert, strlen(lowert));
        close(fd);
        tlo.tv_nsec = periodns - thi.tv_nsec;
        if( tlo.tv_nsec <10L ) tlo.tv_nsec=10L;
        if( tlo.tv_nsec >123456789L ) tlo.tv_nsec=123456789L;
        nanosleep( &tlo, NULL);
  }
  pthread_join(thp1, NULL);
}

void gegentum(void){
  printf("nix aufzrauma\n");
}
```

40.7 clock_nanosleep() Analyse als Webseite rtClockNanosleep

```
/* clkNanoSleepX03.c example cfs@PbJbm
 *   Loesg. mit "clock_nanosleep( ... TIMER_ABSTIME ... )"
 *
 * a) Abfrage Systemzeit alle 100ms exakt
 * b) Statistik der Abweichungen von den 100ms
 *    Minimalwert, Maximalwert, Mittelwert, Varianz,
 *    Haufigkeit im Bereich +/-5% vom Mittelwert,
 *    Haufigkeit im Bereich >110% vom Mittelwert,
 *    Haufigkeit im Bereich < 90% vom Mittelwert
 * c) Systemzeit und Statistik in ansprechender HTML-Gestaltung
 *    jederzeit als Webseite abrufbar
```

```
 * d) per Kommandozeilenparameter die
 *    Auswahl der Scheduling-Strategie plus ggf. Prioritaet erlaubt:
 *    -- standart usermode Scheduling ("SCHED_OTHER")
 *    -- Round-Robin Scheduling ("SCHED_RR")
 *    -- FIFO Scheduling ("SCHED_FIFO")
 *
 * compile: gcc clkNanoSleepX03.c -o clkNSX03 -lrt -lpthread
 * run:               ./clkNSX03 <-r|-f> <IPportNr>
 * run m.SCHED_OTHER: ./clkNSX03 1234
 * run m.SCHED_RR   : ./clkNSX03 -r 1235
 * run m.SCHED_FIFO : ./clkNSX03 -f 1236
 * anbrowsen: http://<host>:1234, http://<host>:1235, http://<host>:1236
 */
#include <stdio.h>      //printf
#include <stdlib.h>
#include <string.h>     //memcpy
#include <stdint.h>     //uint64-t
#include <time.h>       //struct timespec
#include <signal.h>     //sigval_t
#include <semaphore.h>  //sem_init, sem_post, sem_wait
#include <limits.h>     //LONG_MAX
#include <unistd.h>     //pipe()
#include <pthread.h>    //pthread_create
#include <sys/prctl.h>  //PR_GET_TIMERSLACK
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define IpPort  (argv[argc-1])

#define DebugModeXH 0
#define tINTVL (100L*1000000L)

  //Statistik-Speicher:
struct statistik{
  double dtSum, sqSum, minVal, maxVal, avgVal, varVal;
  long   fq5Avg, fq10Minus, fq10Plus, valCnt;
} statsMem={0.0, 0.0, 1E30, 0.0, 0.0, 0.0, 0L, 0L, 0L, 0L};
  //ZeitSpeicher:
long            tickCtr=0;
struct timespec t;
  //timer:
timer_t         tickTmrID;
const struct itimerspec tmrStopSpec={{0L,0L},{0L,0L}};
pthread_t       tickTmrTH;
int             tickTmrSlackVal=0;
  //semaphore:
sem_t           sempho1;
int             semphoSharing=0;
  //scheduling:
int             schedPolicy=SCHED_OTHER, schedPol2, schedPrty2;
  //statistik-Prozess:
pthread_t       statsTH;
  //comm.pipe:
int             pipeFD[2];
#define pipeEINI  (pipeFD[1])
#define pipeAUSSA (pipeFD[0])

void *tickTmrSR(void *nix){
  struct timespec       t1,t2;
  pthread_attr_t        attrR;
  struct sched_param    par1;
  pthread_getattr_np( tickTmrTH, &attrR );
  pthread_attr_getschedpolicy( &attrR, &schedPol2 );
  pthread_attr_getschedparam(  &attrR, &par1);
  schedPrty2=par1.sched_priority;
  printf("Policy:Wanted=[%s] Granted=[%s]/%d\n",

(SCHED_FIFO==schedPolicy?"SCHED_FIFO":(SCHED_RR==schedPolicy?"SCHED_RR":(SCHED_OTHER==
    schedPolicy?"SCHED_OTHER":"???"))),\
        (SCHED_FIFO==schedPol2?"SCHED_FIFO":(SCHED_RR==schedPol2?"SCHED_RR":(
            SCHED_OTHER==schedPol2?"SCHED_OTHER":"???"))),\
        schedPrty2 );\
```

```
  for(;;){
    if(0!= clock_gettime(CLOCK_REALTIME, &t1)) perror("tickTmrSR clock_gettime failed.
        ");
    write( pipeEINI, &t1, sizeof(t1) );
    if(0!= sem_post(&sempho1)) perror("tickTmrSR sem_post failed.");
    tickCtr++;
    t2.tv_nsec = (t1.tv_nsec+tINTVL)%1000000000L;
    t2.tv_sec  = t1.tv_sec+(t1.tv_nsec+tINTVL)/1000000000L;
    tickTmrSlackVal=prctl(PR_GET_TIMERSLACK,1,1,1,1);
    #if (DebugModeXH)
      printf("ticker:Nr=%4ld, t1=%ld.%09ld[s], t2=%ld.%09ld[s]\n",
              tickCtr, t1.tv_sec, t1.tv_nsec, t2.tv_sec, t2.tv_nsec);
    #endif
    clock_nanosleep(CLOCK_REALTIME, TIMER_ABSTIME, &t2, NULL);
  }
}

void *statsSR(void *nix){
  struct timespec tWert, tPrev;
  double        dt, dsq2;
  for(;;){
    if(0!= sem_wait(&sempho1)) perror("Stats sem_wait failed.");
    read( pipeAUSSA, &tWert, sizeof(tWert) );
    if(statsMem.valCnt++){
      dt= (tWert.tv_sec-tPrev.tv_sec) + (tWert.tv_nsec-tPrev.tv_nsec)/1000000000.0;
      statsMem.dtSum += dt;
      statsMem.avgVal = (statsMem.dtSum) /statsMem.valCnt;
      dsq2 = dt - statsMem.avgVal;
      statsMem.sqSum += (dsq2*dsq2);
      statsMem.varVal = statsMem.sqSum /statsMem.valCnt;
      if(dt > statsMem.maxVal) statsMem.maxVal = dt;
      if(dt < statsMem.minVal) statsMem.minVal = dt;
      if(0.95*statsMem.avgVal < dt  &&  dt < 1.05*statsMem.avgVal) statsMem.fq5Avg++;
      if( dt > 1.10*statsMem.avgVal ) statsMem.fq10Plus++;
      if( dt < 0.90*statsMem.avgVal ) statsMem.fq10Minus++;
      #if (DebugModeXH)
        printf("StatistikRechnung: tWert=%ld.%09ld", tWert.tv_sec, tWert.tv_nsec);
        printf(" dt=%lf\n", dt);
        printf("min=%g, max=%g, avg=%g, var=%g\n",
                statsMem.minVal, statsMem.maxVal, statsMem.avgVal, statsMem.varVal );
      #endif
    }
    memcpy(&tPrev, &tWert, sizeof(tWert));
  }
}


#define DO_INITIAL_KREMPEL(a)   {\
    if(argc<2){ printf("usage: %s <PortNumber> -r -f",*argv); exit(0); }        \
    schedPolicy=SCHED_OTHER;    \
    while ((opt = getopt(argc, argv, "rf")) != -1) {    \
      switch(opt){      \
      case 'f': schedPolicy=SCHED_FIFO; printf("FIFO %s\n",IpPort); break;      \
      case 'r': schedPolicy=SCHED_RR;   printf("RoundRobin %s\n",IpPort); break;
                \
      } \
    }   \
    struct timespec tstrt={5L,0L};      \
    struct timespec tres={5L,0L};               \
    if(0!= clock_gettime(CLOCK_REALTIME, &tstrt)) perror("init clock_gettime failed.")
        ; \
    if(0!= clock_getres(CLOCK_REALTIME, &tres))   perror("init clock_getres failed.");
          \
    if(0!= sem_init(&sempho1, semphoSharing, 0))  perror("init sem_init failed.");
             \
    if(0!= pipe(pipeFD)) perror("init pipe failed.");   \
    if(a) printf("start @t=[%ld.%09lds],\ttimerInterval=%ld\n"  \
        "CLOCK_REALTIME Granularity=%lds+%ldns,\tCLOCKS_PER_SEC =(%ld)\n",      \
        tstrt.tv_sec, tstrt.tv_nsec, tINTVL, tres.tv_sec, tres.tv_nsec, CLOCKS_PER_SEC
            );        \
  }
#define START_100ms_TIMER()     {\
    pthread_attr_t pat1;                \
```

```
    struct sched_param par1;            \
    pthread_attr_init(&pat1);           \
    pthread_attr_setdetachstate( &pat1, PTHREAD_CREATE_JOINABLE);       \
    pthread_attr_setinheritsched(&pat1, PTHREAD_EXPLICIT_SCHED);        \
    pthread_attr_setschedpolicy( &pat1, schedPolicy);   \
    par1.sched_priority=1;      \
    pthread_attr_setschedparam(&pat1, &par1);   \
    if(pthread_create( &tickTmrTH, &pat1, tickTmrSR, NULL)) perror("timer
        pthread_create failed.");     \
  }
#define START_STATISTIK()       {\
    if(0!= pthread_create( &statsTH, NULL, statsSR, NULL)) perror("stats
        pthread_create failed.");\
  }
#define MACH_LAN_SOCKET()       {\
    gethostname(hostName,255);  \
    fds=socket(AF_INET,SOCK_STREAM,0);  \
    s_address.sin_family=AF_INET;       \
    s_address.sin_addr.s_addr = htonl(INADDR_ANY);      \
    s_address.sin_port = htons(atoi(IpPort));   \
    if(bind(fds,(struct sockaddr *) &s_address, sizeof(s_address))){    \
      perror("bind failed"); abort();   \
    }   \
    listen(fds,5);\
  }
#define WAIT_some_Time()        {\
    struct timespec wm={8L,0L}; \
    nanosleep( &wm, NULL);      \
  }
#define KILL_Timer()    {\
    timer_settime(tickTmrID, 0, &tmrStopSpec, NULL);    \
    timer_delete(tickTmrID);    \
  }


  /*geordnetes Programm-Ende mit 'q' und Strg+C*/
int fds; //Achtung:GlobaleVariable
void exitCode(int snum, siginfo_t *sinfo, void *nix){
  shutdown(fds,SHUT_RDWR);
  close(fds);
  printf("*** ->server terminated on signal %i.\n", snum);
  exit(EXIT_SUCCESS);
}
void *exit_on_Q(void*nix){
  while( NULL == strchr("QqCc\003",getchar()) );
  exitCode(-12345, NULL, NULL);
}
#define INSTALL_EXIT_FUNCTION() {\
    struct sigaction sa;        \
    sa.sa_handler       = NULL; \
    sa.sa_sigaction     = exitCode;     \
    sa.sa_flags         = SA_SIGINFO;   \
    sigemptyset(&sa.sa_mask);   \
    if(sigaction(SIGABRT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGALRM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGHUP , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGINT , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGQUIT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTERM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTSTP, &sa, NULL)) perror("setting sigaction() failed.");    \
    pthread_t   THx;    \
    pthread_create( &THx, NULL, exit_on_Q, NULL);       \
    alarm(300); /*stop after 300[s]*/   \
  }


  //***************//
   //*** MAIN ***//
    //**********//
      //*****//
int main(int argc, char *argv[]){
  int           fdc, lenC,r1=0, opt;
  char          buf[4096], hostName[256];
```

```
  struct sockaddr_in    s_address, c_address;

  DO_INITIAL_KREMPEL(1);

  START_100ms_TIMER();
  START_STATISTIK();
  MACH_LAN_SOCKET();
  INSTALL_EXIT_FUNCTION();

  for(;;){
   printf("server \"%s\" @ \"%s\" waiting\n", argv[0], hostName);
   lenC=sizeof(c_address);
   fdc=accept(fds,(struct sockaddr *) &c_address, &lenC);
    if(1 > (r1=read(fdc, buf, sizeof(buf)-2)) ) break; //buf[r1]=0;
    sprintf(buf,
        "HTTP/1.0 200 OK\r\n"
        "Server: RWS\r\n"
        "Content-Type: text/html\r\n\r\n"
        "<html><head><title>RWS - ein kleiner Webserver</title></head>\n"
        "<body bgcolor=tan><h2>NWES3a</h2><hr>\n"
        "<div align=center bgcolor=silver>"
        "<table border=1 align=center bgcolor=wheat"
        "  style=\"font:500 14pt monospace;\">\n"
        "<tr><td colspan=2 style='font:900 14pt serif;'>"
        "    100ms Timer Statistik timer_create():  \n"
        "<tr><td bgcolor=silver>sched.policy<td bgcolor=silver>%s /%d\n"
        "<tr><td bgcolor=silver>timerSlack<td bgcolor=silver>%d\n"
        "<tr><td bgcolor=silver>values<td bgcolor=silver>%ld\n"
        "<tr><td bgcolor=yellow>range <td bgcolor=yellow align=center>[%g, %g]\n"
        "<tr><td bgcolor=red   >.<td bgcolor=red align=center>%g (%g \%)\n"
        "<tr><td bgcolor=orange>average<td bgcolor=orange align=center>%g\n"
        "<tr><td bgcolor=aqua  >variance<td bgcolor=aqua   align=center>%g\n"
        "<tr><td colspan=2 align=center>frequencies:"
        "<tr><td bgcolor='#FFD0C0'>< 90%% avg<td>%ld"
        "<tr><td bgcolor='#FFD0C0'>95..105 %% avg<td>%ld"
        "<tr><td bgcolor='#FFD0C0'>>110 %% avg<td>%ld"
        "</table></div><hr>\n"
        "<h9>aus (c)2005 richard weinberger</h9>"
        "<script>"
        "window.setInterval('reloader()', 3000);"
        "function reloader(){"
        "  window.location.reload();"
        "}</script>"
        "</body></html>\r\n"
        "\r\n",
        (SCHED_FIFO==schedPol2?"SCHED_FIFO":(SCHED_RR==schedPol2?"SCHED_RR":(
            SCHED_OTHER==schedPol2?"SCHED_OTHER":"???"))),
        schedPrty2, tickTmrSlackVal,
        statsMem.valCnt,
        statsMem.minVal, statsMem.maxVal, statsMem.maxVal-statsMem.minVal,
        100.0*(statsMem.maxVal-statsMem.minVal)/statsMem.avgVal,
        statsMem.avgVal, statsMem.varVal,
        statsMem.fq10Minus, statsMem.fq5Avg, statsMem.fq10Plus
        );
    r1=write(fdc, buf, strlen(buf));
    close(fdc);
  }
  shutdown(fds,SHUT_RDWR);
  close(fds);
  printf("\nserver \"%s\" terminated.\n",*argv);

  KILL_Timer();
  printf("done.\n");
}
```

40.8 Timer Jitter Analyse als Webseite

Listing 73: TimerJitterAnalyse dlyJittAnX04.c

```
/* dlyJittAnX04.c Linux example cfs@PbS7i
 *   Loesg. mit "clock_nanosleep( ... TIMER_ABSTIME ... )"
 *         +mit HeaderFile clkNsleepX04.h
```

```
 *
 * a) Abfrage Systemzeit alle 100ms exakt
 * b) Statistik der Abweichungen von den 100ms
 *    Minimalwert, Maximalwert, Mittelwert, Varianz,
 *    Haufigkeit im Bereich +/-5% vom Mittelwert,
 *    Haufigkeit im Bereich >110% vom Mittelwert,
 *    Haufigkeit im Bereich < 90% vom Mittelwert
 * c) Systemzeit und Statistik in ansprechender HTML-Gestaltung
 *    jederzeit als Webseite abrufbar
 * d) per Kommandozeilenparameter die
 *    Auswahl der Scheduling-Strategie plus ggf. Prioritaet erlaubt:
 *    -- standart usermode Scheduling ("SCHED_OTHER")
 *    -- Round-Robin Scheduling ("SCHED_RR")
 *    -- FIFO Scheduling ("SCHED_FIFO")
 *
 * compile: gcc dlyJittAnX04.c -o dlyJAX04 -lrt -lpthread
 * run:               ./dlyJAX04 <-r|-f> <IPportNr>
 * run m.SCHED_OTHER: ./dlyJAX04 1234
 * run m.SCHED_RR   : ./dlyJAX04 -r 1235
 * run m.SCHED_FIFO : ./dlyJAX04 -f 1236
 * anbrowsen: http://<host>:1234, http://<host>:1235, http://<host>:1236
 */
#include <stdio.h>      //printf
#include <stdlib.h>
#include <string.h>     //memcpy
#include <stdint.h>     //uint64-t
#include <time.h>       //struct timespec
#include <signal.h>     //sigval_t
#include <semaphore.h>  //sem_init, sem_post, sem_wait
#include <limits.h>     //LONG_MAX
#include <unistd.h>     //pipe()
#include <pthread.h>    //pthread_create
#include <sys/prctl.h>  //PR_GET_TIMERSLACK
//#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define IpPort  (argv[argc-1])
#define pXit(s) {perror(s">Exit");exit(EXIT_FAILURE);}
#define DebugModeXH 0
#define tINTVL (100L*1000000L)
#include "wsock.h"

  //Statistik-Speicher:
struct statistik{
  double dtSum, sqSum, minVal, maxVal, avgVal, varVal;
  long   fq5Avg, fq10Minus, fq10Plus, valCnt;
} statsMem={0.0, 0.0, 1E30, 0.0, 0.0, 0.0, 0L, 0L, 0L, 0L};
  //ZeitSpeicher:
long            tickCtr=0;
struct timespec t;
  //timer:
timer_t         tickTmrID;
const struct itimerspec tmrStopSpec={{0L,0L},{0L,0L}};
pthread_t       tickTmrTH;
int             tickTmrSlackVal=0;
  //semaphore:
sem_t           sempho1;
int             semphoSharing=0;
  //scheduling:
int             schedPolicy=SCHED_OTHER, schedPol2, schedPrty2;
  //statistik-Prozess:
pthread_t       statsTH;
  //comm.pipe:
int             pipeFD[2];
#define pipeEINI  (pipeFD[1])
#define pipeAUSSA (pipeFD[0])

#define SCHEDULING_POLICY_AS_STRING(x) (\
        (SCHED_FIFO == (x)?"SCHED_FIFO":                \
          (SCHED_RR == (x)?"SCHED_RR":                  \
            (SCHED_OTHER == (x)?"SCHED_OTHER":"???")    \
          )     \
```

```
        )       \
  )

void *tickTmrSR(void *nix){
  struct timespec       t1,t2;
  pthread_attr_t        attrR;
  struct sched_param    par1;
  pthread_getattr_np( tickTmrTH, &attrR );
  pthread_attr_getschedpolicy( &attrR, &schedPol2 );
  pthread_attr_getschedparam(  &attrR, &par1);
  schedPrty2=par1.sched_priority;
  printf("Policy:Wanted=[%s] Granted=[%s]/Prty=%d\n",
        SCHEDULING_POLICY_AS_STRING( schedPolicy ),\
        SCHEDULING_POLICY_AS_STRING( schedPol2 ),\
        schedPrty2 );\
  for(;;){
    if(0!= clock_gettime(CLOCK_REALTIME, &t1)) perror("tickTmrSR clock_gettime failed.");
    write( pipeEINI, &t1, sizeof(t1) );
    if(0!= sem_post(&sempho1)) perror("tickTmrSR sem_post failed.");
    tickCtr++;
    t2.tv_nsec = (t1.tv_nsec+tINTVL)%1000000000L;
    t2.tv_sec  = t1.tv_sec+(t1.tv_nsec+tINTVL)/1000000000L;
    tickTmrSlackVal=prctl(PR_GET_TIMERSLACK,1,1,1,1);
    #if (DebugModeXH)
      printf("ticker:Nr=%4ld, t1=%ld.%09ld[s], t2=%ld.%09ld[s]\n",
                tickCtr, t1.tv_sec, t1.tv_nsec, t2.tv_sec, t2.tv_nsec);
    #endif
    clock_nanosleep(CLOCK_REALTIME, TIMER_ABSTIME, &t2, NULL);
  }
}

void *statsSR(void *nix){
  struct timespec tWert, tPrev;
  double        dt, dsq2;
  for(;;){
    if(0!= sem_wait(&sempho1)) perror("Stats sem_wait failed.");
    read( pipeAUSSA, &tWert, sizeof(tWert) );
    if(statsMem.valCnt++){
      dt= (tWert.tv_sec-tPrev.tv_sec) + (tWert.tv_nsec-tPrev.tv_nsec)/1000000000.0;
      statsMem.dtSum += dt;
      statsMem.avgVal = (statsMem.dtSum) /statsMem.valCnt;
      dsq2 = dt - statsMem.avgVal;
      statsMem.sqSum += (dsq2*dsq2);
      statsMem.varVal = statsMem.sqSum /statsMem.valCnt;
      if(dt > statsMem.maxVal) statsMem.maxVal = dt;
      if(dt < statsMem.minVal) statsMem.minVal = dt;
      if(0.95*statsMem.avgVal < dt  &&  dt < 1.05*statsMem.avgVal) statsMem.fq5Avg++;
      if( dt > 1.10*statsMem.avgVal ) statsMem.fq10Plus++;
      if( dt < 0.90*statsMem.avgVal ) statsMem.fq10Minus++;
      #if (DebugModeXH)
        printf("StatistikRechnung: tWert=%ld.%09ld", tWert.tv_sec, tWert.tv_nsec);
        printf(" dt=%lf\n", dt);
        printf("min=%g, max=%g, avg=%g, var=%g\n",
                statsMem.minVal, statsMem.maxVal, statsMem.avgVal, statsMem.varVal );
      #endif
    }
    memcpy(&tPrev, &tWert, sizeof(tWert));
  }
}


#define DO_INITIAL_KREMPEL(a)   {\
    if(argc<2){ printf("usage: %s <PortNumber> -r -f",*argv); exit(0); }        \
    schedPolicy=SCHED_OTHER;    \
    while ((opt = getopt(argc, argv, "rf")) != -1) {    \
      switch(opt){      \
      case 'f': schedPolicy=SCHED_FIFO; printf("FIFO %s\n",IpPort); break;      \
      case 'r': schedPolicy=SCHED_RR;   printf("RoundRobin %s\n",IpPort); break;        \
      } \
    }   \
    struct timespec tstrt={5L,0L};      \
    struct timespec tres={5L,0L};               \
    if(0!= clock_gettime(CLOCK_REALTIME, &tstrt)) pXit("init clock_gettime failed.");   \
```

```
    if(0!= clock_getres(CLOCK_REALTIME, &tres))   pXit("init clock_getres failed.");    \
    if(0!= sem_init(&sempho1, semphoSharing, 0))  pXit("init sem_init failed.");        \
    if(0!= pipe(pipeFD)) pXit("init pipe failed.");     \
    if(a) printf("start @t=[%ld.%09ld],\ttimerInterval=%ld\n"  \
        "CLOCK_REALTIME Granularity=%lds+%ldns,\tCLOCKS_PER_SEC =(%ld)\n",      \
        tstrt.tv_sec, tstrt.tv_nsec, tINTVL, tres.tv_sec, tres.tv_nsec, CLOCKS_PER_SEC);
                    \
  }
#define START_100ms_TIMER()     {\
    pthread_attr_t pat1;                \
    struct sched_param par1;            \
    pthread_attr_init(&pat1);           \
    pthread_attr_setdetachstate( &pat1, PTHREAD_CREATE_JOINABLE);       \
    pthread_attr_setinheritsched(&pat1, PTHREAD_EXPLICIT_SCHED);        \
    pthread_attr_setschedpolicy( &pat1, schedPolicy);   \
    par1.sched_priority=1;      \
    pthread_attr_setschedparam(&pat1, &par1);   \
    if(pthread_create( &tickTmrTH, &pat1, tickTmrSR, NULL)) pXit("--tickTimer
        pthread_create FAILED!");\
  }
#define START_STATISTIK()       {\
    if(0!= pthread_create( &statsTH, NULL, statsSR, NULL)) pXit("stats pthread_create
        FAILED!");\
  }
#define MACH_LAN_SOCKET()       {\
    gethostname(hostName,255);  \
    fds=socket(AF_INET,SOCK_STREAM,0);  \
    s_address.sin_family=AF_INET;       \
    s_address.sin_addr.s_addr = htonl(INADDR_ANY);      \
    s_address.sin_port = htons(atoi(IpPort));   \
    if(bind(fds,(struct sockaddr *) &s_address, sizeof(s_address))) pXit("bind failed"); \
    listen(fds,5);\
  }
#define WAIT_some_Time()        {\
    struct timespec wm={8L,0L}; \
    nanosleep( &wm, NULL);      \
  }
#define KILL_Timer()    {\
    timer_settime(tickTmrID, 0, &tmrStopSpec, NULL);    \
    timer_delete(tickTmrID);    \
  }


  /*geordnetes Programm-Ende mit 'q' und Strg+C*/
int fds; //Achtung:GlobaleVariable
void exitCode(int snum, siginfo_t *sinfo, void *nix){
  shutdown(fds,SHUT_RDWR);
  close(fds);
  printf("*** ->server terminated on signal %i.\n", snum);
  exit(EXIT_SUCCESS);
}
void *exit_on_Q(void*nix){
  while( NULL == strchr("QqCc\003",getchar()) );
  exitCode(-12345, NULL, NULL);
}
#define INSTALL_EXIT_FUNCTION() {\
    struct sigaction sa;        \
    sa.sa_handler       = NULL; \
    sa.sa_sigaction     = exitCode;     \
    sa.sa_flags         = SA_SIGINFO;   \
    sigemptyset(&sa.sa_mask);   \
    if(sigaction(SIGABRT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGALRM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGHUP , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGINT , &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGQUIT, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTERM, &sa, NULL)) perror("setting sigaction() failed.");    \
    if(sigaction(SIGTSTP, &sa, NULL)) perror("setting sigaction() failed.");    \
    pthread_t   THx;    \
    pthread_create( &THx, NULL, exit_on_Q, NULL);       \
    alarm(300); /*stop after 300[s]*/   \
  }
```

```
  //***************//
   //*** MAIN ***//
    //**********//
      //*****//
int main(int argc, char *argv[]){
  int           fdPORT, fdCOMM, r1=0, opt;
  char          rxdata[4096], outstr[4096], *s, hostName[256];
  struct sockaddr_in    s_address, c_address;

  DO_INITIAL_KREMPEL(1);

  START_100ms_TIMER();
  START_STATISTIK();
  MAKE_LISTEN_PORT( fdPORT, IpPort );
  INSTALL_EXIT_FUNCTION();

  for(;;){
    WAIT_FOR_REQUEST( fdCOMM, fdPORT, IpPort );
      /*Anfrage lesen:*/
    if(1 > (r1=read(fdCOMM, rxdata, sizeof(rxdata)-2 ))) break; //rxdata[r1]=0;
      /*antworten:*/
    snprintf( outstr, sizeof(outstr)-1,
        "HTTP/1.0 200 OK\r\n"
        "Server: RWS\r\n"
        "Content-Type: text/html\r\n\r\n"
        "<html><head><title>RWS - ein kleiner Webserver</title></head>\n"
        "<body bgcolor=tan><h2>NWES3a</h2><hr>\n"
        "<div align=center bgcolor=silver>"
        "<table border=1 align=center bgcolor=wheat"
        "  style=\"font:500 14pt monospace;\">\n"
        "<tr><td colspan=2 style='font:900 14pt serif;'>"
        "    100ms Timer Statistik timer_create():  \n"
        "<tr><td bgcolor=silver>sched.policy<td bgcolor=silver>%s /Prty:%d\n"
        "<tr><td bgcolor=silver>timerSlack<td bgcolor=silver>%d\n"
        "<tr><td bgcolor=silver>values<td bgcolor=silver>%ld\n"
        "<tr><td bgcolor=yellow>range <td bgcolor=yellow align=center>[%g, %g]\n"
        "<tr><td bgcolor=red   >&Delta;<td bgcolor=red align=center>%g (%g \%)\n"
        "<tr><td bgcolor=orange>average<td bgcolor=orange align=center>%g\n"
        "<tr><td bgcolor=aqua  >variance<td bgcolor=aqua   align=center>%g\n"
        "<tr><td colspan=2 align=center>frequencies:"
        "<tr><td bgcolor='#FFD0C0'>&lt; 90%% avg<td>%ld"
        "<tr><td bgcolor='#FFD0C0'>95..105 %% avg<td>%ld"
        "<tr><td bgcolor='#FFD0C0'>&gt;110 %% avg<td>%ld"
        "</table></div><hr>\n"
        "<h9>aus (c)2005 richard weinberger</h9>"
        "<script>"
        " window.setInterval('reloader()', 3000);"
        " function reloader(){"
        "   window.location.reload();"
        " }"
        "</script></body></html>\r\n\r\n",
        SCHEDULING_POLICY_AS_STRING( schedPol2 ),
        schedPrty2, tickTmrSlackVal,
        statsMem.valCnt,
        statsMem.minVal, statsMem.maxVal, statsMem.maxVal-statsMem.minVal,
        100.0*(statsMem.maxVal-statsMem.minVal)/statsMem.avgVal,
        statsMem.avgVal, statsMem.varVal,
        statsMem.fq10Minus, statsMem.fq5Avg, statsMem.fq10Plus
        );
    r1=write(fdCOMM, outstr, strlen(outstr) );
    close(fdCOMM);
  }
  shutdown(fdPORT,SHUT_RDWR);
  close(fdPORT);
  printf("\nserver \"%s\" terminated.\n",*argv);

  KILL_Timer();
  printf("done.\n");
}
```

Single tasking OS erlauben keine Paralleltätigkeiten und damit keine unmittelbare Reaktion auf (Input-) Ereignisse. Man müsste ausschliesslich *'Polling'* incl. Priorisierungen in der Anwendung programmieren — das bedeutet aber nichts anderes, als ein *Taskmanagement selber nachzubauen*. Den Entwicklungsaufwand für ein Taskmanagement kann man sich sparen, wenn das OS schon einen (wesentlich besser erprobten) Taskmanager bereitstellt.

Single user OS sind auch uninteressant, weil (nicht mehr zeitgemäß und) man Zugriffsrechte verwenden möchte (was ohne 'user' nicht geht).

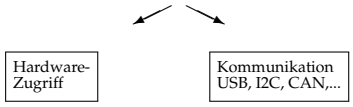
Multitasking, Taskmanager Ohne Multitasking/ Taskmanager (Scheduler) gibt es kein Scheduling, d.h. keine Parallelthreads, keine OS-Timer, keine Prioritäten, keine RT-Prozesse, kein timed waitig usw.

VMM
Da die Programmiersprachencompiler (ie. die Linker) jedes Programm (ausser bei spez. Linkerparametern) so generieren, als wären sie die einzige Software im Programm- und Datenspeicher, ist eine MMU mit damit verbundener VMM-Relocation unabdingbar (es ürden sich sonst schon OS und einzige Application gegenseitig überschreiben)

File System
Ohne VFS (virtual file system) gibt es gar keine Dateizugriffe auch nicht auf **Pseudo- und Devicefiles**, also auch kein *embedded* Hardwarezugriff (auf UNIXoiden Systemen unvorstellbar).

Multi User + Zugriffsrechte
Ein Single User System unterscheidet keine User, geht nicht von von mehrfachen Usern aus, auch nicht zeitlich hintereinander und kennt daher keine User-Schutzmechanismen, dh. gar keine Zugriffsregelungen (wie zB. bei μC Standalone-Systemen

Hardwarezugriff / -abstraktion
Ein ES muss natürlich auf die umgebende Hardware — und damit iA. auf die eigene — zugreifen können!



Networking
Zuminset in IoT Umgebungen will man Kommandos, Messwerte, Status- und Notmeldungen von und nach 'aussen' intensiv einsetzen. Zurzeit hat man dazu vor allem LAN und WLAN im Auge.

Jedoch ist die OS- Netzwerkkomponente nicht nur für die tatsächliche Datenübertragung erforderlich. Die Kommunikation von Prozessen untereinander wird gern über Netzwerksockets auf 'eigenen' Ports, ie. auf 'localhost'/'127.0.0.1', implementiert — es hat etliche Vorteile:

- jederzeit auf andere Rechner umleiten
- 'Firewalls' einschleusen
- mitprotokollieren
- durch Testumgebungen simulieren
- Zugriffsrechte und -Regelung
- Prozesskommunikation chiffrieren

usw.

Security, Funktionssicherheit
'Security' ist

- einerseit der Schutz vor Ungebetenen.
  - Geheimhaltung
  - Fälschung v. Kommandos u. Messwerten
- anderseits die 'Funktionssichertheit'
  - Ausfall
  - falsche Funktion
  - Notsituationen

Diese Aspekte sind Software-Angelegenheiten und erfordern daher keine speziellen OS-Komponenten

LogFiles
Um Intrudern die Möglichkeit der Verwischung ihrer Spuren auch in Logfiles zu nehmen, implementiert man Logschreibungen via Kommunikationsverbindung auf entfernte Anlagen, die idealerweise sonst keine Zugänge aufweisen.

Datenbanken
Da ein OS schon für sich selbst Datenbanken braucht (Userlisten, USB-Device-Kataloge, Spooling, Konfigurationen (zB 'Registry') ...), muss auch ein RTOS ein Mindestmaß an DB Funktionalität enthalten.

Services? (Client/Server)
Dienste für die Allgemeinheit sind im Embdedded System und sein OS eher nicht vorzusehen — mit Einschränkung, Authentifizierung und Kryptografie aber durchaus zweckdienlich, zB. für IoT Implementation.
Da Multitasking (zur Hintergrundausführung) und Netzwerkerei aus anderen Gründen (s.o.) schon ins OS übernommen werden, ist für Serverprogramme bereits alles drin.

UI: CLI? GUI?

Man erkennt: Bis auf UI braucht sich das alles mehr oder weniger gegenseitig, sodass man von einem Kernel fallweise nur einige spezielle Peripherikomponenten weg lassen kann, wie etwa 'RAID', 'Bluetooth' oder 'IPX/SPX'

40.9 Echtzeit (real time RT) ↔ Simulationszeit

Echtzeit:

Fortschritt im Programmcode	↓ ↓ ↓	↓ Fortschritt ↓ in der ↓ Realität

↔

zB.:
'digitalWrite(AmpelRed,1)' ⟷ Ampel wird *zeitgleich* rot

Simulationszeit:

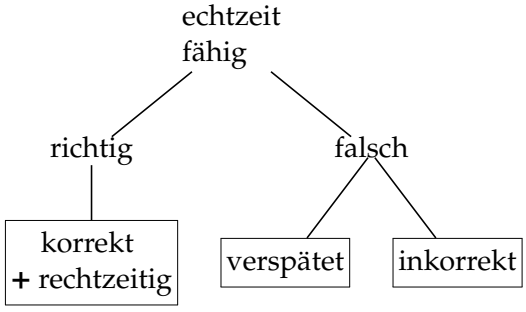
zB. • Chemie-
• Physik-
• Elektronikschaltungssimulation:

5ns simulierte Zeit ≡ 20s Rechenzeit

zB. Computer Games:

5 Jahre gespielter Krieg ≡ 1h Spielzeit

40.10 'echtzeitfähig':



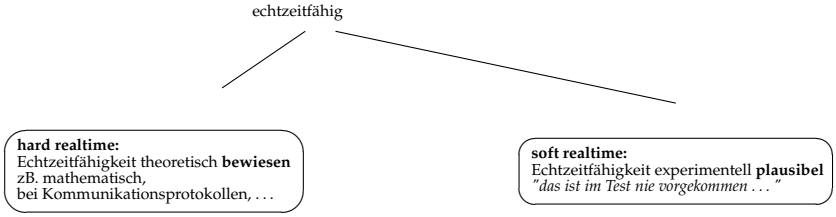
echtzeitfähige Systeme liefern
(1) das **richtige** Ergebnis binnen vorgegebenen **Zeitlimits** –
(2) **verspätete** Ergebnisse gelten als **falsche** Ergebnisse. (und werden ignoriert)

echtzeitfähige Kommunikation
(1) bringt Nachrichten binnen vorgegebenen **Zeitlimits** ans Ziel –
(2) **verspätete** Nachrichten gelten als **fehlerhaft** übertragene Nachrichten (und werden verworfen)

Echzeitbetrieb ≡ wie realisiert man es

- IO-Signale
- Interrupts
- Timersteuerung
- Parallelprozesse

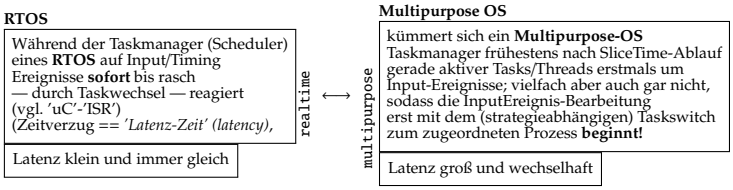
hard- und soft realtime:



40.11 embedded RTOS

40.11.1 Kennzeichen

Die *realtime*-Eigenschaften kompatibler OS äussern sich in präziserem Timing und Scheduling:



Neuere UNIXoide Systeme haben die RT-Funktionen grundsätzlich eingebaut, vollführen sie aber unterschiedlich exakt. Das hat den Vorteil, auch RT-Software auf multipurpose- Desktoprechnern entwickeln zu können; erst für *präzises* Timing ist das RT-Zielsystem erforderlich (zB. ein RTlinux)

40.11.2 Marktschreierei

Allzu gern wird jeder *Mistfurz* propagandahalber **RTOS** genannt, nu weilma vielleicht mal was einlesen oder ausgeben kann; auch gibts uralten Kas aus der (vor-) M$DOS-Zeit mit ISRs statt an gscheitn Scheduling.

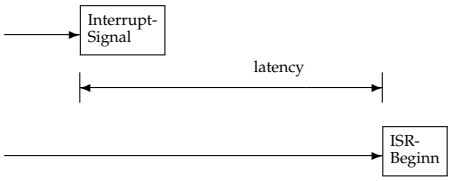
(Es wurde sogar versucht, mir eine kleine Arduino- dazulink- Library als 'RTOS' zu 'verkaufen', obwohl es sich dabei ja ganz offensichtlich nicht einmal um ein OS, geschweige denn ein RTOS handeln kann!)

Viele sich 'realtime operating system' nennenden Unterprogrammsammlungen verfügen über sehr eingeschränkte, allgemeine OS-Funktionalität, teilweise nicht einmal ein VMM, kaum Kommunikation, Device-Erkennung (zB nix USB), User-Interface und laufen nur auf ganz spezieller Hardware weniger Hersteller.

Mit einem *richtigen* OS-Mainline-Kernel hat man immerhin schon VMM, Scheduling, User-/Zugriffsrechte, Filesysteme, meistens auch Entwicklungswerkzeuge (development tools) wie Sprachencompiler u. Debugger und viel Kommunikation.

40.11.3 Zeitverhalten: Latenz

'Latenz' ('latency') ist die Verzugszeit der Reaktion auf die Anforderung, zB. die Zeit von einer *'Interrupt'*-Signalisierung bis zum Beginn der Bearbeitung.



Diese Reaktions-Verzugszeit ist ein maßgebliches Qualitätskriterium eines RTOS und wie Antwortzeiten *'mit der Stopuhr'* messbar (natürlich muss man solche idealerweise Nanosekundenbeträge elektronisch erfassen; mit etwas *'data acquisition'* zur Triggerung (und geeigneten Tastköpfen!) geht das schon mit einfachen Oszilloskopen) (s. Jesus Screenshot)

40.12 RTOS Konfiguration

- Treiber installieren
- Programme/Pakete installieren
- User einrichten
- Zugriffsrechte anpassen
- Files/Directories anlegem (zB. Logfiles)
- Konfigurationsdateien editieren
- Service-Starts konfigurieren (zB. via cron oder Systemstart)

40.13 'systemd' system + service manager

systemctl isolate multi-user.target(or) systemctl isolate runlevel3.target
systemctl isolate graphical.target (or) systemctl isolate runlevel5.target

40.13.1 'man 1 systemd' man page

Listing 74: man 1 systemd

```
SYSTEMD(1)              systemd                 SYSTEMD(1)
NAME
       systemd, init - systemd system and service manager

SYNOPSIS
       systemd [OPTIONS...]

       init [OPTIONS...] {COMMAND}

DESCRIPTION
       systemd is a system and service manager for Linux operating systems. When run as first
           process
       on boot (as PID 1), it acts as init system that brings up and maintains userspace
           services.

       For compatibility with SysV, if systemd is called as init and a PID that is not 1, it
           will
       execute telinit and pass all command line arguments unmodified. That means init and
           telinit are
       mostly equivalent when invoked from normal login sessions. See telinit(8) for more
           information.

       When run as system instance, systemd interprets the configuration file system.conf,
           otherwise
       user.conf. See systemd-system.conf(5) for more information.

OPTIONS
```

```
       The following options are understood:

       --test
           Determine startup sequence, dump it and exit. This is an option useful for debugging
                only.

       --dump-configuration-items
           Dump understood unit configuration items. This outputs a terse but complete list of
           configuration items understood in unit definition files.

       --unit=
           Set default unit to activate on startup. If not specified, defaults to default.
               target.

       --system, --user
           For --system, tell systemd to run a system instance, even if the process ID is not
               1, i.e.
           systemd is not run as init process.  --user does the opposite, running a user
               instance even
           if the process ID is 1. Normally it should not be necessary to pass these options,
               as
           systemd automatically detects the mode it is started in. These options are hence of
               little
           use except for debugging. Note that it is not supported booting and maintaining a
               full
           system with systemd running in --system mode, but PID not 1. In practice, passing --
               system
           explicitly is only useful in conjunction with --test.

       --dump-core
           Dump core on crash. This switch has no effect when run as user instance.

       --crash-shell
           Run shell on crash. This switch has no effect when run as user instance.

       --confirm-spawn
           Ask for confirmation when spawning processes. This switch has no effect when run as
               user
           instance.

       --show-status=
           Show terse service status information while booting. This switch has no effect when
               run as
           user instance. Takes a boolean argument which may be omitted which is interpreted as
                true.

       --log-target=
           Set log target. Argument must be one of console, journal, syslog, kmsg, journal-or-
               kmsg,
           syslog-or-kmsg, null.

       --log-level=
           Set log level. As argument this accepts a numerical log level or the well-known
               syslog(3)
           symbolic names (lowercase): emerg, alert, crit, err, warning, notice, info, debug.

       --log-color=
           Highlight important log messages. Argument is a boolean value. If the argument is
               omitted,
           it defaults to true.

       --log-location=
           Include code location in log messages. This is mostly relevant for debugging
               purposes.
           Argument is a boolean value. If the argument is omitted it defaults to true.

       --default-standard-output=, --default-standard-error=
           Sets the default output or error output for all services and sockets, respectively.
               That
           is, controls the default for StandardOutput= and StandardError= (see systemd.exec(5)
                for
           details). Takes one of inherit, null, tty, journal, journal+console, syslog,
           syslog+console, kmsg, kmsg+console. If the argument is omitted --default-standard-
```

```
               output=
           defaults to journal and --default-standard-error= to inherit.

       -h, --help
           Print a short help text and exit.

       --version
           Print a short version string and exit.

CONCEPTS
       systemd provides a dependency system between various entities called "units" of 12
           different
       types. Units encapsulate various objects that are relevant for system boot-up and
           maintenance.
       The majority of units are configured in unit configuration files, whose syntax and basic
            set of
       options is described in systemd.unit(5), however some are created automatically from
           other
       configuration, dynamically from system state or programmatically at runtime. Units may
           be
       "active" (meaning started, bound, plugged in, ..., depending on the unit type, see below
           ), or
       "inactive" (meaning stopped, unbound, unplugged, ...), as well as in the process of
           being
       activated or deactivated, i.e. between the two states (these states are called "
           activating",
       "deactivating"). A special "failed" state is available as well, which is very similar to
       "inactive" and is entered when the service failed in some way (process returned error
           code on
       exit, or crashed, or an operation timed out). If this state is entered, the cause will
           be
       logged, for later reference. Note that the various unit types may have a number of
           additional
       substates, which are mapped to the five generalized unit states described here.

       The following unit types are available:

        1. Service units, which start and control daemons and the processes they consist of.
            For
           details see systemd.service(5).

        2. Socket units, which encapsulate local IPC or network sockets in the system, useful
            for
           socket-based activation. For details about socket units see systemd.socket(5), for
               details
           on socket-based activation and other forms of activation, see daemon(7).

        3. Target units are useful to group units, or provide well-known synchronization points
             during
           boot-up, see systemd.target(5).

        4. Device units expose kernel devices in systemd and may be used to implement device-
            based
           activation. For details see systemd.device(5).

        5. Mount units control mount points in the file system, for details see systemd.mount
            (5).

        6. Automount units provide automount capabilities, for on-demand mounting of file
            systems as
           well as parallelized boot-up. See systemd.automount(5).

        7. Snapshot units can be used to temporarily save the state of the set of systemd units
            , which
           later may be restored by activating the saved snapshot unit. For more information
               see
           systemd.snapshot(5).

        8. Timer units are useful for triggering activation of other units based on timers. You
             may
           find details in systemd.timer(5).

        9. Swap units are very similar to mount units and encapsulate memory swap partitions or
```

```
       files
    of the operating system. They are described in systemd.swap(5).

10. Path units may be used to activate other services when file system objects change or
     are
    modified. See systemd.path(5).

11. Slice units may be used to group units which manage system processes (such as
    service and
    scope units) in a hierarchical tree for resource management purposes. See systemd.
        slice(5).

12. Scope units are similar to service units, but manage foreign processes instead of
    starting
    them as well. See systemd.scope(5).

Units are named as their configuration files. Some units have special semantics. A
    detailed
list is available in systemd.special(7).

systemd knows various kinds of dependencies, including positive and negative requirement
dependencies (i.e.  Requires= and Conflicts=) as well as ordering dependencies (After=
    and
Before=). NB: ordering and requirement dependencies are orthogonal. If only a
    requirement
dependency exists between two units (e.g.  foo.service requires bar.service), but no
    ordering
dependency (e.g.  foo.service after bar.service) and both are requested to start, they
    will be
started in parallel. It is a common pattern that both requirement and ordering
    dependencies are
placed between two units. Also note that the majority of dependencies are implicitly
    created
and maintained by systemd. In most cases, it should be unnecessary to declare additional
dependencies manually, however it is possible to do this.

Application programs and units (via dependencies) may request state changes of units. In
systemd, these requests are encapsulated as 'jobs' and maintained in a job queue. Jobs
    may
succeed or can fail, their execution is ordered based on the ordering dependencies of
    the units
they have been scheduled for.

On boot systemd activates the target unit default.target whose job is to activate on-
    boot
services and other on-boot units by pulling them in via dependencies. Usually the unit
    name is
just an alias (symlink) for either graphical.target (for fully-featured boots into the
    UI) or
multi-user.target (for limited console-only boots for use in embedded or server
    environments,
or similar; a subset of graphical.target). However, it is at the discretion of the
administrator to configure it as an alias to any other target unit. See systemd.special
    (7) for
details about these target units.

Processes systemd spawns are placed in individual Linux control groups named after the
    unit
which they belong to in the private systemd hierarchy. (see cgroups.txt[1] for more
    information
about control groups, or short "cgroups"). systemd uses this to effectively keep track
    of
processes. Control group information is maintained in the kernel, and is accessible via
    the
file system hierarchy (beneath /sys/fs/cgroup/systemd/), or in tools such as ps(1) (ps
    xawf -eo
pid,user,cgroup,args is particularly useful to list all processes and the systemd units
    they
belong to.).

systemd is compatible with the SysV init system to a large degree: SysV init scripts are
supported and simply read as an alternative (though limited) configuration file format.
    The
```

```
       SysV /dev/initctl interface is provided, and compatibility implementations of the
           various SysV
       client tools are available. In addition to that, various established Unix functionality
           such as
       /etc/fstab or the utmp database are supported.

       systemd has a minimal transaction system: if a unit is requested to start up or shut
           down it
       will add it and all its dependencies to a temporary transaction. Then, it will verify if
            the
       transaction is consistent (i.e. whether the ordering of all units is cycle-free). If it
           is not,
       systemd will try to fix it up, and removes non-essential jobs from the transaction that
           might
       remove the loop. Also, systemd tries to suppress non-essential jobs in the transaction
           that
       would stop a running service. Finally it is checked whether the jobs of the transaction
       contradict jobs that have already been queued, and optionally the transaction is aborted
            then.
       If all worked out and the transaction is consistent and minimized in its impact it is
           merged
       with all already outstanding jobs and added to the run queue. Effectively this means
           that
       before executing a requested operation, systemd will verify that it makes sense, fixing
           it if
       possible, and only failing if it really cannot work.

       Systemd contains native implementations of various tasks that need to be executed as
           part of
       the boot process. For example, it sets the hostname or configures the loopback network
           device.
       It also sets up and mounts various API file systems, such as /sys or /proc.

       For more information about the concepts and ideas behind systemd, please refer to the
           Original
       Design Document[2].

       Note that some but not all interfaces provided by systemd are covered by the Interface
       Stability Promise[3].

       Units may be generated dynamically at boot and system manager reload time, for example
           based on
       other configuration files or parameters passed on the kernel command line. For details
           see the
       Generators Specification[4].

       Systems which invoke systemd in a container or initrd environment should implement the
       Container Interface[5] or initrd Interface[6] specifications, respectively.

DIRECTORIES
       System unit directories
           The systemd system manager reads unit configuration from various directories.
               Packages that
           want to install unit files shall place them in the directory returned by pkg-config
               systemd
           --variable=systemdsystemunitdir. Other directories checked are
           /usr/local/lib/systemd/system and /lib/systemd/system. User configuration always
               takes
           precedence.  pkg-config systemd --variable=systemdsystemconfdir returns the path of
               the
           system configuration directory. Packages should alter the content of these
               directories only
           with the enable and disable commands of the systemctl(1) tool. Full list of
               directories is
           provided in systemd.unit(5).

       User unit directories
           Similar rules apply for the user unit directories. However, here the XDG Base
               Directory
           specification[7] is followed to find units. Applications should place their unit
               files in
           the directory returned by pkg-config systemd --variable=systemduserunitdir. Global
           configuration is done in the directory reported by pkg-config systemd
```

```
            --variable=systemduserconfdir. The enable and disable commands of the systemctl(1)
                tool can
            handle both global (i.e. for all users) and private (for one user) enabling/
                disabling of
            units. Full list of directories is provided in systemd.unit(5).

        SysV init scripts directory
            The location of the SysV init script directory varies between distributions. If
                systemd
            cannot find a native unit file for a requested service, it will look for a SysV init
                 script
            of the same name (with the .service suffix removed).

        SysV runlevel link farm directory
            The location of the SysV runlevel link farm directory varies between distributions.
                systemd
            will take the link farm into account when figuring out whether a service shall be
                enabled.
            Note that a service unit with a native unit configuration file cannot be started by
            activating it in the SysV runlevel link farm.

SIGNALS
        SIGTERM
            Upon receiving this signal the systemd system manager serializes its state,
                reexecutes
            itself and deserializes the saved state again. This is mostly equivalent to
                systemctl
            daemon-reexec.

            systemd user managers will start the exit.target unit when this signal is received.
                This is
            mostly equivalent to systemctl --user start exit.target.

        SIGINT
            Upon receiving this signal the systemd system manager will start the ctrl-alt-del.
                target
            unit. This is mostly equivalent to systemctl start ctl-alt-del.target.

            systemd user managers treat this signal the same way as SIGTERM.

        SIGWINCH
            When this signal is received the systemd system manager will start the kbrequest.
                target
            unit. This is mostly equivalent to systemctl start kbrequest.target.

            This signal is ignored by systemd user managers.

        SIGPWR
            When this signal is received the systemd manager will start the sigpwr.target unit.
                This is
            mostly equivalent to systemctl start sigpwr.target.

        SIGUSR1
            When this signal is received the systemd manager will try to reconnect to the D-Bus
                bus.

        SIGUSR2
            When this signal is received the systemd manager will log its complete state in
                human
            readable form. The data logged is the same as printed by systemctl dump.

        SIGHUP
            Reloads the complete daemon configuration. This is mostly equivalent to systemctl
            daemon-reload.

        SIGRTMIN+0
            Enters default mode, starts the default.target unit. This is mostly equivalent to
                systemctl
            start default.target.

        SIGRTMIN+1
            Enters rescue mode, starts the rescue.target unit. This is mostly equivalent to
                systemctl
```

```
           isolate rescue.target.

       SIGRTMIN+2
           Enters emergency mode, starts the emergency.service unit. This is mostly equivalent
               to
           systemctl isolate emergency.service.

       SIGRTMIN+3
           Halts the machine, starts the halt.target unit. This is mostly equivalent to
               systemctl
           start halt.target.

       SIGRTMIN+4
           Powers off the machine, starts the poweroff.target unit. This is mostly equivalent
               to
           systemctl start poweroff.target.

       SIGRTMIN+5
           Reboots the machine, starts the reboot.target unit. This is mostly equivalent to
               systemctl
           start reboot.target.

       SIGRTMIN+6
           Reboots the machine via kexec, starts the kexec.target unit. This is mostly
               equivalent to
           systemctl start kexec.target.

       SIGRTMIN+13
           Immediately halts the machine.

       SIGRTMIN+14
           Immediately powers off the machine.

       SIGRTMIN+15
           Immediately reboots the machine.

       SIGRTMIN+16
           Immediately reboots the machine with kexec.

       SIGRTMIN+20
           Enables display of status messages on the console, as controlled via systemd.
               show_status=1
           on the kernel command line.

       SIGRTMIN+21
           Disables display of status messages on the console, as controlled via systemd.
               show_status=0
           on the kernel command line.

       SIGRTMIN+22, SIGRTMIN+23
           Sets the log level to "debug" (or "info" on SIGRTMIN+23), as controlled via
           systemd.log_level=debug (or systemd.log_level=info on SIGRTMIN+23) on the kernel
               command
           line.

       SIGRTMIN+24
           Immediately exits the manager (only available for --user instances).

       SIGRTMIN+26, SIGRTMIN+27, SIGRTMIN+28, SIGRTMIN+29
           Sets the log level to "journal-or-kmsg" (or "console" on SIGRTMIN+27, "kmsg" on
           SIGRTMIN+28, or "syslog-or-kmsg" on SIGRTMIN+29), as controlled via
           systemd.log_target=journal-or-kmsg (or systemd.log_target=console on SIGRTMIN+27,
           systemd.log_target=kmsg on SIGRTMIN+28, or systemd.log_target=syslog-or-kmsg on
           SIGRTMIN+29) on the kernel command line.

ENVIRONMENT
       $SYSTEMD_LOG_LEVEL
           systemd reads the log level from this environment variable. This can be overridden
               with
           --log-level=.

       $SYSTEMD_LOG_TARGET
           systemd reads the log target from this environment variable. This can be overridden
```

```
               with
           --log-target=.

       $SYSTEMD_LOG_COLOR
           Controls whether systemd highlights important log messages. This can be overridden
               with
           --log-color=.

       $SYSTEMD_LOG_LOCATION
           Controls whether systemd prints the code location along with log messages. This can
               be
           overridden with --log-location=.

       $XDG_CONFIG_HOME, $XDG_CONFIG_DIRS, $XDG_DATA_HOME, $XDG_DATA_DIRS
           The systemd user manager uses these variables in accordance to the XDG Base
               Directory
           specification[7] to find its configuration.

       $SYSTEMD_UNIT_PATH
           Controls where systemd looks for unit files.

       $SYSTEMD_SYSVINIT_PATH
           Controls where systemd looks for SysV init scripts.

       $SYSTEMD_SYSVRCND_PATH
           Controls where systemd looks for SysV init script runlevel link farms.

       $LISTEN_PID, $LISTEN_FDS
           Set by systemd for supervised processes during socket-based activation. See
           sd_listen_fds(3) for more information.

       $NOTIFY_SOCKET
           Set by systemd for supervised processes for status and start-up completion
               notification.
           See sd_notify(3) for more information.

KERNEL COMMAND LINE
       When run as system instance systemd parses a number of kernel command line arguments[8]:

       systemd.unit=, rd.systemd.unit=
           Overrides the unit to activate on boot. Defaults to default.target. This may be used
                to
           temporarily boot into a different boot unit, for example rescue.target or
           emergency.service. See systemd.special(7) for details about these units. The option
           prefixed with "rd."  is honored only in the initial RAM disk (initrd), while the one
                that
           is not prefixed only in the main system.

       systemd.dump_core=
           Takes a boolean argument. If true, systemd dumps core when it crashes. Otherwise, no
                core
           dump is created. Defaults to true.

       systemd.crash_shell=
           Takes a boolean argument. If true, systemd spawns a shell when it crashes. Otherwise
               , no
           shell is spawned. Defaults to false, for security reasons, as the shell is not
               protected by
           any password authentication.

       systemd.crash_chvt=
           Takes an integer argument. If positive systemd activates the specified virtual
               terminal
           when it crashes. Defaults to -1.

       systemd.confirm_spawn=
           Takes a boolean argument. If true, asks for confirmation when spawning processes.
               Defaults
           to false.

       systemd.show_status=
           Takes a boolean argument or the constant auto. If true, shows terse service status
               updates
```

```
            on the console during bootup.  auto behaves like false until a service fails or
                there is a
            significant delay in boot. Defaults to true, unless quiet is passed as kernel
                command line
            option in which case it defaults to auto.

        systemd.log_target=, systemd.log_level=, systemd.log_color=, systemd.log_location=
            Controls log output, with the same effect as the $SYSTEMD_LOG_TARGET,
                $SYSTEMD_LOG_LEVEL,
            $SYSTEMD_LOG_COLOR, $SYSTEMD_LOG_LOCATION environment variables described above.

        systemd.default_standard_output=, systemd.default_standard_error=
            Controls default standard output and error output for services, with the same effect
                 as the
            --default-standard-output= and --default-standard-error= command line arguments
                described
            above, respectively.

        systemd.setenv=
            Takes a string argument in the form VARIABLE=VALUE. May be used to set default
                environment
            variables to add to forked child processes. May be used more than once to set
                multiple
            variables.

        quiet
            Turn off status output at boot, much like systemd.show_status=false would. Note that
                 this
            option is also read by the kernel itself and disables kernel log output. Passing
                this
            option hence turns off the usual output from both the system manager and the kernel.

        debug
            Turn on debugging output. This is equivalent to systemd.log_level=debug. Note that
                this
            option is also read by the kernel itself and enables kernel debug output. Passing
                this
            option hence turns on the debug output from both the system manager and the kernel.

        -b, emergency
            Boot into emergency mode. This is equivalent to systemd.unit=emergency.target and
                provided
            for compatibility reasons and to be easier to type.

        single, s, S, 1
            Boot into rescue mode. This is equivalent to systemd.unit=rescue.target and provided
                 for
            compatibility reasons and to be easier to type.

        2, 3, 4, 5
            Boot into the specified legacy SysV runlevel. These are equivalent to
            systemd.unit=runlevel2.target, systemd.unit=runlevel3.target,
            systemd.unit=runlevel4.target, and systemd.unit=runlevel5.target, respectively, and
            provided for compatibility reasons and to be easier to type.

        locale.LANG=, locale.LANGUAGE=, locale.LC_CTYPE=, locale.LC_NUMERIC=, locale.LC_TIME=,
        locale.LC_COLLATE=, locale.LC_MONETARY=, locale.LC_MESSAGES=, locale.LC_PAPER=,
        locale.LC_NAME=, locale.LC_ADDRESS=, locale.LC_TELEPHONE=, locale.LC_MEASUREMENT=,
        locale.LC_IDENTIFICATION=
            Set the system locale to use. This overrides the settings in /etc/locale.conf. For
                more
            information see locale.conf(5) and locale(7).

        For other kernel command line parameters understood by components of the core OS, please
             refer
        to kernel-command-line(7).

SOCKETS AND FIFOS
        /run/systemd/notify
            Daemon status notification socket. This is an AF_UNIX datagram socket and is used to
            implement the daemon notification logic as implemented by sd_notify(3).

        /run/systemd/shutdownd
```

```
           Used internally by the shutdown(8) tool to implement delayed shutdowns. This is an
               AF_UNIX
           datagram socket.

       /run/systemd/private
           Used internally as communication channel between systemctl(1) and the systemd
               process. This
           is an AF_UNIX stream socket. This interface is private to systemd and should not be
               used in
           external projects.

       /dev/initctl
           Limited compatibility support for the SysV client interface, as implemented by the
           systemd-initctl.service unit. This is a named pipe in the file system. This
               interface is
           obsolete and should not be used in new applications.

SEE ALSO
       The systemd Homepage[9], systemd-system.conf(5), locale.conf(5), systemctl(1),
           journalctl(1),
       systemd-notify(1), daemon(7), sd-daemon(3), systemd.unit(5), systemd.special(5), pkg-
           config(1),
       kernel-command-line(7), bootup(7), systemd.directives(7)

NOTES
        1. cgroups.txt
           https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt

        2. Original Design Document
           http://0pointer.de/blog/projects/systemd.html

        3. Interface Stability Promise
           http://www.freedesktop.org/wiki/Software/systemd/InterfaceStabilityPromise

        4. Generators Specification
           http://www.freedesktop.org/wiki/Software/systemd/Generators

        5. Container Interface
           http://www.freedesktop.org/wiki/Software/systemd/ContainerInterface

        6. initrd Interface
           http://www.freedesktop.org/wiki/Software/systemd/InitrdInterface

        7. XDG Base Directory specification
           http://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html

        8. If run inside a Linux container these arguments may be passed as command line
            arguments to
           systemd itself, next to any of the command line options listed in the Options
               section
           above. If run outside of Linux containers, these arguments are parsed from /proc/
               cmdline
           instead.

        9. systemd Homepage
           http://www.freedesktop.org/wiki/Software/systemd/

systemd 215
    SYSTEMD(1)
```

40.13.2 'man 1 systemctl man page

Listing 75: man 1 systemctl

```
SYSTEMCTL(1)            systemctl                        SYSTEMCTL(1)
NAME
       systemctl - Control the systemd system and service manager

SYNOPSIS
       systemctl [OPTIONS...] COMMAND [NAME...]

DESCRIPTION
```

```
       systemctl may be used to introspect and control the state of the systemd(1) system and
           service
       manager.

OPTIONS
       The following options are understood:

       -t, --type=
           The argument should be a comma-separated list of unit types such as service and
               socket.

           If one of the arguments is a unit type, when listing units, limit display to certain
                unit
           types. Otherwise, units of all types will be shown.

           As a special case, if one of the arguments is help, a list of allowed values will be
           printed and the program will exit.

       --state=
           The argument should be a comma-separated list of unit LOAD, SUB, or ACTIVE states.
               When
           listing units, show only those in specified states.

       -p, --property=
           When showing unit/job/manager properties with the show command, limit display to
               certain
           properties as specified as argument. If not specified, all set properties are shown.
                The
           argument should be a comma-separated list of property names, such as "MainPID". If
           specified more than once, all properties with the specified names are shown.

       -a, --all
           When listing units, show all loaded units, regardless of their state, including
               inactive
           units. When showing unit/job/manager properties, show all properties regardless
               whether
           they are set or not.

           To list all units installed on the system, use the list-unit-files command instead.

       -r, --recursive
           When listing units, also show units of local containers. Units of local containers
               will be
           prefixed with the container name, separated by a single colon character (":").

       --reverse
           Show reverse dependencies between units with list-dependencies, i.e. units with
           dependencies of type Wants= or Requires= on the given unit.

       --after
           With list-dependencies, show the units that are ordered before the specified unit.
               In other
           words, list the units that are in the After= directive of the specified unit, have
               the
           specified unit in their Before= directive, or are otherwise implicit dependencies of
                the
           specified unit.

       --before
           With list-dependencies, show the units that are ordered after the specified unit. In
                other
           words, list the units that are in the Before= directive of the specified unit, have
               the
           specified unit in their After= directive, or otherwise depend on the specified unit.

       -l, --full
           Do not ellipsize unit names, process tree entries, journal output, or truncate unit
           descriptions in the output of status, list-units, list-jobs, and list-timers.

       --show-types
           When showing sockets, show the type of the socket.

       --job-mode=
```

```
        When queuing a new job, this option controls how to deal with already queued jobs.
            It takes
        one of "fail", "replace", "replace-irreversibly", "isolate", "ignore-dependencies",
        "ignore-requirements" or "flush". Defaults to "replace", except when the isolate
            command is
        used which implies the "isolate" job mode.

        If "fail" is specified and a requested operation conflicts with a pending job (more
        specifically: causes an already pending start job to be reversed into a stop job or
            vice
        versa), cause the operation to fail.

        If "replace" (the default) is specified, any conflicting pending job will be
            replaced, as
        necessary.

        If "replace-irreversibly" is specified, operate like "replace", but also mark the
            new jobs
        as irreversible. This prevents future conflicting transactions from replacing these
            jobs
        (or even being enqueued while the irreversible jobs are still pending). Irreversible
             jobs
        can still be cancelled using the cancel command.

        "isolate" is only valid for start operations and causes all other units to be
            stopped when
        the specified unit is started. This mode is always used when the isolate command is
            used.

        "flush" will cause all queued jobs to be canceled when the new job is enqueued.

        If "ignore-dependencies" is specified, then all unit dependencies are ignored for
            this new
        job and the operation is executed immediately. If passed, no required units of the
            unit
        passed will be pulled in, and no ordering dependencies will be honored. This is
            mostly a
        debugging and rescue tool for the administrator and should not be used by
            applications.

        "ignore-requirements" is similar to "ignore-dependencies", but only causes the
            requirement
        dependencies to be ignored, the ordering dependencies will still be honoured.

    -i, --ignore-inhibitors
        When system shutdown or a sleep state is requested, ignore inhibitor locks.
            Applications
        can establish inhibitor locks to avoid that certain important operations (such as CD
        burning or suchlike) are interrupted by system shutdown or a sleep state. Any user
            may take
        these locks and privileged users may override these locks. If any locks are taken,
            shutdown
        and sleep state requests will normally fail (regardless of whether privileged or not
            ) and a
        list of active locks is printed. However, if --ignore-inhibitors is specified, the
            locks
        are ignored and not printed, and the operation attempted anyway, possibly requiring
        additional privileges.

    -q, --quiet
        Suppress output to standard output in snapshot, is-active, is-failed, is-enabled,
        is-system-running, enable and disable.

    --no-block
        Do not synchronously wait for the requested operation to finish. If this is not
            specified,
        the job will be verified, enqueued and systemctl will wait until it is completed. By
        passing this argument, it is only verified and enqueued.

    --no-legend
        Do not print the legend, i.e. the column headers and the footer with hints.

    --user
```

```
        Talk to the service manager of the calling user, rather than the service manager of
            the
        system.

    --system
        Talk to the service manager of the system. This is the implied default.

    --failed
        List units in failed state. This is equivalent to --state=failed.

    --no-wall
        Do not send wall message before halt, power-off, reboot.

    --global
        When used with enable and disable, operate on the global user configuration
            directory, thus
        enabling or disabling a unit file globally for all future logins of all users.

    --no-reload
        When used with enable and disable, do not implicitly reload daemon configuration
            after
        executing the changes.

    --no-ask-password
        When used with start and related commands, disables asking for passwords. Background
        services may require input of a password or passphrase string, for example to unlock
             system
        hard disks or cryptographic certificates. Unless this option is specified and the
            command
        is invoked from a terminal, systemctl will query the user on the terminal for the
            necessary
        secrets. Use this option to switch this behavior off. In this case, the password
            must be
        supplied by some other means (for example graphical password agents) or the service
            might
        fail. This also disables querying the user for authentication for privileged
            operations.

    --kill-who=
        When used with kill, choose which processes to send a signal to. Must be one of main
            ,
        control or all to select whether to kill only the main process, the control process
            or all
        processes of the unit. The main process of the unit is the one that defines the life
            -time
        of it. A control process of a unit is one that is invoked by the manager to induce
            state
        changes of it. For example, all processes started due to the ExecStartPre=, ExecStop
            = or
        ExecReload= settings of service units are control processes. Note that there is only
             one
        control process per unit at a time, as only one state change is executed at a time.
            For
        services of type Type=forking, the initial process started by the manager for
            ExecStart= is
        a control process, while the process ultimately forked off by that one is then
            considered
        the main process of the unit (if it can be determined). This is different for
            service units
        of other types, where the process forked off by the manager for ExecStart= is always
             the
        main process itself. A service unit consists of zero or one main process, zero or
            one
        control process plus any number of additional processes. Not all unit types manage
        processes of these types however. For example, for mount units, control processes
            are
        defined (which are the invocations of /bin/mount and /bin/umount), but no main
            process is
        defined. If omitted, defaults to all.

    -s, --signal=
        When used with kill, choose which signal to send to selected processes. Must be one
            of the
```

```
        well known signal specifiers such as SIGTERM, SIGINT or SIGSTOP. If omitted,
            defaults to
        SIGTERM.

    -f, --force
        When used with enable, overwrite any existing conflicting symlinks.

        When used with halt, poweroff, reboot or kexec, execute the selected operation
            without
        shutting down all units. However, all processes will be killed forcibly and all file
        systems are unmounted or remounted read-only. This is hence a drastic but relatively
             safe
        option to request an immediate reboot. If --force is specified twice for these
            operations,
        they will be executed immediately without terminating any processes or unmounting
            any file
        systems. Warning: specifying --force twice with any of these operations might result
             in
        data loss.

    --root=
        When used with enable/disable/is-enabled (and related commands), use alternative
            root path
        when looking for unit files.

    --runtime
        When used with enable, disable, (and related commands), make changes only
            temporarily, so
        that they are lost on the next reboot. This will have the effect that changes are
            not made
        in subdirectories of /etc but in /run, with identical immediate effects, however,
            since the
        latter is lost on reboot, the changes are lost too.

        Similarly, when used with set-property, make changes only temporarily, so that they
            are
        lost on the next reboot.

    --preset-mode=
        Takes one of "full" (the default), "enable-only", "disable-only". When used with the
             preset
        or preset-all commands, controls whether units shall be disabled and enabled
            according to
        the preset rules, or only enabled, or only disabled.

    -n, --lines=
        When used with status, controls the number of journal lines to show, counting from
            the most
        recent ones. Takes a positive integer argument. Defaults to 10.

    -o, --output=
        When used with status, controls the formatting of the journal entries that are shown
            . For
        the available choices, see journalctl(1). Defaults to "short".

    --plain
        When used with list-dependencies, the output is printed as a list instead of a tree.

    -H, --host=
        Execute the operation remotely. Specify a hostname, or a username and hostname
            separated by
        "@", to connect to. The hostname may optionally be suffixed by a container name,
            separated
        by ":", which connects directly to a specific container on the specified host. This
            will
        use SSH to talk to the remote machine manager instance. Container names may be
            enumerated
        with machinectl -H HOST.

    -M, --machine=
        Execute operation on a local container. Specify a container name to connect to.

    -h, --help
```

```
           Print a short help text and exit.

       --version
           Print a short version string and exit.

       --no-pager
           Do not pipe output into a pager.

COMMANDS
       The following commands are understood:

   Unit Commands
       list-units [PATTERN...]
           List known units (subject to limitations specified with -t). If one or more PATTERNs
                are
           specified, only units matching one of them are shown.

           This is the default command.

       list-sockets [PATTERN...]
           List socket units ordered by listening address. If one or more PATTERNs are
               specified, only
           socket units matching one of them are shown. Produces output similar to

               LISTEN           UNIT                         ACTIVATES
               /dev/initctl     systemd-initctl.socket       systemd-initctl.service
               ...
               [::]:22          sshd.socket                  sshd.service
               kobject-uevent 1 systemd-udevd-kernel.socket  systemd-udevd.service

               5 sockets listed.

           Note: because the addresses might contains spaces, this output is not suitable for
           programmatic consumption.

           See also the options --show-types, --all, and --failed.

       list-timers [PATTERN...]
           List timer units ordered by the time they elapse next. If one or more PATTERNs are
           specified, only units matching one of them are shown.

           See also the options --all and --failed.

       start PATTERN...
           Start (activate) one or more units specified on the command line.

           Note that glob patterns operate on a list of currently loaded units. Units which are
                not
           active and are not in a failed state usually are not loaded, and would not be
               matched by
           any pattern. In addition, in case of instantiated units, systemd is often unaware of
                the
           instance name until the instance has been started. Therefore, using glob patterns
               with
           start has limited usefulness.

       stop PATTERN...
           Stop (deactivate) one or more units specified on the command line.

       reload PATTERN...
           Asks all units listed on the command line to reload their configuration. Note that
               this
           will reload the service-specific configuration, not the unit configuration file of
               systemd.
           If you want systemd to reload the configuration file of a unit, use the daemon-
               reload
           command. In other words: for the example case of Apache, this will reload Apache's
           httpd.conf in the web server, not the apache.service systemd unit file.

           This command should not be confused with the daemon-reload or load commands.

       restart PATTERN...
           Restart one or more units specified on the command line. If the units are not
```

```
            running yet,
        they will be started.

    try-restart PATTERN...
        Restart one or more units specified on the command line if the units are running.
            This does
        nothing if units are not running. Note that, for compatibility with Red Hat init
            scripts,
        condrestart is equivalent to this command.

    reload-or-restart PATTERN...
        Reload one or more units if they support it. If not, restart them instead. If the
            units are
        not running yet, they will be started.

    reload-or-try-restart PATTERN...
        Reload one or more units if they support it. If not, restart them instead. This does
        nothing if the units are not running. Note that, for compatibility with SysV init
            scripts,
        force-reload is equivalent to this command.

    isolate NAME
        Start the unit specified on the command line and its dependencies and stop all
            others.

        This is similar to changing the runlevel in a traditional init system. The isolate
            command
        will immediately stop processes that are not enabled in the new unit, possibly
            including
        the graphical environment or terminal you are currently using.

        Note that this is allowed only on units where AllowIsolate= is enabled. See systemd.
            unit(5)
        for details.

    kill PATTERN...
        Send a signal to one or more processes of the unit. Use --kill-who= to select which
            process
        to kill. Use --signal= to select the signal to send.

    is-active PATTERN...
        Check whether any of the specified units are active (i.e. running). Returns an exit
            code 0
        if at least one is active, or non-zero otherwise. Unless --quiet is specified, this
            will
        also print the current unit state to standard output.

    is-failed PATTERN...
        Check whether any of the specified units are in a "failed" state. Returns an exit
            code 0 if
        at least one has failed, non-zero otherwise. Unless --quiet is specified, this will
            also
        print the current unit state to standard output.

    status [PATTERN...|PID...]]
        Show terse runtime status information about one or more units, followed by most
            recent log
        data from the journal. If no units are specified, show system status. If combined
            with
        --all, also show the status of all units (subject to limitations specified with -t).
             If a
        PID is passed, show information about the unit the process belongs to.

        This function is intended to generate human-readable output. If you are looking for
        computer-parsable output, use show instead. By default this function only shows 10
            lines of
        output and ellipsizes lines to fit in the terminal window. This can be changes with
            --lines
        and --full, see above. In addition, journalctl --unit=NAME or journalctl --user-unit
            =NAME
        use a similar filter for messages and might be more convenient.

    show [PATTERN...|JOB...]
```

```
           Show properties of one or more units, jobs, or the manager itself. If no argument is
           specified, properties of the manager will be shown. If a unit name is specified,
               properties
           of the unit is shown, and if a job id is specified, properties of the job is shown.
               By
           default, empty properties are suppressed. Use --all to show those too. To select
               specific
           properties to show, use --property=. This command is intended to be used whenever
           computer-parsable output is required. Use status if you are looking for formatted
           human-readable output.

       cat PATTERN...
           Show backing files of one or more units. Prints the "fragment" and "drop-ins" (
               source
           files) of units. Each file is preceded by a comment which includes the file name.

       set-property NAME ASSIGNMENT...
           Set the specified unit properties at runtime where this is supported. This allows
               changing
           configuration parameter properties such as resource control settings at runtime. Not
                all
           properties may be changed at runtime, but many resource control settings (primarily
               those
           in systemd.resource-control(5)) may. The changes are applied instantly, and stored
               on disk
           for future boots, unless --runtime is passed, in which case the settings only apply
               until
           the next reboot. The syntax of the property assignment follows closely the syntax of
           assignments in unit files.

           Example: systemctl set-property foobar.service CPUShares=777

           Note that this command allows changing multiple properties at the same time, which
               is
           preferable over setting them individually. Like unit file configuration settings,
               assigning
           the empty list to list parameters will reset the list.

       help PATTERN...|PID...
           Show manual pages for one or more units, if available. If a PID is given, the manual
                pages
           for the unit the process belongs to are shown.

       reset-failed [PATTERN...]
           Reset the "failed" state of the specified units, or if no unit name is passed, reset
                the
           state of all units. When a unit fails in some way (i.e. process exiting with non-
               zero error
           code, terminating abnormally or timing out), it will automatically enter the "failed
               " state
           and its exit code and status is recorded for introspection by the administrator
               until the
           service is restarted or reset with this command.

       list-dependencies NAME
           Shows required and wanted units of the specified unit. If no unit is specified,
           default.target is implied. Target units are recursively expanded. When --all is
               passed, all
           other units are recursively expanded as well.

   Unit File Commands
       list-unit-files [PATTERN...]
           List installed unit files. If one or more PATTERNs are specified, only units whose
               filename
           (just the last component of the path) matches one of them are shown.

       enable NAME...
           Enable one or more unit files or unit file instances, as specified on the command
               line.
           This will create a number of symlinks as encoded in the "[Install]" sections of the
               unit
           files. After the symlinks have been created, the systemd configuration is reloaded (
               in a
```

```
            way that is equivalent to daemon-reload) to ensure the changes are taken into
                account
            immediately. Note that this does not have the effect of also starting any of the
                units
            being enabled. If this is desired, a separate start command must be invoked for the
                unit.
            Also note that in case of instance enablement, symlinks named the same as instances
                are
            created in the install location, however they all point to the same template unit
                file.

            This command will print the actions executed. This output may be suppressed by
                passing
            --quiet.

            Note that this operation creates only the suggested symlinks for the units. While
                this
            command is the recommended way to manipulate the unit configuration directory, the
            administrator is free to make additional changes manually by placing or removing
                symlinks
            in the directory. This is particularly useful to create configurations that deviate
                from
            the suggested default installation. In this case, the administrator must make sure
                to
            invoke daemon-reload manually as necessary to ensure the changes are taken into
                account.

            Enabling units should not be confused with starting (activating) units, as done by
                the
            start command. Enabling and starting units is orthogonal: units may be enabled
                without
            being started and started without being enabled. Enabling simply hooks the unit into
            various suggested places (for example, so that the unit is automatically started on
                boot or
            when a particular kind of hardware is plugged in). Starting actually spawns the
                daemon
            process (in case of service units), or binds the socket (in case of socket units),
                and so
            on.

            Depending on whether --system, --user, --runtime, or --global is specified, this
                enables
            the unit for the system, for the calling user only, for only this boot of the system
                , or
            for all future logins of all users, or only this boot. Note that in the last case,
                no
            systemd daemon configuration is reloaded.

        disable NAME...
            Disables one or more units. This removes all symlinks to the specified unit files
                from the
            unit configuration directory, and hence undoes the changes made by enable. Note
                however
            that this removes all symlinks to the unit files (i.e. including manual additions),
                not
            just those actually created by enable. This call implicitly reloads the systemd
                daemon
            configuration after completing the disabling of the units. Note that this command
                does not
            implicitly stop the units that are being disabled. If this is desired, an additional
                 stop
            command should be executed afterwards.

            This command will print the actions executed. This output may be suppressed by
                passing
            --quiet.

            This command honors --system, --user, --runtime and --global in a similar way as
                enable.

        is-enabled NAME...
            Checks whether any of the specified unit files are enabled (as with enable). Returns
                 an
```

```
        exit code of 0 if at least one is enabled, non-zero otherwise. Prints the current
            enable
        status (see table). To suppress this output, use --quiet.

        Table 1.  is-enabled output
        íÔÌíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ‹íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ

        íÔÁPrinted string    íÔÁ Meaning                    íÔÁ Return value íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ...íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ

        íÔÁ"enabled"         íÔÁ Enabled through a symlink íÔÁ              íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ/ in .wants directory
                  íÔÁ 0             íÔÁ
        íÔÁ"enabled-runtime" íÔÁ (permanently or just in   íÔÁ              íÔÁ
        íÔÁ                  íÔÁ /run)                      íÔÁ              íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ...íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ

        íÔÁ"linked"          íÔÁ Made available through a  íÔÁ              íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ/ symlink to the unit
            file  íÔÁ 1            íÔÁ
        íÔÁ"linked-runtime"  íÔÁ (permanently or just in   íÔÁ              íÔÁ
        íÔÁ                  íÔÁ /run)                      íÔÁ              íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ...íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ

        íÔÁ"masked"          íÔÁ Disabled entirely         íÔÁ              íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ/ (permanently or just in
               íÔÁ 1           íÔÁ
        íÔÁ"masked-runtime"  íÔÁ /run)                     íÔÁ              íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ...íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ

        íÔÁ"static"          íÔÁ Unit is not enabled, but  íÔÁ 0            íÔÁ
        íÔÁ                  íÔÁ has no provisions for     íÔÁ              íÔÁ
        íÔÁ                  íÔÁ enabling in [Install]     íÔÁ              íÔÁ
        íÔÁ                  íÔÁ section                   íÔÁ              íÔÁ
        íÔÞíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ...íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ

        íÔÁ"disabled"        íÔÁ Unit is not enabled       íÔÁ 1            íÔÁ
        íÔÔíÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ·íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ íÔ


    reenable NAME...
        Reenable one or more unit files, as specified on the command line. This is a
            combination of
        disable and enable and is useful to reset the symlinks a unit is enabled with to the
        defaults configured in the "[Install]" section of the unit file.

    preset NAME...
        Reset one or more unit files, as specified on the command line, to the defaults
            configured
        in the preset policy files. This has the same effect as disable or enable, depending
             how
        the unit is listed in the preset files.

        Use --preset-mode= to control whether units shall be enabled and disabled, or only
            enabled,
        or only disabled.

        For more information on the preset policy format, see systemd.preset(5). For more
        information on the concept of presets, please consult the Preset[1] document.

    preset-all
        Resets all installed unit files to the defaults configured in the preset policy file
             (see
        above).

        Use --preset-mode= to control whether units shall be enabled and disabled, or only
            enabled,
        or only disabled.

    mask NAME...
        Mask one or more unit files, as specified on the command line. This will link these
            units
        to /dev/null, making it impossible to start them. This is a stronger version of
```

```
            disable,
        since it prohibits all kinds of activation of the unit, including manual activation.
             Use
        this option with care. This honors the --runtime option to only mask temporarily
            until the
        next reboot of the system.

    unmask NAME...
        Unmask one or more unit files, as specified on the command line. This will undo the
            effect
        of mask.

    link FILENAME...
        Link a unit file that is not in the unit file search paths into the unit file search
             path.
        This requires an absolute path to a unit file. The effect of this can be undone with
        disable. The effect of this command is that a unit file is available for start and
            other
        commands although it is not installed directly in the unit search path.

    get-default
        Get the default target specified via default.target link.

    set-default NAME
        Set the default target to boot into. Command links default.target to the given unit.

Machine Commands
    list-machines [PATTERN...]
        List the host and all running local containers with their state. If one or more
            PATTERNs
        are specified, only containers matching one of them are shown.

Job Commands
    list-jobs [PATTERN...]
        List jobs that are in progress. If one or more PATTERNs are specified, only jobs for
             units
        matching one of them are shown.

    cancel JOB...
        Cancel one or more jobs specified on the command line by their numeric job IDs. If
            no job
        ID is specified, cancel all pending jobs.

Snapshot Commands
    snapshot [NAME]
        Create a snapshot. If a snapshot name is specified, the new snapshot will be named
            after
        it. If none is specified, an automatic snapshot name is generated. In either case,
            the
        snapshot name used is printed to standard output, unless --quiet is specified.

        A snapshot refers to a saved state of the systemd manager. It is implemented itself
            as a
        unit that is generated dynamically with this command and has dependencies on all
            units
        active at the time. At a later time, the user may return to this state by using the
            isolate
        command on the snapshot unit.

        Snapshots are only useful for saving and restoring which units are running or are
            stopped,
        they do not save/restore any other state. Snapshots are dynamic and lost on reboot.

    delete PATTERN...
        Remove a snapshot previously created with snapshot.

Environment Commands
    show-environment
        Dump the systemd manager environment block. The environment block will be dumped in
        straight-forward form suitable for sourcing into a shell script. This environment
            block
        will be passed to all processes the manager spawns.
```

```
       set-environment VARIABLE=VALUE...
           Set one or more systemd manager environment variables, as specified on the command
               line.

       unset-environment VARIABLE...
           Unset one or more systemd manager environment variables. If only a variable name is
           specified, it will be removed regardless of its value. If a variable and a value are
           specified, the variable is only removed if it has the specified value.

       import-environment VARIABLE...
           Import all, one or more environment variables set on the client into the systemd
               manager
           environment block. If no arguments are passed, the entire environment block is
               imported.
           Otherwise, a list of one or more environment variable names should be passed, whose
           client-side values are then imported into the manager's environment block.

   Manager Lifecycle Commands
       daemon-reload
           Reload systemd manager configuration. This will reload all unit files and recreate
               the
           entire dependency tree. While the daemon is being reloaded, all sockets systemd
               listens on
           on behalf of user configuration will stay accessible.

           This command should not be confused with the load or reload commands.

       daemon-reexec
           Reexecute the systemd manager. This will serialize the manager state, reexecute the
               process
           and deserialize the state again. This command is of little use except for debugging
               and
           package upgrades. Sometimes, it might be helpful as a heavy-weight daemon-reload.
               While the
           daemon is being reexecuted, all sockets systemd listening on behalf of user
               configuration
           will stay accessible.

   System Commands
       is-system-running
           Checks whether the system is running. This returns success when the system is fully
               up and
           running, meaning not in startup, shutdown or maintainance mode. Failure is returned
           otherwise. In addition, the current state is printed in a short string to standard
               output.
           Use --quiet to suppress output of this state string.

       default
           Enter default mode. This is mostly equivalent to isolate default.target.

       rescue
           Enter rescue mode. This is mostly equivalent to isolate rescue.target, but also
               prints a
           wall message to all users.

       emergency
           Enter emergency mode. This is mostly equivalent to isolate emergency.target, but
               also
           prints a wall message to all users.

       halt
           Shut down and halt the system. This is mostly equivalent to start halt.target
           --irreversible, but also prints a wall message to all users. If combined with --
               force,
           shutdown of all running services is skipped, however all processes are killed and
               all file
           systems are unmounted or mounted read-only, immediately followed by the system halt.
                If
           --force is specified twice, the operation is immediately executed without
               terminating any
           processes or unmounting any file systems. This may result in data loss.

       poweroff
```

```
        Shut down and power-off the system. This is mostly equivalent to start poweroff.
            target
        --irreversible, but also prints a wall message to all users. If combined with --
            force,
        shutdown of all running services is skipped, however all processes are killed and
            all file
        systems are unmounted or mounted read-only, immediately followed by the powering off
            . If
        --force is specified twice, the operation is immediately executed without
            terminating any
        processes or unmounting any file systems. This may result in data loss.

    reboot [arg]
        Shut down and reboot the system. This is mostly equivalent to start reboot.target
        --irreversible, but also prints a wall message to all users. If combined with --
            force,
        shutdown of all running services is skipped, however all processes are killed and
            all file
        systems are unmounted or mounted read-only, immediately followed by the reboot. If
            --force
        is specified twice, the operation is immediately executed without terminating any
            processes
        or unmounting any file systems. This may result in data loss.

        If the optional argument arg is given, it will be passed as the optional argument to
             the
        reboot(2) system call. The value is architecture and firmware specific. As an
            example,
        "recovery" might be used to trigger system recovery, and "fota" might be used to
            trigger a
        "firmware over the air" update.

    kexec
        Shut down and reboot the system via kexec. This is mostly equivalent to start kexec.
            target
        --irreversible, but also prints a wall message to all users. If combined with --
            force,
        shutdown of all running services is skipped, however all processes are killed and
            all file
        systems are unmounted or mounted read-only, immediately followed by the reboot.

    exit
        Ask the systemd manager to quit. This is only supported for user service managers (i
            .e. in
        conjunction with the --user option) and will fail otherwise.

    suspend
        Suspend the system. This will trigger activation of the special suspend.target
            target.

    hibernate
        Hibernate the system. This will trigger activation of the special hibernate.target
            target.

    hybrid-sleep
        Hibernate and suspend the system. This will trigger activation of the special
        hybrid-sleep.target target.

    switch-root ROOT [INIT]
        Switches to a different root directory and executes a new system manager process
            below it.
        This is intended for usage in initial RAM disks ("initrd"), and will transition from
             the
        initrd's system manager process (a.k.a "init" process) to the main system manager
            process.
        This call takes two arguments: the directory that is to become the new root
            directory, and
        the path to the new system manager binary below it to execute as PID 1. If the
            latter is
        omitted or the empty string, a systemd binary will automatically be searched for and
             used
        as init. If the system manager path is omitted or equal to the empty string, the
            state of
```

```
           the initrd's system manager process is passed to the main system manager, which
               allows
           later introspection of the state of the services involved in the initrd boot.

   Parameter Syntax
       Unit commands listed above take either a single unit name (designated as NAME), or
           multiple
       unit specifications (designated as PATTERN...). In the first case, the unit name with or
       without a suffix must be given. If the suffix is not specified, systemctl will append a
       suitable suffix, ".service" by default, and a type-specific suffix in case of commands
           which
       operate only on specific unit types. For example,

           # systemctl start sshd

       and

           # systemctl start sshd.service

       are equivalent, as are

           # systemctl isolate snapshot-11

       and

           # systemctl isolate snapshot-11.snapshot

       Note that (absolute) paths to device nodes are automatically converted to device unit
           names,
       and other (absolute) paths to mount unit names.

           # systemctl status /dev/sda
           # systemctl status /home

       are equivalent to:

           # systemctl status dev-sda.device
           # systemctl status home.mount

       In the second case, shell-style globs will be matched against currently loaded units;
           literal
       unit names, with or without a suffix, will be treated as in the first case. This means
           that
       literal unit names always refer to exactly one unit, but globs may match zero units and
           this is
       not considered an error.

       Glob patterns use fnmatch(3), so normal shell-style globbing rules are used, and "*", "?
           ", "[]"
       may be used. See glob(7) for more details. The patterns are matched against the names of
       currently loaded units, and patterns which do not match anything are silently skipped.
           For
       example:

           # systemctl stop sshd@*.service

       will stop all sshd@.service instances.

       For unit file commands, the specified NAME should be the full name of the unit file, or
           the
       absolute path to the unit file:

           # systemctl enable foo.service

       or

           # systemctl link /path/to/foo.service


EXIT STATUS
       On success, 0 is returned, a non-zero failure code otherwise.

ENVIRONMENT
```

```
       $SYSTEMD_PAGER
           Pager to use when --no-pager is not given; overrides $PAGER. Setting this to an
               empty
           string or the value "cat" is equivalent to passing --no-pager.

       $SYSTEMD_LESS
           Override the default options passed to less ("FRSXMK").

SEE ALSO
       systemd(1), systemadm(1), journalctl(1), loginctl(1), systemd.unit(5), systemd.resource-
       management(5), systemd.special(7), wall(1), systemd.preset(5)glob(7)

NOTES
        1. Preset
           http://freedesktop.org/wiki/Software/systemd/Preset

systemd 215                                                                    SYSTEMCTL(1)
```

40.13.3 'man 7 systemd.special' man page

Listing 76: man 7 systemd.special

```
SYSTEMD.SPECIAL(7)              systemd.special          SYSTEMD.SPECIAL(7)
NAME
       systemd.special - Special systemd units

SYNOPSIS
       basic.target, bluetooth.target, ctrl-alt-del.target, cryptsetup.target, cryptsetup-pre.
           target,
       dbus.service, dbus.socket, default.target, display-manager.service, emergency.target,
       exit.target, final.target, getty.target, graphical.target, halt.target, hibernate.target
           ,
       hybrid-sleep.target, initrd-fs.target, kbrequest.target, kexec.target, local-fs.target,
       local-fs-pre.target, multi-user.target, network.target, network-online.target,
       network-pre.target, nss-lookup.target, nss-user-lookup.target, paths.target, poweroff.
           target,
       printer.target, reboot.target, remote-fs.target, remote-fs-pre.target, rescue.target,
       initrd-root-fs.target, rpcbind.target, runlevel2.target, runlevel3.target, runlevel4.
           target,
       runlevel5.target, shutdown.target, sigpwr.target, sleep.target, smartcard.target,
       sockets.target, sound.target, suspend.target, swap.target, sysinit.target, syslog.socket
           ,
       system-update.target, time-sync.target, timers.target, umount.target, -.slice, system.
           slice,
       user.slice, machine.slice

DESCRIPTION
       A few units are treated specially by systemd. They have special internal semantics and
           cannot
       be renamed.

SPECIAL SYSTEM UNITS
       basic.target
           A special target unit covering basic boot-up.

           systemd automatically adds dependencies of the types Requires= and After= for this
               target
           unit to all services (except for those with DefaultDependencies=no).

           Usually this should pull-in all mount points, swap devices, sockets, timers, and
               path units
           and other basic initialization necessary for general purpose daemons.

       ctrl-alt-del.target
           systemd starts this target whenever Control+Alt+Del is pressed on the console.
               Usually this
           should be aliased (symlinked) to reboot.target.

       cryptsetup.target
           A target that pulls in setup services for all encrypted block devices.

       dbus.service
```

```
        A special unit for the D-Bus bus daemon. As soon as this service is fully started up
        systemd will connect to it and register its service.

    dbus.socket
        A special unit for the D-Bus system bus socket. All units with Type=dbus
            automatically gain
        a dependency on this unit.

    default.target
        The default unit systemd starts at bootup. Usually this should be aliased (symlinked
            ) to
        multi-user.target or graphical.target.

        The default unit systemd starts at bootup can be overridden with the systemd.unit=
            kernel
        command line option.

    display-manager.service
        The display manager service. Usually this should be aliased (symlinked) to gdm.
            service or a
        similar display manager service.

    emergency.target
        A special target unit that starts an emergency shell on the main console. This unit
            is
        supposed to be used with the kernel command line option systemd.unit= and has
            otherwise
        little use.

    final.target
        A special target unit that is used during the shutdown logic and may be used to pull
             in
        late services after all normal services are already terminated and all mounts
            unmounted.

    getty.target
        A special target unit that pulls in statically configured local TTY getty instances.

    graphical.target
        A special target unit for setting up a graphical login screen. This pulls in
        multi-user.target.

        Units that are needed for graphical logins shall add Wants= dependencies for their
            unit to
        this unit (or multi-user.target) during installation. This is best configured via
        WantedBy=graphical.target in the unit's "[Install]" section.

    hibernate.target
        A special target unit for hibernating the system. This pulls in sleep.target.

    hybrid-sleep.target
        A special target unit for hibernating and suspending the system at the same time.
            This
        pulls in sleep.target.

    halt.target
        A special target unit for shutting down and halting the system. Note that this
            target is
        distinct from poweroff.target in that it generally really just halts the system
            rather than
        powering it down.

        Applications wanting to halt the system should start this unit.

    initrd-fs.target
        systemd-fstab-generator(3) automatically adds dependencies of type Before= to
        sysroot-usr.mount and all mount points found in /etc/fstab that have x-initrd.mount
            and not
        have noauto mount options set.

    kbrequest.target
        systemd starts this target whenever Alt+ArrowUp is pressed on the console. This is a
             good
```

```
        candidate to be aliased (symlinked) to rescue.target.

    kexec.target
        A special target unit for shutting down and rebooting the system via kexec.

        Applications wanting to reboot the system with kexec should start this unit.

    local-fs.target
        systemd-fstab-generator(3) automatically adds dependencies of type Before= to all
            mount
        units that refer to local mount points for this target unit. In addition, it adds
        dependencies of type Wants= to this target unit for those mounts listed in /etc/
            fstab that
        have the auto mount option set.

        systemd automatically adds dependencies of type After= for this target unit to all
            SysV
        init script service units with an LSB header referring to the "$local_fs" facility.

    multi-user.target
        A special target unit for setting up a multi-user system (non-graphical). This is
            pulled in
        by graphical.target.

        Units that are needed for a multi-user system shall add Wants= dependencies for
            their unit
        to this unit during installation. This is best configured via WantedBy=multi-user.
            target in
        the unit's "[Install]" section.

    network-online.target
        Units that strictly require a configured network connection should pull in
        network-online.target (via a Wants= type dependency) and order themselves after it.
            This
        target unit is intended to pull in a service that delays further execution until the
        network is sufficiently set up. What precisely this requires is left to the
            implementation
        of the network managing service.

        Note the distinction between this unit and network.target. This unit is an active
            unit
        (i.e. pulled in by the consumer rather than the provider of this functionality) and
            pulls
        in a service which possibly adds substantial delays to further execution. In
            contrast,
        network.target is a passive unit (i.e. pulled in by the provider of the
            functionality,
        rather than the consumer) that usually does not delay execution much. Usually,
        network.target is part of the boot of most systems, while network-online.target is
            not,
        except when at least one unit requires it. Also see Running Services After the
            Network is
        up[1] for more information.

        All mount units for remote network file systems automatically pull in this unit, and
             order
        themselves after it. Note that networking daemons that simply provide functionality
            to
        other hosts generally do not need to pull this in.

    paths.target
        A special target unit that sets up all path units (see systemd.path(5) for details)
            that
        shall be active after boot.

        It is recommended that path units installed by applications get pulled in via Wants=
        dependencies from this unit. This is best configured via a WantedBy=paths.target in
            the
        path unit's "[Install]" section.

    poweroff.target
        A special target unit for shutting down and powering off the system.
```

```
           Applications wanting to power off the system should start this unit.

           runlevel0.target is an alias for this target unit, for compatibility with SysV.

       reboot.target
           A special target unit for shutting down and rebooting the system.

           Applications wanting to reboot the system should start this unit.

           runlevel6.target is an alias for this target unit, for compatibility with SysV.

       remote-fs.target
           Similar to local-fs.target, but for remote mount points.

           systemd automatically adds dependencies of type After= for this target unit to all
               SysV
           init script service units with an LSB header referring to the "$remote_fs" facility.

       rescue.target
           A special target unit for setting up the base system and a rescue shell.

           runlevel1.target is an alias for this target unit, for compatibility with SysV.

       initrd-root-fs.target
           systemd-fstab-generator(3) automatically adds dependencies of type Before= to the
           sysroot.mount unit, which is generated from the kernel command line.

       runlevel2.target, runlevel3.target, runlevel4.target, runlevel5.target
           These are targets that are called whenever the SysV compatibility code asks for
               runlevel 2,
           3, 4, 5, respectively. It is a good idea to make this an alias for (i.e. symlink to)
           multi-user.target (for runlevel 2) or graphical.target (the others).

       shutdown.target
           A special target unit that terminates the services on system shutdown.

           Services that shall be terminated on system shutdown shall add Conflicts=
               dependencies to
           this unit for their service unit, which is implicitly done when DefaultDependencies=
               yes is
           set (the default).

       sigpwr.target
           A special target that is started when systemd receives the SIGPWR process signal,
               which is
           normally sent by the kernel or UPS daemons when power fails.

       sleep.target
           A special target unit that is pulled in by suspend.target, hibernate.target and
           hybrid-sleep.target and may be used to hook units into the sleep state logic.

       sockets.target
           A special target unit that sets up all socket units.(see systemd.socket(5) for
               details)
           that shall be active after boot.

           Services that can be socket-activated shall add Wants= dependencies to this unit for
                their
           socket unit during installation. This is best configured via a WantedBy=sockets.
               target in
           the socket unit's "[Install]" section.

       suspend.target
           A special target unit for suspending the system. This pulls in sleep.target.

       swap.target
           Similar to local-fs.target, but for swap partitions and swap files.

       sysinit.target
           A special target unit covering early boot-up scripts.

       syslog.socket
           The socket unit syslog implementations should listen on. All userspace log messages
```

```
               will be
           made available on this socket. For more information about syslog integration, please
           consult the Syslog Interface[2] document.

       system-update.target
           A special target unit that is used for off-line system updates.  systemd-system-
               update-
           generator(8) will redirect the boot process to this target if /system-update exists.
                For
           more information see the System Updates Specification[3].

       timers.target
           A special target unit that sets up all timer units (see systemd.timer(5) for details
               ) that
           shall be active after boot.

           It is recommended that timer units installed by applications get pulled in via Wants
               =
           dependencies from this unit. This is best configured via WantedBy=timers.target in
               the
           timer unit's "[Install]" section.

       umount.target
           A special target unit that umounts all mount and automount points on system shutdown
               .

           Mounts that shall be unmounted on system shutdown shall add Conflicts dependencies
               to this
           unit for their mount unit, which is implicitly done when DefaultDependencies=yes is
               set
           (the default).

SPECIAL SYSTEM UNITS FOR DEVICES
       Some target units are automatically pulled in as devices of certain kinds show up in the
       system. These may be used to automatically activate various services based on the
           specific type
       of the available hardware.

       bluetooth.target
           This target is started automatically as soon as a Bluetooth controller is plugged in
                or
           becomes available at boot.

           This may be used to pull in Bluetooth management daemons dynamically when Bluetooth
           hardware is found.

       printer.target
           This target is started automatically as soon as a printer is plugged in or becomes
           available at boot.

           This may be used to pull in printer management daemons dynamically when printer
               hardware is
           found.

       smartcard.target
           This target is started automatically as soon as a smartcard controller is plugged in
                or
           becomes available at boot.

           This may be used to pull in smartcard management daemons dynamically when smartcard
           hardware is found.

       sound.target
           This target is started automatically as soon as a sound card is plugged in or
               becomes
           available at boot.

           This may be used to pull in audio management daemons dynamically when audio hardware
                is
           found.

SPECIAL PASSIVE SYSTEM UNITS
       A number of special system targets are defined that can be used to properly order boot-
```

```
        up of
       optional services. These targets are generally not part of the initial boot transaction,
            unless
       they are explicitly pulled in by one of the implementing services. Note specifically
        that these
       passive target units are generally not pulled in by the consumer of a service, but by
        the
       provider of the service. This means: a consuming service should order itself after these
       targets (as appropriate), but not pull it in. A providing service should order itself
        before
       these targets (as appropriate) and pull it in (via a Wants= type dependency).

       Note that these passive units cannot be started manually, i.e.  "systemctl start
       time-sync.target" will fail with an error. They can only be pulled in by dependency.
        This is
       enforced since they exist for ordering purposes only and thus are not useful as only
        unit
       within a transaction.

       cryptsetup-pre.target
           This passive target unit may be pulled in by services that want to run before any
            encrypted
           block device is set up. All encrypted block devices are set up after this target has
                been
           reached. Since the shutdown order is implicitly the reverse start-up order between
            units,
           this target is particularly useful to ensure that a service is shut down only after
            all
           encrypted block devices are fully stopped.

       local-fs-pre.target
           This target unit is automatically ordered before all local mount points marked with
            auto
           (see above). It can be used to execute certain units before all local mounts.

       network.target
           This unit is supposed to indicate when network functionality is available, but it is
                only
           very weakly defined what that is supposed to mean, with one exception: at shutdown,
            a unit
           that is ordered after network.target will be stopped before the network -- to
            whatever
           level it might be set up then -- is shut down. It is hence useful when writing
            service
           files that require network access on shutdown, which should order themselves after
            this
           target, but not pull it in. Also see Running Services After the Network is up[1] for
                more
           information. Also see network-online.target described above.

           systemd automatically adds dependencies of type After= for this target unit to all
            SysV
           init script service units with an LSB header referring to the "$network" facility.

       network-pre.target
           This passive target unit may be pulled in by services that want to run before any
            network
           is set up, for example for the purpose of setting up a firewall. All network
            management
           software orders itself after this target, but does not pull it in.

       nss-lookup.target
           A target that should be used as synchronization point for all host/network name
            service
           lookups. Note that this is independent of user/group name lookups for which
           nss-user-lookup.target should be used. All services for which the availability of
            full
           host/network name resolution is essential should be ordered after this target, but
            not pull
           it in. systemd automatically adds dependencies of type After= for this target unit
            to all
           SysV init script service units with an LSB header referring to the "$named" facility
            .
```

```
       nss-user-lookup.target
           A target that should be used as synchronization point for all user/group name
               service
           lookups. Note that this is independent of host/network name lookups for which
           nss-lookup.target should be used. All services for which the availability of the
               full
           user/group database is essential should be ordered after this target, but not pull
               it in.
           Note that system users are always resolvable, and hence do not require any special
               ordering
           against this target.

       remote-fs-pre.target
           This target unit is automatically ordered before all remote mount point units (see
               above).
           It can be used to run certain units before the remote mounts are established. Note
               that
           this unit is generally not part of the initial transaction, unless the unit that
               wants to
           be ordered before all remote mounts pulls it in via a Wants= type dependency. If the
                unit
           wants to be pulled in by the first remote mount showing up, it should use
           network-online.target (see above).

       rpcbind.target
           The portmapper/rpcbind pulls in this target and orders itself before it, to indicate
                its
           availability. systemd automatically adds dependencies of type After= for this target
                unit
           to all SysV init script service units with an LSB header referring to the "$portmap"
           facility.

       time-sync.target
           Services responsible for synchronizing the system clock from a remote source (such
               as NTP
           client implementations) should pull in this target and order themselves before it.
               All
           services where correct time is essential should be ordered after this unit, but not
               pull it
           in. systemd automatically adds dependencies of type After= for this target unit to
               all SysV
           init script service units with an LSB header referring to the "$time" facility.

SPECIAL USER UNITS
       When systemd runs as a user instance, the following special units are available, which
           have
       similar definitions as their system counterparts: default.target, shutdown.target,
       sockets.target, timers.target, paths.target, bluetooth.target, printer.target,
       smartcard.target, sound.target.

       In addition, the following special unit is understood only when systemd runs as service
       instance:

       exit.target
           A special service unit for shutting down the user service manager.

           Applications wanting to terminate the user service manager should start this unit.
               If
           systemd receives SIGTERM or SIGINT when running as user service daemon, it will
               start this
           unit.

           Normally, this pulls in shutdown.target which in turn should be conflicted by all
               units
           that want to be shut down on user service manager exit.

SPECIAL SLICE UNITS
       There are four ".slice" units which form the basis of the hierarchy for assignment of
           resources
       for services, users, and virtual machines or containers.

       -.slice
```

```
           The root slice is the root of the hierarchy. It usually does not contain units
               directly,
           but may be used to set defaults for the whole tree.

       system.slice
           By default, all services services started by systemd are found in this slice.

       user.slice
           By default, all user processes and services started on behalf of the user, including
                the
           per-user systemd instance are found in this slice.

       machine.slice
           By default, all virtual machines and containers registered with systemd-machined are
                found
           in this slice.

SEE ALSO
       systemd(1), systemd.unit(5), systemd.service(5), systemd.socket(5), systemd.target(5),
       systemd.slice(5), bootup(7), systemd-fstab-generator(8)

NOTES
        1. Running Services After the Network is up
           http://www.freedesktop.org/wiki/Software/systemd/NetworkTarget

        2. Syslog Interface
           http://www.freedesktop.org/wiki/Software/systemd/syslog

        3. System Updates Specification
           http://freedesktop.org/wiki/Software/systemd/SystemUpdates

systemd 215                                                                    SYSTEMD.
    SPECIAL(7)
```

41 embedded Schaltnetzteil

Schaltnetzteile werden modernerweise von ES (embedded Systems), typischerweise uC (Microcontroller), geregelt, wozu sie mit PWM-Ausgängen, Puls-Countern und ADCs ausgestattet werden. Da der Arbeitskreis hierbei aus Leistungsschaltern (MosFET, IGBT) und Spulenbauteilen (Drossel (choke), Transformator/Übertrager (transformer)) besteht, trifft vieles auch auf E-Motorsteuerungen (Stator-/Rotorwicklungen) zu.

41.1 Die SPULE (inductor, coil, solenoid, choke)

Das Denken, Berechnen und Entwerfen mit *Spulen* ist derzeit leider unbeliebt, wohl weil zu fremd, weswegen iXH hier zuerst mit etwas kindischen Vergleichen ein *"Bauchgefühl"* für die Wickeldinger vermitteln möchte.

Die SPULE wirkt auf den Strom
wie ein SCHWUNGRAD
man muss es
mit *Kraft (U)* erst in *Schwung (I)*
bringen

d.h. anfangs fließt kein Strom.
↓
Der STROMfluss ist mittels *SPANNUNG* erst ins *"Fließen" (I)* zu bringen.

Wir machen gedanklich folgenden Vergleich:

Spannung U	≡	Kraft F
Strom I	≡	Drehzahl n (Winkelgeschwindigkeit $\omega = n \cdot 2\pi$
Energie $W = L\frac{I^2}{2} \equiv \frac{mv^2}{2}$	≡	*'Schwung'* $W = T\frac{\omega^2}{2}$
Induktivität L	≡	Trägheitsmoment T

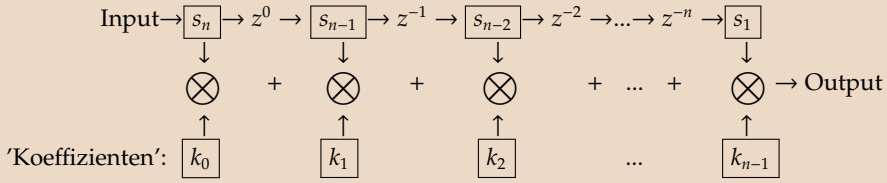
Ein Schwungrad dreht sich schneller, wenn ich
stärker (Drehmoment M)
oder
länger (Zeit t)
anschiebe (jedes Teilchen):

schneller	=	**stärker** * **länger**
$L * \Delta I$	=	$\underbrace{U_{const} * \Delta t}$
	=	*'Spannungs - Zeit - Fläche'*

Die Vorstellung, man müsse die Elektronen der Spulenwicklung mittels Antriebskraft U in Schwung bringen, ist falsch aber *hilfreich*.

Teil X

Finde die Fehler



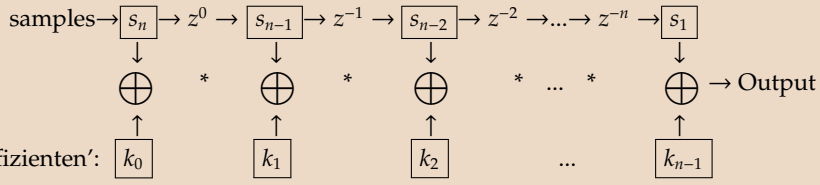
s ... samples-Schieberegister
z ... Zeit-Verschieber
k ... Filterkoeffizienten

```
for(i=0; i<=N; i++)
    SinMag = sample[i]*coeff[i];
coeff[N] = coeff[N-1];
coeff[0] = coeff[1];
```

```
for(i=0; i<N; i++)
    SinMag = sample[i]*coeff[i];
for(i=N; i>0; i--)
    samp[i] += samp[i+1];
```

Input → ADC → S&H → Aliases

Input → S&H → ADC → Anti-Aliasing Filter → Aliases

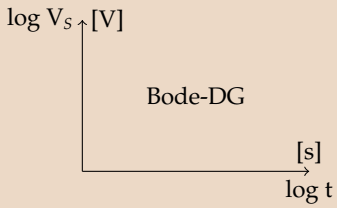


```
for(i=0; i<N; i++)
    SinMag = sample[i-1]*coeff[i-1];
for(i=N; i>1; i--)
    samp[i] = samp[i+1];
```

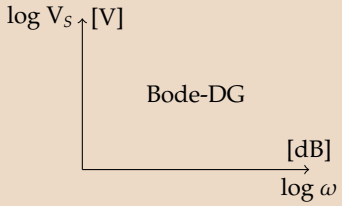
Finde Fehler:

Spektrum → S&H → ADC → Anti-Aliasing Filter → FFT

Input → Anti-Aliasing Filter → DFT → FFT → Spektrum



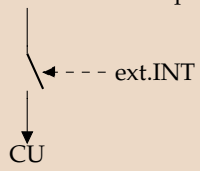
V_S . . . Verstärkung der Spannung (engl. A_V . . . amplification of voltage)



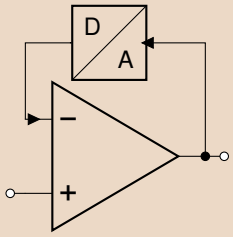
V_S . . . Verstärkung der Spannung (engl. A_V . . . amplification of voltage)

Finde Fehler:

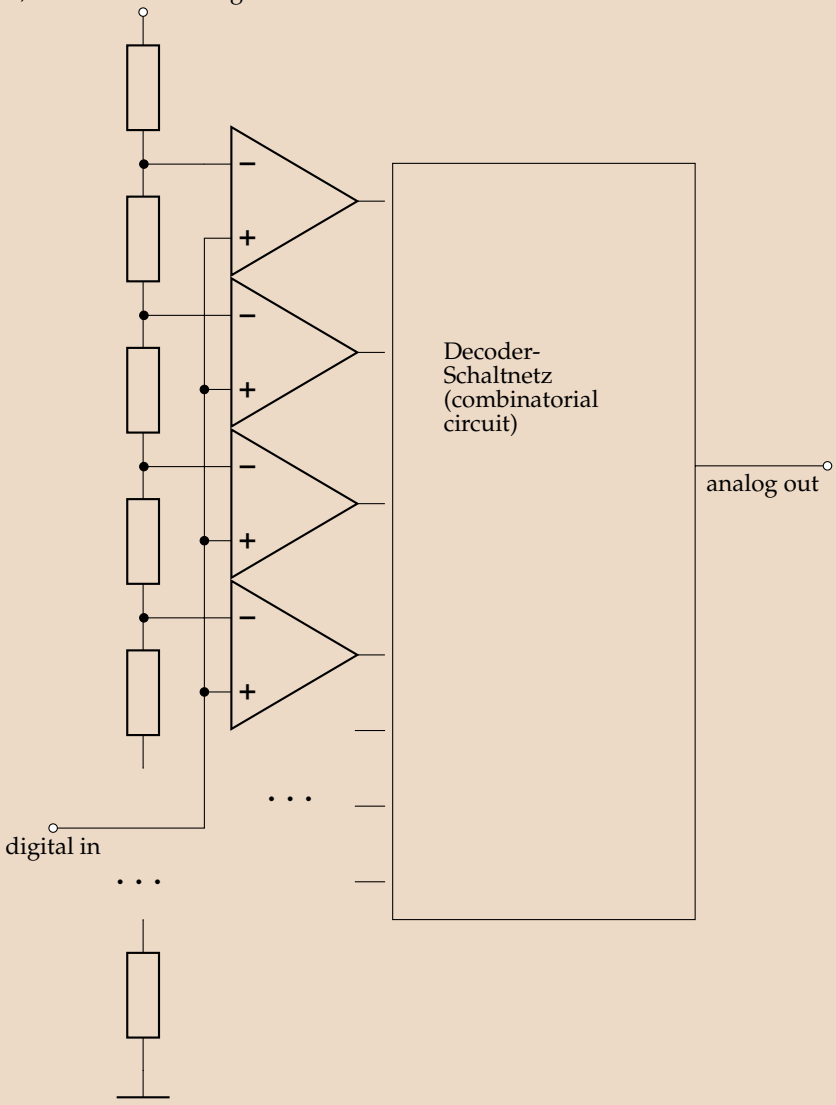
Bei externem Interrupt-Signal wird einfach der CU die CPU-clock 'clk' abgeschaltet?



Wenn ma einen DAC in die Gegenkopplung eines OPamp schaltet, entsteht ein ADC?



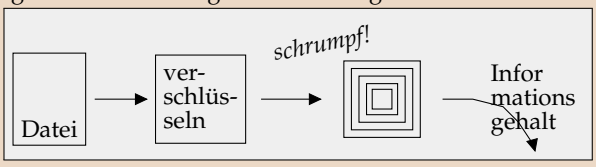
Aus einer Spannungsteilerkette und 255 OPamps lässt sich ein wunderbarer 8-Bit-'Flash-DAC' herstellen, der ohne Wandlungs-Zwischenschritte auskommt?



Finde Fehler:

Versierte Kryptoanalysten können auch einen guten Code *'knacken'*

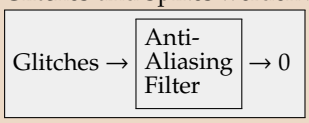
Eine gute Chiffre verringert auch Dateigröße und Informationsgehalt



Zufallszahlengenerator:

```
long long integer P1,P2,zufzi;
zufzi = (zufzi % P1) * P2;
```

Glitches und Spikes werden beim Anti-Aliasing sowieso rausgefiltert



Teil XI
Dic4a U-Mitschrift

Dic4 ab XcV=31Mar'23 GesamtWH
- VHDL Prozessor 114
- Fourier-Analyse 399
- VHDL-FIR 411
- Digitalisierung CD, (anti-)Aliasing, Oversampling, dB-Rechnung, 435
- MAC in VHDL 186
- DAC und ADC 188
- Logic-Specs 12
- TTL-NAND, Dimensg., Anpassungen 12
- USB 221
- SPI 197
- I2C 200
- CAN 205
- LVDS, TMDS, HDMI... 212
- IPsocket 374
- Timer,PWM 183
- ISR 262
- ISR 256
- ISR 100
- PWM in VHDL 215
- RasPi 374

Dic4 XcO=24Mar'23 RasPi-Linux/ES
- RaspberryPi Inbetriebnahme
- IP-Socket-Programmierungen aufm RasPi
- →Utility 'netcat' (auch 'nc' oder 'ncat': man 1 netcat) S.386
- kochenMitNick:
  - Zwiebel und Speck dazugeben
  - würzen mit Suppenpulver
  - lass die Papf ganz ('mehlige'), evtl halbieren, Butterschmalz dazu
  - Braten braucht viel Zeit (halben Tag), nicht kochen – nur 'simmern'/pochieren
    viel Zwiebel, Tomaten, Karotten und Papf mitkochen (prima Sauce), großer, hoher Topf,
    idealerweise mehrmals aufwärmen
  - Fritatten statt Nudeln schmecken viel besser (zB. Gulasch), in Butterschmalz backen
  - panierte Schnitzel in Butterschmalz herausbacken
  - Papf wern sowieso nur mit Eisenpfanne und Holzherd +Majoran+Zwiebel+Speck

Dic4 XcH=17Mar'23 SAMBA+IPsocket+USBprüfung
- →Windows-Netzlaufwerk mappen (S.616)
- →selfmade Webserver (S.374)

Dic4 Xc7=07Mar'23 obelixAccount+USBprüfung
- →obelix-Account (S.616)
- →LinuxCommands (S.298)
- USB-Prüfungen

Dic4 XbO=24Feb'23 USB
- antworte auf englisch
- ist USB-Signalisierung 'ground referenced', 'single-sided', 'commom-mode', 'symmetrisch'?
- ist USB eine Bus-, Stern-, Baum-, Ring- oder eine Peer-to-Peer-Topologie?
- USB zieht im J-Zustand die Datenleitungen D+ bzw. D- auf $\geq 2.8V$, die jeweils andere nach $\leq 0.3V$. Ein K wird mit vertauschten Spannungen signalisiert. warum kann USB nicht mittels Übertrager potentialgetrennt werden?

- USB codiert Bits mit NRZI: No-Return-to-Zero-Inverted dh. '1' wird durch Pegelwechsel dargestell, ohne den eine '0' entsteht.
- nenne die USB-Specs
- welche Spannungen und Ströme tragen USB-Leitungen im J-Zustand
  bei USB-1?
  bei USB-2?
  bei USB-3?
- welche Spannungen und Ströme tragen USB-Leitungen im K-Zustand
  bei USB-1?
  bei USB-2?
  bei USB-3?
- wie kommt man von J und K Zuständen zu binären '0' und '1'?
- welche 'Data Flow Type' genannten Kommunikationsvarianten gibt es
  bei USB-1?
  bei USB-2?
  bei USB-3?
- wofür dienen sie?
- was unterscheidet 'Bitrate' und 'Datenrate'?
- wie heißt dieser Unterschied?
- warum spezifizert USB-3-SuperSpeed eine Konstantstromstärke von 17,78mA anstelle von Spannungen?

Dic4 XaV=31Jan'23 Logik-ICs

- zeichne die TTL-NAND Innenschaltung (7400)
- nenne die TTL Output-Specs
- nenne die TTL Input-Specs
- zeichne die CMOS Inverter-, NAND- und NOR-Schaltungen
- ermittle die Knotenspannungen im Output Low- und High-Zustand der TTL-NAND-Gatter
- was braucht eine 5V-CMOS-zu-LS-TTL Anpassung?
- was benötigt eine 5V-LSTTL-zu-5V-CMOS Anpassung?
- wie kompatibel sind 3.3V-IIC und 5V-CMOS?
- wieviel Strom liefert ein AVR-Tiny13-Digital-Ausgang?
- die Strom-Zählrichtung ist immer ins Bauteil hinein?

Dic4 XaO=24Jan'23 Specs+Anpassung

- zeichne die TTL-NAND Schaltung (7400)
- nenne die TTL Output-Specs
- nenne die TTL Input-Specs
- was braucht eine 5V-CMOS-zu-LS-TTL Anpassung?
- was benötigt eine 5V-LSTTL-zu-5V-CMOS Anpassung?
- wie kompatibel sind 3.3V-IIC und 5V-CMOS?
- wieviel Strom liefert ein AVR-Tiny13-Digital-Ausgang?
- die Strom-Zählrichtung ist immer ins Bauteil hinein?
- was bringt 'differentielle Signalisierung'?
- kann man ein 'Twisted Pair' via Übertrager einspeisen?

Dic4 XaK=20Jan'23 DAC und ADC

- 'Wägeverfahren' auf Englisch?
- 'accuracy' deutsch?
- 'Auflösung' englisch?
- Wie funktioniert der slope-ADC?
- Was ist ein Monotonie-Fehler?
- Was ist ein Verstärkings-Fehler? (gain error)?
- Was ist ein differentieller Linearitäts-Fehler (differential non-linearitiy)?
- Was ist ein integraler Linearitäts-Fehler (integral non-linearitiy)?
- erkläre folgende Datenblattangaben:
  Resolution 60[uV]
  Supply Current 3.5[mA]

Integral Non-Linearity ±0.5[LSB]
Differential Non-Linearity ±0.5[LSB]
Offset Voltage -40[mV] to 55[mV]
Analog Input (-1dB) BW 18[MHz]
Differential Gain Error 1.0[%]
Aperture Jitter t_AJ 40[ps]
Sampling Delay t_SD 4[ns]
clock skew [4ns]

Dic4 XaA=10Jan'23 DAC und ADC

- 'Verstärkungsfehler' auf Englisch?
- 'Linearitätsfehler' auf Englisch?
- 'Wägeverfahren' auf Englisch?
- 'accuracy' deutsch?
- Wie funktioniert der R-2R-DAC?
- Wie funktioniert der Flash-ADC?
- Wie funktioniert der slope-ADC?
- Wie funktioniert der SAR-ADC?
- Was ist ein Monotonie-Fehler?
- Was ist ein Offset-Fehler (offset error)?
- Was ist ein Verstärkings-Fehler? (gain error)?
- Was ist ein differentieller Linearitäts-Fehler (differential non-linearitiy)?
- Was ist ein integraler Linearitäts-Fehler (integral non-linearitiy)?
- erkläre folgende Datenblattangaben:
  Resolution 60[uV]
  Resolution 20[Bit]
  Supply Current 3.5[mA]
  Integral Non-Linearity ±0.5[LSB]
  Differential Non-Linearity ±0.5[LSB]
  Offset Voltage -40[mV] to 55[mV]
  Analog Input (-1dB) BW 18[MHz]
  Differential Gain Error 1.0[%]
  Aperture Jitter t_AJ 40[ps]
  Sampling Delay t_SD 4[ns]
  clock skew [4ns]

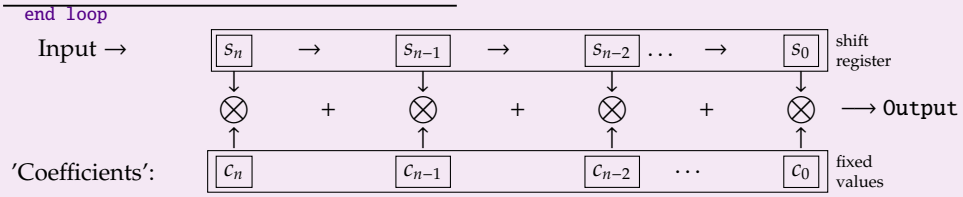
Dic4 WkQ=26Nov FIR mit VHDL-'FOR'

iXH bin durch Rückfragen und Hinweise überzeugt worden, die Aufgabe doch mit dem VHDL-FOR zu lösen.

++ mit 'FOR' ist es einfacher
++ es entsteht genau die Struktur wie im Blockdiagramm
-- (iXH wollte uns die MAC-Unit eines DSP codieren lassen)

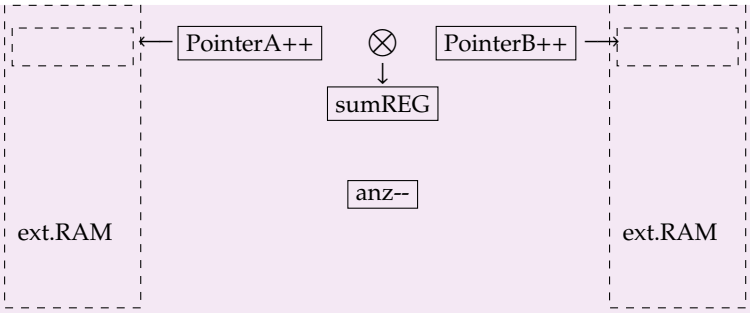
FIR Filter:

```
FOR i in 0 to 1023 loop
  sum <= sum + sample(i)*coeff(i);
end loop
```



Des erzeugt allerdings ein komplettes, aber unflexibles FIR (kann nur FIR mit fix 1024 Schieberegister-Zellen. So wird es in einem DSP (Digital SignalProcessor) nicht gemacht! Ein DSP hat nur 1 Stk Multiplizierwerk und hat Samples sowie Koeffizienten in Pointer-addressierbaren RAM-Speichern ('PointerA', 'PointerB').

DSP MAC Unit:



Dic4 WkP=25Nov Hansis VHDL

'Chrischtoff, ischdes des FIR-VHDL, was Du wolltesch?'
→ wie kimpma auf die Zahl '5580'? - weissi nit
→ was isch des "5-1" füra extragi Syntax? - weissi nit
☺

```
architecture MyHDL of sflt is
  type t_array_taps is array(0 to 6-1) of signed (15 downto 0);
  signal taps: t_array_taps;
begin
  SFLT_RTL_FILTER: process (clk) is
    variable sum: integer;
  begin
    if rising_edge(clk) then
        sum := to_integer(x * 5580);
        sum := to_integer(sum + (taps(0) * 5750));
        sum := to_integer(sum + (taps(1) * 6936));
        sum := to_integer(sum + (taps(2) * 6936));
        sum := to_integer(sum + (taps(3) * 5750));
        sum := to_integer(sum + (taps(4) * 5580));
        taps(0) <= x;
        for ii in 1 to 5-1 loop
            taps(ii) <= taps((ii - 1));
        end loop;
        y <= to_signed(sum, 16);
    end if;
  end process SFLT_RTL_FILTER;
end architecture MyHDL;
```

Wennma 't_array_taps' googelt, kimp ma sfort direkt auf den Code da. Der schreibt [...] I'm trying to get started with DSP in my Spartan-3 board. I made a AC97 board with a chip from an old motherboard, and so far I got it to do ADC, multiply the samples for a number <1 (decrease volume) and then DAC.

Now I'd like to do some basic DSP stuff, like a low-pass filter, high-pass etc. But I'm really confused about numeric representation (integers? fixed point? Q0.15? Overflow or saturation?).

I just want some example code of an actual simple filter to get me started. No high-efficiency, fast, or anything like that. Just the theoretical filter implemented in VHDL.

I've been searching but I just find theoretical formulas - I get that, what I don't understand is how to process the signed 16-bit, 48KHz audio samples I'm getting from the ADC. I've been using these libraries: http://www.vhdl.org/fphdl/. If I multiply my samples by 0.5, 0.25, etc, I can hear the difference. But a larger filter gives me just noise.

Another simple code snippet (just the guts). Note I didn't write the VHDL directly, I used MyHDL to generate the VHDL. [...]

Dic4 WkM=12.Nov CU in VHDL, FIR

- Was schreibt man in die 'ENTITY':
  → s. unten 'WjB+WjI'
- Was schreibt man in die 'ARCHUTECTURE':
  → s. unten 'WjB+WjI'
- Skalar-Produkt-Rechnung wird in der digitalen Signalverarbeitung (DSP)
  auch **MAC-operation** und **convolution**
  (korrekt: *Kreuzkorrelation cross correlation*) genannt.

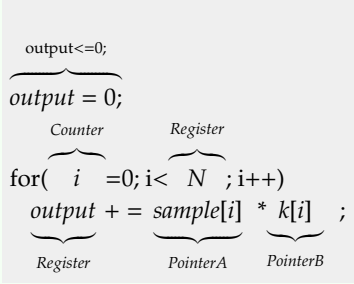
SMUE-Fragen:

- schreib das FIR-Filter von Peudo-C nach VHDL um
  (dann entsteht eine Hardware "MAC-Unit")

  Ein Prozessor mit **MAC**-Unit (**M**ultiply **A**nd **C**umulate)
  wird "DSP" genannt (Digital Signal Processor).
- studiere bei den *'Anwendungen der MAC Operation'* die →*Fourier-Transformation*.
- studiere bei den *'Anwendungen der MAC Operation'* das →*Bier-Pizza-Blumen-Gleichnis*.
- was ermittelt die MAC-operation eigentlich?

Dic4 WkF=15.Nov MAC in VHDL

. wie macht man aus 'C' ein VHDL:



SMUE-Fragen:

? schreib das FIR-Filter von Peudo-C nach VHDL um
(dann entsteht eine Hardware "MAC-Unit")

Ein Prozessor mit **MAC**-Unit (**M**ultiply **A**nd **C**umulate)
wird "DSP" genannt (Digital Signal Processor).

? studiere bei den *'Anwendungen der MAC Operation'* die →*Fourier-Transformation*.

? studiere bei den *'Anwendungen der MAC Operation'* das →*Bier-Pizza-Blumen-Gleichnis*.

Dic4 Wk8=8.Nov Aliasing-Antialiasing-Oversampling

. Aliasing im Film: Stroboskopeffekt:
Anzahl(Bilder)/s < 2*Anzahl(Ereignisse)/s
(Verletzung des Abtasttheorems ⟹ Aliases
dazu:

. `Nathanaels Link:` optisches Aliasing – die 'fliegenden Wassertropfen':
→ `https://youtu.be/ms-9GtBepNA` (08Nov'22)

. Audio-CD-Digitalisierung: Nyquistfrequenz 22050 Hz, Samplingrate: 44100 Hz, Wortbreite: 16 Bit, Störabstand ('signal-to-noise-ratio'): $2^{16}/_1 = 65536 \equiv 20log(65536) = 96.33dB$. Ein Faktor 2 ist in dB umgerechnet 20log(2)=+6.02dB → 16Bit*6.02dB = 96.32 dB
Zu addieren ist 1.76 dB, der Effektivwert des Quantisierungsfehlers (-'rauschens').

Also landen wir bei 6.02dB * 16 + 1.76 dB = 96.33 + 1.76 dB = 98.09 dB

s. → `https://www.analog.com/media/en/training-seminars/tutorials/MT-001.pdf` 09Nov'22

s. → `https://microchipdeveloper.com/adc:adc-snr` 09Nov'22

s. → `https://www.edn.com/what-does-the-adc-snr-mean/` 09Nov'22

. Filter-Ordnung:
"Die Ordnung eines Filters beschreibt die Verstärkungsabnahme (Dämpfung und Flankensteilheit) von Frequenzen (weit) oberhalb oder unterhalb der jeweiligen Grenzfrequenz des Filters. Sie ist bei Tiefpass- oder Hochpassfilter über der Frequenz etwa n · 6 dB pro Oktave (n · 20 dB pro Dekade), wobei n die Ordnung des Filters darstellt. Für Bandpässe bzw. Bandsperren, welche Kombinationen aus Tiefpass- und Hochpassfiltern darstellen und somit zwei Filterflanken aufweisen, ist die Filterordnung als Funktion der Steilheit der Filterflanke doppelt so hoch:
Filter höherer Ordnung können entweder durch Hintereinanderschaltung von Filtern niedriger (1. und 2. Ordnung) realisiert werden"

(`https://de.wikipedia.org/wiki/Filter_(Elektrotechnik)` 09Nov'22)

Ein CD-Anti-Aliasing-Filter muss alle Signale > 22050Hz (NyquistFrequenz) um 96 dB absenken! Wenn man das im Frequenzbereich 20-22kHz schaffen will, braucht man ein Filter mit (Schlussrechnung):

$$\frac{96dB}{(22-16=6kHz)} == \frac{?dB}{Dekade}$$

$$\frac{96dB}{Faktor\,6/16=0.375} == \frac{?dB}{Faktor 10}$$

$$?=96dB*\frac{10}{0.375}=96*26.7dB=2560dB$$

das entspricht bei 20dB/Dekade einem Filter 128ter Ordnung und gilt als *unrealisierbar*

SMUE-Fragen:

? was hat Aliasing mit dem Stroboskop-Effekt zu tun?
? was ist das Problem der Anti-Aliasing-Filter?
? was ist 'Überabtastung' ('oversampling')?
? wie hilft 'oversampling' beim Antialiasing?
? wie programmiert man ein digitales 'FIR'-Filter in Pseudo-'C'?
? was heißt 'FIR'?

Dic4 Wk4=4.Nov Digitalisierung

. Sampling, Abtastrate: Telefon 4kHz, CD 22.050kHz, Soundkarte 24kHz: 1/44100 Hz = 22.7 us
. ADC *'analog-to-digital converter* liefert pro Wandlung eine 8-Bit Binärzahl (Telefon) oder 16-Bit-Binärzahl (CD), Souncards können 8-Bit-mono, 8-Bit stereo,16-Bit-mono und 16-Bit-stereo Formate wandeln.
. 'DAC': *digital-to-analog converter*

- Nyquistrate = 1/2 Abtastrate (n. Harry Nyquist)
- Aliasing=Phantomsignal:
  Aliasing JScript Simulation links →
  →http://10.10.63.61/SMUE/shr/js/DSP/KSN15/Aliasing/jsAliasXhQbR.html
  →http://www.qsl.net/oe7csj/zuig/jsAliasXhQbR.html
- Soundcard, CD, mp3: Digitalsignal-Quellen
- FIR Filter:

```
output=0;
for(i=0; i<N; i++)   output+= sample[i]*k[i];
```

SMUE-Fragen:
? was ist *sampling*?
? was ist *sampling rate*?
? welche Hardware erfordert sampling?
? wie entsteht *aliasing*?
? wie ist *aliasing* zu vermeiden (anti-aliasing)?
? was ist ein *digitales FIR-Filter*?
? gib den Pseudo-Code eines FIR-Filters an

Dic4 WjL=21Okt Mitarbeitsüberprüfungen Dave+Basti+Ivan+Tobi+Jonas+Sandro+Hans+Elias ++

Wenn wir die Leistungsbeurteilung auf der Mitarbeit aufbauen wollen, so geht das nur für diejenigen gut aus, die entsprechend mitarbeiten, dh.

⊕ Unterrichtsmitschrift
⊕ Mitarbeitsüberprüfungen (mündl. u. schr. 'SMÜ')
⊕ kooperative Erarbeitung der Unterrichtsinhalte

Bei fehlenden Mitarbeitserfolgen muss (s. Vorschriften) der Wissensstand mit angesagten Prüfungen erhoben werden.

§3 "(4) Unbeschadet der Bestimmungen des § 5 Abs. 2 sind zum Zweck der Leistungsbeurteilung über die Leistungsfeststellungen auf Grund der Mitarbeit der Schüler im Unterricht und über die lehrplanmäßig vorgeschriebenen Schularbeiten hinaus nur so viele mündliche und schriftliche Leistungsfeststellungen vorzusehen, wie für eine sichere Leistungsbeurteilung für ein Semester oder für eine Schulstufe unbedingt notwendig sind."
§11. (1) Die Beurteilung der Leistungen der Schüler in den einzelnen Unterrichtsgegenständen hat der Lehrer durch die im §3 Abs. 1 angeführten Formen der Leistungsfeststellung zu gewinnen. Maßstab für die Leistungsbeurteilung sind die Forderungen des Lehrplanes unter Bedachtnahme auf den jeweiligen Stand des Unterrichtes.

Dic4 WjL=21Okt Super-Ego

Die EDV begann mit dem Wunsch, menschliches Denken nachzubilden: Zuerst mit Glockenspielen, Prager *Golem*, Frankensteins Monster, mechanischen, dann elektrischen Rechenmaschinen, Lochkartensortierern, Buchungsmaschinen, Zuse-Rechnern, Elektronenrechnern, von-Neumann-Computern — bis hin zum zeitgemäßen **ANN "artificial neural network"** (künstliches (digitales) Neuronen-Netz).

Das Studium menschlicher Denkprozesse führt so ziemlich direkt zu unserem DIC.

Existenzphilosophen streiten schon Jahrhunderte, ob menschengeschaffene Existenzen jemals den Menschen übertreffen könnten oder eine *Seele* hätten. Von Kollegin SA wurde heute der Begriff *Super-Ego*, das "über-Ich", ins Spiel gebracht: *The 'id', 'ego', and 'super-ego' are ideas created by Sigmund Freud (nicht Einstein ;). They are three concepts used to explain the way the human mind works.*

- *The 'id' (das 'Es') remains unconscious,*
  the 'id' is the set of uncoordinated instinctual trends.
- *The 'ego' (das 'Ich'), and to some extent the 'super-ego', is conscious or on the surface; the 'ego' is the organized realistic part*
- *the 'super-ego' (das 'über-Ich') plays the critical and moralising role.*

The 'id', 'ego' and 'super-ego' are functions of the mind, not parts of the brain. They do not correspond one-to-one with actual structures of the kind dealt with by neuroscience.
(frei nach https://simple.wikipedia.org/wiki/Id,_ego,_and_super-ego
und https://de.wikipedia.org/wiki/Strukturmodell_der_Psyche, 23Okt'22)

Zu meiXhner Schulzeit erklärte man uns im Fach 'Philosophie': Im SuperEgo habe man ein optimistisches, idealisiertes Selbst-Bild, sei intelligent, sportlich, sympathisch, beliebt, könne alles wirklich besser und sei unfehlbar. Nach Wikipedia und SA scheint es eher eine moralische Instanz zu sein.

Dic4 WjB+WjII=11/18 Okt CU in VHDL

```
DBus     : inout  STD_LOGIC_VECTOR(7 downto 0);
ABus     : out    STD_LOGIC_VECTOR(15 downto 0);
clk      : in     STD_LOGIC;
```

```
notWrite : out    STD_LOGIC; --negativ logisch==LO-aktiv
notRead  : out    STD_LOGIC; --negativ logisch==LO-aktiv
```

```
              /* PC=ProgramCounter, InstructionPointer: */
SIGNAL PC: STD_LOGIC_VECTOR (15 downto 0);
              /* von-Neumann-Phasen-Zaehler 'PhaseCTR' 1-5 (0-4) ==3 Bit:*/
SIGNAL PhaseCTR: STD_LOGIC_VECTOR (2 downto 0);  --sind 3 Bit!
              /* 'Data Pointer', auch 'Memory Pointer'
               * == RAM-Adresse fuer Daten-Byte: */
SIGNAL DPointer: STD_LOGIC_VECTOR (15 downto 0);
              /* 'instruction register'
               * speichert den 'instruction code' waehrend der Datenbus
               * ALU-Daten zu uebertragen hat: */
SIGNAL InstReg: STD_LOGIC_VECTOR ( 7 downto 0);
              /* CPU-Rechen-Register A und B fuer arithmetische und
               * logische Operationen wie 'A+B' oder 'A and B': */
SIGNAL CpuRegA,
       CpuRegB: STD_LOGIC_VECTOR ( 7 downto 0);--Rechen-Register
              /* CPU-Rechen-Ergebnis-Register zur Zwischenspeicherung der Ergebnisse
               * bis zur von-Neumann-Phase-Nr5-'Data SAVE' */
SIGNAL ErgReg: STD_LOGIC_VECTOR ( 7 downto 0);
```

```
CU: process(clk) begin
  if rising_edge(clk) then
    PhaseCTR <= PhaseCTR+1;
  end if

  if (PhaseCTR=1) then
    if rising_edge(clk) then
      PC <= PC+1;
      ABus <=  PC;
      notRead <= '0';
    elsif falling_edge(clk) then--delay=1/2 clock cycle
      InstReg <= DBus;
      notRead <= '1';
      ABus <= (others => 'z');
    end if
```

```
  elseif (PhaseCTR=2) then
      --(tue gar nichts==Wartezeit f. InstructionDecoder)
```

```
  elseif (PhaseCTR=3) then
      --'data LOAD'
    if rising_edge(clk) then
      ABus <= DPointer;
      notRead <= '0';
    elsif falling_edge(clk) then
      /* Huch - da fehlen uns ja noch Befehle!
      * -> bitte selber erfinden: */
      case InstReg is
        when InstReg= ... => A <= DBus;
        when InstReg= ... => B <= DBus;
        when InstReg= ... => DPointer <= DBus;
        oderso
      end case;
      notRead <= '1';
      ABus <= (others => 'z');
    end if
```

```
  elseif (PhaseCTR=4) then
      --'instruction EXECUTE'

    if rising_edge(clk) then
      case InstReg is
          when '00000000' => ErgReg <= A  +  1;
          when '00000001' => ErgReg <= B  +  1;
          when '00001000' => ErgReg <= A  -  1;
          when '00001001' => ErgReg <= B  -  1;
          when '00000100' => ErgReg <= A  +  B;
          when '00001100' => ErgReg <= A  -  B;
          when '00010000' => ErgReg <=    -  A;--'CPL'
          when '00010001' => ErgReg <=    -  B;--'CPL'
          when '00101000' => ErgReg <=   not A;
          when '00101001' => ErgReg <=   not B;
          when '00110000' => ErgReg <= A and B;
          when '00111000' => ErgReg <= A or  B;
```

```
            when '....' => A <= B;
            when others => ;--(NOP=no operation)
          end case;
        end if
```

```
      elseif (PhaseCTR=5) then
        --'data SAVE'

      if rising_edge(clk) then
        ABus <= DPointer;
        DBUS <= RegA;
        notWrite <= '0';
      elsif falling_edge(clk) then--1/2 clk delay
        notWrite <= '1';
        DBus <= (others => 'z'); --==alle auf 'z'
        ABus <= (others => 'z');
      end if;
    end if; --PhaseCTR
  end process CU;
```

```
Dic4 Wj7
------------
ALU: Y<=A+B;

 8Bit_DBUS: port( DBUS: inout std_logic_vector ( 7 downto 0); )
16Bit_ABUS: port( ABUS:   out std_logic_vector (15 downto 0); )

48000ksps*200ms=9600samp?

Paznauner-englisch
Terasse

Halbaddierer: XOR-Gate
Volladdierer: XOR mit Carry-in und Carry-out
Carry: Uebertrag
Addierwerk: kaskadierte Volladdierer
Multiplizierwerk: multipliziert Binaerzahlen, zB. 16x16 -> 32Bit

1 MAC =
        N Multiplikationen + (N-1) Additionen
'ksps' = 'kilo samples per second'
'Msps' = 'mega samples per second'

DFT='discrete fourier transform'

Fouriertransform-Algorithmus:

1. digitalisiere das Inputsignal (zB 200ms/48ksps == 9600 samples)
2. FOR (alle Frequenzen)
3.   digitalisiere cos(Frequenz) (9600 samples)
4.   Inputsignal -> MAC <- cos(Frequenz)
5.   digitalisiere sin(Frequenz) (auch 9600 samples)
6.   Inputsignal -> MAC <- sin(Frequenz)
7.   speichere die MAC-Summen fuers Diagramm
8. Ausgabe als Spektrum
```

```
Dic4 WiU
-----------
Prozessor in VHDL: Y<=A+B;
1MB RAM in VHDL: signal RAM: ARRAY(0 TO 1048575) OF std_logic_vector(7 downto 0);
PWM+Tastverhaeltnis
AVR Tiny13 (TN13): 290uA at 1MHz
Analogrechner '1+1=1.99': (slide rule: https://www.sliderules.org/)
Analogrechnermuseum: http://www.analogmuseum.org/deutsch/
13 Minuten konzentrieren
taub is schlimmer als blind
Gedankenlesen
alls, was XH sag, ischa Bledsinn
Schneeschaufeln und Leut umbringen sind "uberfl"ussige Arbeit
EngelAloisius: https://www.youtube.com/watch?v=VvdEgkqei6c
schnurren
```

Teil XII
Anhang

42 VHDL Praxis in HWE

43 Programmierung width

43.1 VHDL-Code

```
  ----------------------------------------------------------------------
  -- Kommentarschreim tui grundsaetzli nit, soa Bledsinn.
  -- Undscho gornit, wennma so insistierend drauf hingwiesen wird,
  -- undscho dreimal nid fyr den ollen XH, den Gschaftlhuaber.
  -- geahtn Shas un, va wem des is und vu wann;
  -- wers nit sieht, was des is und wiamas braucht
  -- is eh die volle Pfeifn
  -- und tuat gscheiter was unders, zB TV schaun oder notenausrechna
  ----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

signal SWn     : BIT_VECTOR (3 downto 0);   -- 4 Input Lines
signal DISPLAY : BIT_VECTOR (6 downto 0);   -- die 7 LED Segmente

architecture main of VHDL_7Segment is
begin
        SWn <= not SW;          -- 'SW' is ... aehm ... brauxt ned wissn
        LED <= SWn;             -- 'LED' is, aaah, ''Low Education Decade''

        with (SWn) select DISPLAY <=
                "0000001" when "0000",
                "1001111" when "0001",
                "0010010" when "0010",
                "0000110" when "0011",
                "1001100" when "0100",
                "0100100" when "0101",
                "0100000" when "0110",
                "0001111" when "0111",
                "0000000" when "1000",
                "0000100" when "1001",
                "0001000" when "1010",
                "1100000" when "1011",
                "1110010" when "1100",
                "1000010" when "1101",
                "0110000" when "1110",
                "0111000" when "1111";

        Segment_a <= DISPLAY(6);
        Segment_b <= DISPLAY(5);
        Segment_c <= DISPLAY(4);
        Segment_d <= DISPLAY(3);
        Segment_e <= DISPLAY(2);
        Segment_f <= DISPLAY(1);
        Segment_g <= DISPLAY(0);
end architecture
```

43.2 Pinbelegung

```
FPGA Pin & Beschreibung:
------------------------

  PIN_E11:  Digit 'a'        PIN_J6:  Switch 0
  PIN_F11:  Digit 'b'        PIN_H5:  Switch 1
  PIN_H12:  Digit 'c'        PIN_H6:  Switch 2
  PIN_H13:  Digit 'd'        PIN_G4:  Switch 3
  PIN_G12:  Digit 'e'
  PIN_F12:  Digit 'f'
  PIN_F13:  Digit 'g'
```

44 VHDL CODE woldi

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY SevenSEG IS
port (
  bcd      : in std_logic_vector (3 downto 0);  --BCD in
  segment7 : out std_logic_vector (6 downto 0)  --Decoded out
);
END SevenSEG;


ARCHITECTURE Behavioral OF SevenSEG IS BEGIN
  process (bcd) BEGIN
    case bcd is
      when "0000" => segment7 <= "0000001"; -- '0'
      when "0001" => segment7 <= "1001111"; -- '1'
      when "0010" => segment7 <= "0010010"; -- '2'
      when "0011" => segment7 <= "0000110"; -- '3'
      when "0100" => segment7 <= "1001100"; -- '4'
      when "0101" => segment7 <= "0100100"; -- '5'
      when "0110" => segment7 <= "0100000"; -- '6'
      when "0111" => segment7 <= "0001111"; -- '7'
      when "1000" => segment7 <= "0000000"; -- '8'
      when "1001" => segment7 <= "0000100"; -- '9'
      when "1010" => segment7 <= "0001000"; -- 'A'
      when "1011" => segment7 <= "1100000"; -- 'B'
      when "1100" => segment7 <= "0110001"; -- 'C'
      when "1101" => segment7 <= "1000010"; -- 'D'
      when "1110" => segment7 <= "0110000"; -- 'E'
      when "1111" => segment7 <= "0111000"; -- 'F'
    end case;
  end process;
END Behavioral;
```

45 VHDL CODE (7 segment decoder) mikno

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SegmentAnzeige IS
PORT (
  x0,x1,x2,x3          : IN STD_LOGIC ;
  d0,d1,d2,d3,d4,d5,d6 : OUT STD_LOGIC );
END SegmentAnzeige ;

ARCHITECTURE LogicFunction OF SegmentAnzeige IS BEGIN
  d0 <= NOT (( NOT x2 AND NOT x0 )  OR (NOT x3 AND x1 )
         OR (NOT x3 AND x2 AND x0 ) OR ( x2 AND x1 )
         OR (x3 AND NOT x2 AND NOT x1 ) OR (x3 AND NOT x0 ));
  d1 <= NOT ((NOT x3 AND NOT x2) OR (NOT x3 AND NOT x1 AND NOT x0 )
         OR (NOT x2 AND NOT x0)  OR (NOT x3 AND x1 AND x0 )
         OR (x3 AND NOT x1 AND x0));
  d2 <= NOT ((NOT x3 AND NOT x1) OR (NOT x3 AND x0) OR (NOT x1 AND x0)
         OR (NOT x3 AND x2 ) OR ( x3 AND NOT x2 ));
  d3 <= NOT ((NOT x3 AND NOT x2 AND NOT x0) OR (NOT x2 AND x1 AND x0)
         OR (x2 AND NOT x1 AND x0) OR (x2 AND x1 AND NOT x0)
         OR (x3 AND NOT x1));
  d4 <= NOT ((NOT x2 AND NOT x0) OR (x1 AND NOT x0) OR (x3 AND x1 )
         OR (x3 AND x2 ));
  d5 <= NOT ((NOT x1 AND NOT x0 ) OR (NOT x3 AND x2 AND NOT x1)
         OR (x2 AND NOT x0)  OR (x3 AND NOT x2) OR (x3 AND x1));
  d6 <= NOT ((NOT x2 AND x1) OR (x1 AND NOT x0)
         OR (NOT x3 AND x2 AND NOT x1)
         OR (x3 AND NOT x2) OR (x3 AND x0 ));
END LogicFunction;
)
```

46 Generated VHDL Code from Altium(to simulate)

```
-------------------------------------------------------------
-- VHDL Testbench for fpga_project
-- 2016 2 16 8 47 7
-- Created by "EditVHDL"
-- "Copyright(c) 2002 AltiumLimited"
-------------------------------------------------------------
Library IEEE ;
Use IEEE.std_logic_1164.all;
Use IEEE.std_logic_textio.all;
Use STD.textio.all;
-------------------------------------------------------------

entity Testfpga_project is
end Testfpga_project ;
-------------------------------------------------------------

architecture stimulus of Testfpga_project is
file RESULTS : TEXT open WRITE_MODE is "results.txt";
procedure WRITE_RESULTS(
  INPUT1 : std_logic;
  INPUT2 : std_logic;
  OUTPUT : std_logic
) is
variable l_out : line;
begin
  write(l_out, now, right, 15) ;
  write(l_out, INPUT1, right, 2) ;
  write(l_out, INPUT2, right, 2) ;
  write(l_out, OUTPUT, right, 2) ;
  writeline(RESULTS, l_out);
end procedure ;
component fpga_project
port(
  INPUT1 : in std_logic;
  INPUT2 : in std_logic;
  OUTPUT : out std_logic
```

```
);
end component;

  signal INPUT1 :
  signal INPUT2 :
  signal OUTPUT :std_logic

begin
  DUT: fpga_project port map (
    INPUT1 => INPUT1 ,
    INPUT2 => INPUT2 ,
    OUTPUT => OUTPUT
  );

  STIMULUS0 : process
  begin
    -- insert stimulus here
    INPUT1 <= '0' ;
    INPUT2 <= '0' ;
    wait for 20ns ;

    INPUT1 <= '1' ;
    INPUT2 <= '0' ;
    wait for 20ns ;

    INPUT1 <= '0' ;
    INPUT2 <= '1' ;
    wait for 20ns ;

    INPUT1 <= '1' ;
    INPUT2 <= '1' ;
    wait for 20ns ;
    wait ;
  end process ;

  WRITE_RESULTS(INPUT1, INPUT2, OUTPUT);
end architecture;
------------------------------------------------------------
```

47 Der VHDL-Code in Quartus wurde dadurch erstellt - nirra

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SSegmentAnzeige IS
  PORT (
    s0,s1,s2,s3          : IN STD_LOGIC;
    d0,d1,d2,d3,d4,d5,d6 : OUT STD_LOGIC
  );
END SSEgmentAnzeige;

ARCHITECTURE LogicFunction OF SSEgmentAnzeige IS BEGIN
  d0 <= NOT((NOT s2 AND NOT s0) OR (NOT s3 AND s1 )
         OR (NOT s3 AND s2 AND s0) OR (s2 AND s1)
         OR (s3 AND NOT s2 AND NOT s1 ) OR (s3 AND NOT s0));
  d1 <= NOT((NOT s3 AND NOT s2)
         OR (NOT s3 AND NOT s1 AND NOT s0) OR (NOT s2 AND NOT s0)
         OR (NOT s3 AND s1 AND s0) OR (s3 AND NOT s1 AND s0));
  d2 <= NOT((NOT s3 AND NOT s1) OR (NOT s3 AND s0) OR (NOT s1 AND s0)
         OR (NOT s3 AND s2) OR (s3 AND NOT s2));
  d3 <= NOT((NOT s3 AND NOT s2 AND NOT s0) OR (NOT s2 AND s1 AND s0)
         OR (s2 AND NOT s1 AND s0) OR (s2 AND s1 AND NOT s0)
         OR (s3 AND NOT s1 AND NOT s0));
  d4 <= NOT((NOT s2 AND NOT s0) OR (s1 AND NOT s0) OR (s3 AND s1)
         OR (s3 AND s2));
  d5 <= NOT((NOT s1 AND NOT s0) OR (NOT s3 AND s2 AND NOT s1)
         OR (s2 AND NOT s0) OR (s3 AND NOT s2) OR (s3 AND s1));
  d6 <= NOT((NOT s2 AND s1) OR (s1 AND NOT s0)
         OR (NOT s3 AND s2 AND NOT s1)
         OR (s3 AND NOT s2) OR (s3 AND s0));
END LogicFunction;
```

47.1 DE0 Board

Chapter 2
Altera DE0 Board

This chapter presents the features and design characteristics of the DE0 board.

2.1 Layout and Components

A photograph of the DE0 board is shown in Figure 2-1. It depicts the layout of the board and indicates the location of the connectors and key components.

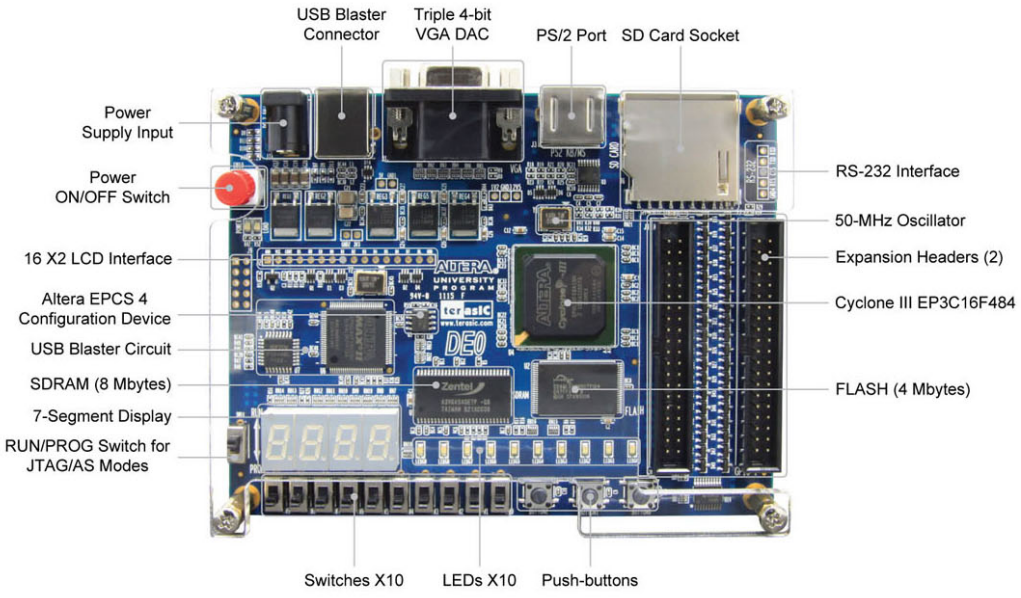


Figure 2-1 The DE0 board.

The DE0 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware is provided on the DE0 board:

- ®Altera Cyclone III 3C16 FPGA device
- Altera Serial Configuration device – EPCS4
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory
- SD Card socket
- 3 pushbutton switches
- 10 toggle switches
- 10 green user LEDs
- 50-MHz oscillator for clock sources
- VGA DAC (4-bit resistor network) with VGA-out connector
- RS-232 transceiver
- PS/2 mouse/keyboard connector
- Two 40-pin Expansion Headers

2.2 Block Diagram of the DE0 Board
Figure 2-2 gives the block diagram of the DE0 board. To provide maximum flexibility for the user, all connections are made through the Cyclone IIII FPGA device. Thus, the user can configure the FPGA to implement any system design.

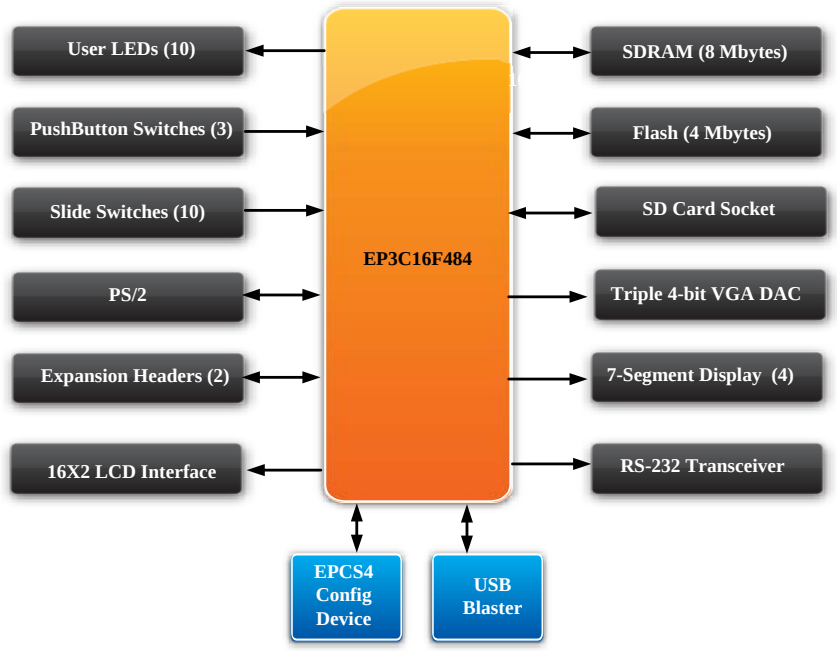


Figure 2-2 Block diagram of the DE0 board.

Following is more detailed information about the blocks in Figure 2-2:

Cyclone IIII 3C16 FPGA

- 15,408 LEs
- 56 M9K Embedded Memory Blocks
- 504K total RAM bits
- 56 embedded multipliers
- 4 PLLs
- 346 user I/O pins
- FineLine BGA 484-pin package

Built-in USB Blaster circuit

- On-board USB Blaster for programming and user API (Application programming interface) control
- Using the Altera EPM240 CPLD

SDRAM

- One **8**-Mbyte Single Data Rate Synchronous Dynamic RAM memory chip
- Supports 16-bits data bus

Flash memory

- 4-Mbyte NOR Flash memory
- Support Byte (8-bits)/Word (16-bits) mode

SD card socket

- Provides both SPI and SD 1-bit mod SD Card access

Pushbutton switches

- 3 pushbutton switches
- Normally high; generates one active-low pulse when the switch is pressed

Slide switches

- 10 Slide switches
- A switch causes logic 0 when in the DOWN position and logic 1 when in the UP position

General User Interfaces

- 10 Green color LEDs (Active high)
- 4 seven-segment displays (Active low)
- 16x2 LCD Interface (Not include LCD module)

Clock inputs

- 50-MHz oscillator

VGA output

- Uses a 4-bit resistor-network DAC
- With 15-pin high-density D-sub connector
- Supports up to 1280x1024 at 60-Hz refresh rate

Serial ports

- One RS-232 port (Without DB-9 serial connector)
- One PS/2 port (Can be used through a PS/2 Y Cable to allow you to connect a keyboard and mouse to one port)

Two 40-pin expansion headers

- 72 Cyclone III I/O pins, as well as 8 power and ground lines, are brought out to two 40-pin expansion connectors
- 40-pin header is designed to accept a standard 40-pin ribbon cable used for IDE hard drives

47.2 DE0 Board mit QuartusII + Altium

s.YHs PDF

47.3 DE0 + Altium Übung.1: 7-segment-decoder

47.3.1 Truth Tables

47.3.2 KV Diagrams

47.3.3 Boolean Equations

47.3.4 Logic Gate Circuit Schematic

47.3.5 VHDL Program

47.4 DE0 + Altium Übung.2: ALU + Control Unit

47.5 VHDL erprobte Aufgabe 'DecisDecoder' /Grado

Listing 77: 'Decis Prozessor VHDL'

```
library ieee;
use ieee.std_logic_1164.all;

entity DE0Board is
    port(
        sw            :    in  std_logic_vector(9 downto 0);
        led       :    out std_logic_vector(9 downto 0);
                  hex0                  :         out std_logic_vector(6 downto 0);
                  key                           :       in std_logic_vector(2 downto
                      0)
    );
end DE0Board;

architecture behavior of DE0Board is
    -- Eingˆ/nge --
    signal instruction     :    std_logic_vector(7 downto 0);
    signal sr_result        :    std_logic;
    -- Ausgˆ/nge --
    signal alu_en            :    std_logic;
    signal alu_sel           :    std_logic_vector(2 downto 0);
    signal rf_a            :    std_logic_vector(1 downto 0);
    signal rf_b            :    std_logic_vector(1 downto 0);
    signal rf_q            :    std_logic_vector(1 downto 0);
         signal rf_q2                           :         std_logic_vector(1 downto 0)
             ;
    signal rf_wr           :    std_logic;
    signal sr_cond           :    std_logic_vector(2 downto 0);
    signal pc_skip           :    std_logic;
    signal bus_imm           :    std_logic_vector(3 downto 0);
    signal bus_imm_hl        :    std_logic;
    signal pher_en           :    std_logic;
    signal pher_wr           :    std_logic;
         signal status                                  : std_logic_vector(3 downto 0)
             ;
         signal clear                                   : std_logic;
begin

    -- Instanz der entity InstructionDecoder
   InstrDec : entity work.InstructionDecoder port map(
                                    -- Eingˆ/nge --
                                    instruction => instruction,
                                    sr_result     => '0',
                                    -- Ausgˆ/nge --
                                    alu_en        => alu_en,
                                    alu_sel         => alu_sel,
                                    rf_a            => rf_a,
                                    rf_b            => rf_b,
                                    rf_q            => rf_q,
                                    rf_wr            => rf_wr,
                                    sr_cond        => sr_cond,
                                    pc_skip        => pc_skip,
                                    bus_imm        => bus_imm,
                                    bus_imm_hl    => bus_imm_hl,
                                    pher_en        => pher_en,
                                    pher_wr        => pher_wr
                                );

        Alu : entity work.alu port map(

                        -- Eingˆ/nge --
                        rf_a => rf_a,
                        rf_b => rf_b,
                        instruction => instruction(6 downto 4),
                        -- Ausgˆ/nge --
                        rf_q => rf_q2
        );

        StatusRegister: entity work.status_register port map(
```

```
                        -- Eingˆ/nge --
                        rf_a => rf_a,
                        rf_b => rf_b,
                        instruction => instruction(6 downto 4),
                        clear => clear,
                        -- Ausgˆ/nge --
                        status => status
        );

    -- Zuweisung der Schalter & LEDs --
    -- Eingˆ/nge
    instruction          <= sw(7 downto 0);
    sr_result            <= sw(8);
         clear                                  <= key(2);

process(sw) begin
    -- Ausgˆ/nge
         case status is
                        when "0000" => hex0 <= "1111111"; -- Clear
                        when "0001" => hex0 <= "1111001"; -- í Û 1 í Û  -- carry
                        when "0010" => hex0 <= "0100100"; -- í Û 2 í Û  -- equal
                        when "0100" => hex0 <= "0110000"; -- í Û 3 í Û  -- greater

                        when "1000" => hex0 <= "0011001"; -- í Û 4 í Û  -- less
                        when others => hex0 <= "1111111";
         end case ;

         if sw(9) = '0' then

                if(alu_en = '1') then
                        led(1 downto 0) <= rf_q2;
                else
                        led(1 downto 0) <= rf_q;
                end if;

                led(3 downto 2) <= rf_b;
                led(4) <= rf_wr;
                led(5) <= alu_en;
                led(8 downto 6) <= alu_sel;
                led(9) <= pc_skip;

         elsif(sw(9) = '1') then

                led(9 downto 6) <= bus_imm;
                led(5) <= bus_imm_hl;
                led(4) <= pher_en;
                led(3) <= pher_wr;
                led(2 downto 0) <= sr_cond;

         else
                led(9 downto 0) <= "0000000000";
         end if;
end process;
end behavior;
```

Listing 78: 'Decis Control Unit VHDL'

```
library ieee;
use ieee.std_logic_1164.all;

entity InstructionDecoder is
    port(
            instruction    :        in  std_logic_vector(7 downto 0);

            alu_en         :        out std_logic;
            alu_sel        :        out std_logic_vector(2 downto 0);

            rf_a           :        out std_logic_vector(1 downto 0);
            rf_b           :        out std_logic_vector(1 downto 0);
            rf_q           :        out std_logic_vector(1 downto 0);
            rf_wr          :        out std_logic;

            sr_cond        :        out    std_logic_vector(2 downto 0);
            sr_result      :        in  std_logic;
            pc_skip        :        out    std_logic;

            bus_imm        :        out    std_logic_vector(3 downto 0);
            bus_imm_hl     :        out    std_logic;

            pher_en        :        out    std_logic;
            pher_wr        :        out    std_logic
    );
end InstructionDecoder;

architecture behavior of InstructionDecoder is
    -- Buffer fÂ´...r die rf_q <= rf_b zuweisung.
    signal rf_q_t : std_logic_vector(1 downto 0);
begin
    decode : process  (instruction, sr_result)
    begin
        if(instruction(7) = '0') then    -- ALU --
            alu_sel     <= instruction(6 downto 4);
            alu_en      <= '1';
            rf_q_t      <= instruction(1 downto 0);
            rf_b        <= instruction(3 downto 2);
            rf_wr       <= '1';
            -- not used
            sr_cond        <= "000";
            pc_skip        <= '0';
            pher_en        <= '0';
            pher_wr     <= '0';
            bus_imm     <= "0000";
            bus_imm_hl    <= '0';

        elsif(instruction(7) = '1' and instruction(6) = '0') then
            alu_en      <= '0';
            rf_q_t      <= instruction(1 downto 0);
            rf_b        <= instruction(3 downto 2);
            sr_cond        <= "000";
            pc_skip        <= '0';
            pher_en        <= '1';
            bus_imm     <= "0000";
            bus_imm_hl    <= '0';
                               rf_wr                    <= instruction(4);
                               pher_wr                          <= instruction(4);

                        elsif(instruction(7) = '1' and instruction(6) = '1' and
                            instruction(5) = '0') then
            alu_en      <= '0';
            rf_q_t      <= "00";
            rf_b        <= "00";
            rf_wr       <= '1';
            sr_cond        <= "000";
            pc_skip        <= '0';
            pher_en        <= '0';
            pher_wr     <= '0';
            bus_imm     <= instruction(3 downto 0);
                               bus_imm_hl       <= instruction(4);
```

```
                        elsif(instruction(7) = '1' and instruction(6) = '1' and
                            instruction(5) = '1' and instruction(4) = '0' and
                            instruction(3) = '0') then
                                alu_en      <= '0';
            rf_q_t      <= "00";
            rf_b        <= "00";
            rf_wr       <= '1';
            sr_cond        <= instruction(2 downto 0);
            pher_en        <= '0';
            pher_wr     <= '0';
            bus_imm     <= "0000";
                                bus_imm_hl      <= '0';
                                pc_skip <= sr_result xor instruction(3);

                end if;

        rf_q        <= rf_q_t;
        rf_a        <= rf_q_t;

    end process;

end behavior;
```

Listing 79: 'Decis Alu VHDL'

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity alu is
        port(
                rf_a, rf_b: in std_logic_vector(1 downto 0);
                instruction: in std_logic_vector(2 downto 0);
                rf_q: out std_logic_vector(1 downto 0)
        );
end alu;

architecture beh_alu of alu is
begin
        with instruction select
                rf_q <=         rf_a                                    when "000",
                                not rf_a                                when "001",
                                rf_a and rf_b           when "010",
                                rf_a or rf_b            when "011",
                                rf_a xor rf_b           when "100",
                                rf_a + rf_b             when "101",
                                rf_a + "01"                             when "110",
                                "ZZ"                                            when
                                    others;

end beh_alu;
```

Listing 80: 'Decis Registerfile VHDL'

```
library ieee;
use ieee.std_logic_1164.all;

entity status_register is
port(
        instruction: in std_logic_vector(2 downto 0);
        rf_a, rf_b: in std_logic_vector(1 downto 0);
        clear:  in std_logic;
        status: out std_logic_vector(3 downto 0)
        );
end status_register;

architecture beh_status_register of status_register is
signal var1: std_logic;
begin

        var1 <= ((rf_a(1) xor rf_b(1)) and (rf_a(0) and rf_b(0))) or (rf_a(1) and rf_b
            (1));

        process(instruction, rf_a, rf_b, clear)
        begin
                if(clear = '0') then
                        status <= "0000";
                elsif(clear = '1') then

                        if(instruction = "111") then

                                if(rf_a = rf_b) then
                                        status <= "0010";
                                elsif(rf_a > rf_b) then
                                        status <= "0100";
                                elsif(rf_a < rf_b) then
                                        status <= "1000";
                                end if;

                        elsif(instruction = "101") then

                                if( var1 = '1') then
                                        status <= "0001";
                                end if;

                        end if;
                end if;
        end process;
end beh_status_register;
```

48 LSV Dic5 YU+ZI 2018

Lehrplan semestriert:

Bereich	Lehrstoff	Bildungs- und Lehraufgabe
9.*Semester – – – Kompetenzmodul*9 :		
Entwurf digitaler Systeme	Realisierung eines digitalen Systems ausgehend von einer Verhaltensbeschreibung.	-komplexe digitale Systeme in Form eines Top-Down-Design entwickeln. (→ z.B.SoftCore - uC Implementierung in FPGA?) (Michael)
Computerarchitekture	Anwendungsspezifische Prozessorarchitekturen. (→ z.B.: Prozessoren für Video/Audio – Anwendungen, Verschlüsselung usw, lexikalisch)	-Architekturen zur Optimierung der Leistungsfähigkeit spezieller Anwendungen beschreiben und deren Funktionsweise erklären.
Embedded Systems	~~Betriebssysteme für Embedded Systems~~, (→ FSST) Echtzeitverarbeitung.	-die Aufgaben und Funktionsweise von Echtzeitbetriebssystemen (→ RTOS) erklären.
Signalverarbeitung	Ausgewählte Beispiele der digitalen Signalverarbeitung und deren Realisierung mittels Signalprozessoren. (→ Theorie)	-Signalprozessor-Architekturen erklären und Basisalgorithmen der digitalen Signalverarbeitung implementieren. (→ kein Signalprozessor, sondern eher z.B. LABVIEW)
10.*Semester – – – Kompetenzmodul*10 :		
Entwurf digitaler Systeme	-Analysetools.	-komplexe digitale Systeme mit Hilfe von Softwaretools optimieren.
Computerarchitekture	-Anwendungsspezifische Prozessorarchitekturen.	-Architekturen für spezielle Anwendungen anhand einer Spezifikation auswählen.
Embedded Systems	-Echtzeitbetriebssysteme.	-ein Echtzeitbetriebssystem für spezielle Anwendungen konfigurieren.
Signalverarbeitung	-Analyse-und Testverfahren.	-Implementierungen von Algorithmen analysieren und testen.

Koordinatoren: **YU+ZI** Imsterbg.2018

Sep:	**Digitale Systeme**	Realisierung eines digitalen Systems ausgehend von einer Verhaltensbeschreibung
Okt:	**Computer Architekturen**	Cortex - PSOC
Nov:	**Computer Architekturen**	Cortex - PSOC
Dez:	**Computer Architekturen -Digitale Signalverarbeitung**	Cortex - PSOC - digitale Signale im Zeit - und Frequenzbereich - Grundoperationen der digitalen Signalverarbeitung
Jan:	**Digitale Signalverarbeitung**	Grundoperationen der digitalen **Signalverarbeitung**
Feb:	**Digitale Signalverarbeitung**	Lineare und Zeitinvariante zeitdiskrete Systeme und deren Beschreibung im Zeit- und Frequenzbereich
Feb:	**Digitale Systeme**	Analysetools
Mar:	**Computer Architekturen**	Anwendungsspezifische Prozessorarchitekturen
Apr:	**Digitale Signalverarbeitung**	Analyse- und Testverfahren

≠ **embedded Systems ??**

XH LSVtlg. Dic5a 22/23:

Auf der Suche nach umsetzbarem Ersatz für die Worthülsen des Lehrplanquargels plane iXH:

Sep: **DSys**
- Prozessor in VHDL

Okt-Jan: **DSP**

"DSP-Grund-OP"
- MAC

"Lineare und Zeitinvariante zeitdiskrete Systeme und deren Beschreibung im Zeit- und Frequenzbereich"
- Fourier
- FIR+IIR
- z-Trf.
- Filterschemata, -charkteristiken
- Digitalisierung
- ANN, Grammatik, Prolog, Fuzzylogic

Feb-Mar: **Comp.Arch.**

"Architekturen"
- von-Neumann in VHDL
- Harvard in VHDL
- DSP in VHDL
- MAC-Unit in VHDL
- ADC+DAC
- Port, Timer, Interrupt
- ISR Standartmuster: retten-verarbeiten-wiederherstellen-RETI
- Timer@uC, Timer@RasPi (Os-Timer)
- Usb

Mar-Apr: **DSys**

"Analysetools:"
- Logic Analyzer
- Boundary Scan + JTAG
- Testen, Testabdeckung
- Simulation

KU: **Comp.Arch.**
- PSOC - KU

'RTOS' zu FSST verschoben: Imsterberg'18

KU: **Comp.Arch.**
- PSOC - KU

Ue: KU

embeddedSystems "Realisierung eines digitalen Systems ausgehend von einer Verhaltensbeschreibung"

48.1 Eingangstest Dic5a-Sep'22

1-	Digital Prozes-sor	skizziere und schreib das externe Bus-Interface eines 8-Bit-Prozessors mit 1 MB max. Memory in VHDL
2-	VHDL	schreib eine 8-Bit-ALU in VHDL schreib in VHDL: Program Counter, Instruction Pointer, Data Pointer, RegA, RegB, Instruction Register, Phasenzähler, Instruction Decoder schreib in VHDL: Eine 24-Bit MAC Unit
3-	Fourier Trans-form	programmiere eine diskrete Fouriertransformation (DFT) in 'C': sampling rate (SR) fs= 22050 ksps resolution res= 16 Bit frequency range rng= 50Hz...4kHz Frequenzauflösung = 50Hz
4-	FIR Filter	programmiere ein FIR Filter mit 1000 Samples und 1000 Koeffizienten in 'C'
5-	PSOC	Beschreibe alle Arbeitsschritte und Programmcodestücke einer temperaturabhängigen PWM-Lüfter-steuerung mit Cypress PSOC5LP
für fehlende Angaben sind sinnvolle Annahmen zu treffen		

49 LSV Dic4 YU+ZI 2018

Lehrplan semestriert:

Bereich	Lehrstoff	Bildungs- und Lehraufgabe
7.Semester – – – Kompetenzmodul7 :		
Entwurf digitaler Systeme	Spezifikation und Darstellung von **digitalen Systemen.**	-den Unterschied zwischen einer Struktur-und einer Verhaltensbeschreibung erklären.
Computerarchitekturen	**Peripheriekomponenten**. (→ADC(DAC/Timer/UART/SPI/dann USB statt UART)	-die Funktionsweise von I/O-Komponenten (→Standard-Peripherie eines uC, eventuell USB?) erklären.
Embedded Systems	Entwicklung von **Mikrocontrollerprogrammen**, Interrupt-Service-Routinen. (→geteilter Unterricht)	-Programme für einen Mikrocontroller in einer höheren Programmiersprache entwickeln, testen und dokumentieren. (→geteilter Unterricht)
Signalverarbeitung	Digitale Signale im **Zeit-und Frequenzbereich** (→ z-Transformation ?), Grundoperatoren(→ MACausreichend ?) der digitalen Signalverarbeitung.	-digitale Signale analysieren und Basisoperationen der digitalen **Signalverarbeitung** anwenden. (→ + Zahlen 6.Semester – Flieskommadarstellung)
8.Semester – – – Kompetenzmodul8 :		
Entwurf digitaler Systeme	(→FPGA) Partitionierung **digitaler Systeme**, Anwendung von Komponenten und Bibliotheksmodulen. (→**digitale Filter FIR**..)	-das Prinzip des hierarchischen Systementwurfs anwenden.
Computerarchitekturen	Prozessorschnittstellen. (→Speichererweiterung/ DMA/ Memoryinterface...)	-das Hardware-Software-Interface Standardschnittstellen erklären.
Embedded Systems	Entwicklung von Mikrocontroller-Programmen mit Peripheriefunktionen. (→ Übermittlung)	-Interface-Programme für einen Mikrocontroller erstellen und testen. (→geteilter Unterricht z.B.: Datenübertragung, PWM-Ausgabe, Einlesen Drehdekoder...)
Signalverarbeitung	Linear-zeitinvariante zeitdiskrete Systeme und deren Beschreibung im Zeit-und Frequenzbereich.	-digitale Systeme zur Signalerzeugung und Filterung beschreiben und deren Funktionsweise erklären.
Realisierung und Test von Systemen	—	—

Koordinatoren: **YU+ZI** Imsterbg.2018

Sep:	Digitale Systeme	Spezifikation und Darstellung **digitaler Systeme**
Okt:	Digitale Systeme	Spezifikation und Darstellung **digitaler Systeme**-Hardwarebasis
Nov:	Computer Architekturen	**Peripheriekomponenten** Standardperipherie: ADC(-DAC)-Timer-UART-SPI
Dez:	Embedded Systems	Entwicklung von **Mikrocontroller**-Programmen, Interruptserviceroutinen
Jan:	Digitale Signalverarbeitung	digitale **Signale** im Zeit- und Frequenzbereich
Feb:	Digitale Signalverarbeitung	Grundoperationen der digitalen **Signalverarbeitung**
Feb:	Digitale Systeme	Partitionierung **digitaler Systeme**, Omponenten und Bibliotheksmodulen (Originaltext!)
Mar:	Digitale Systeme	Anwendung von Komponenten und Bibliotheksmodulen
Apr:	Computer Architekturen	**Prozessorschnittstellen**
Mai:	Computer Architekturen - Embedded Systems	Prozessorschnittstellen - Interface-Programme für einen **Mikrocontroller** erstellen und testen
Jun:	Digitale Signalverarbeitung	Linear-zeitinvariante zeitdiskrete Systeme und deren Beschreibung im **Zeit-und Frequenzbereich**.

XH LSVtlg. Dic4a 21/22:

Auf der Suche nach umsetzbarem Ersatz für die Worthülsen des Lehrplanquargels plane iXH:

Sep-Okt: DSP
- ☐ Signal ↔ Spektrum
- ☐ Auflösung+Sampling, Nyqvist/Shannon, S&H, Over-/Under-/Re-sampling
- ☐ Codierung,
- ☐ Datenkompression,
- ☐ Encryption, Sec'Ty,
- ☐ Convolution-DFT/FFT(Fourier)-SinCosTrf

Nov: Emb. Syst.
- ☐ ISR Standartmuster: retten-verarbeiten-wiederherstellen-RETI
- ☐ Timer@uC, Timer@RasPi (Os-Timer)
- ☐ connect-accept-read-write
- ☐ UART: uC, SPI:RasPi, IIC: uC

Dez: Comp.Arch. Peripherie
- ☐ ADC/DAC:Formen, Sound, DigitalRegler
- ☐ UART, SPI, IIC, Usb

Jan-Feb1: dig.Sys
- ☐ Logic Specs (TTL, CMOS, 3V3, 1V8, LVDS, ECL), Glitch, Spike
- ☐ Dimensionierung+Interfacing (Gatterschaltung, Pegelconversion)
- ☐ diff.Sig, LVDS
- ☐ PWM
- ☐ UML-SequenzDG

Feb2-Mar: DSP
- ☐ FIR-Filter (FOR Schleife $\sum_i^N \text{signal}[t] * \text{coefficient}[t]/_{\text{Samplingrate}}$)
- ☐ dig.TPF+HPF m. JS-Demo-PGM

Mar-Apr: Emb. Syst.
- ☐ uC-PGMG in 'C'
- ☐ pSOC-PGMG in 'C'

Mai: Comp.Arch.
- ☐ A-Bus, +MMU-mit-VMM,
- ☐ D-Bus, +DMA,
- ☐ C-Bus, +Intr.Ctrl,
- ☐ EDO, DDR, BurstMode, eMMC, udgl.
- ☐ HD, SSD, LMS, DVD

Jun: dig.Sys.
- ☐ Partitionierung digitaler Systeme, Omponenten und Bibliotheksmodulen (Originaltext!) → was?
- ☐ Anwendung von Komponenten und Bibliotheksmodulen → tamma ja immer!

49.1 Eingangstest Dic4a-Sep'22

1-	Logik-schaltungen	Was sind Latches und Register? Was kann ein DFF (D-FlipFlop) speichern? Was ist ein 'Hex Inverter'? Was ist ein Datenselektor (zB.CD4067) ? Was ist ein Schmitt-Trigger-NAND? Was ist ein Schaltnetz? Erkläre den Entwurfsprozess. Was ist ein (endlicher) 'Automat'? Eine 'finite-state-machine'? % ein Schaltwerk? Entwurfsprozess? Programmiere einen Schaltwerk-Simulator
2-	Digital Prozes-sor	Woraus besteht ein Prozessor? Woraus besteht ein Computer? Nenne die von-Neumann-Phasen der Programmausführung. Was kennzeichnet von-Neumann-, Harvard- und DSP-Architekturen? Programmiere einen Prozessor-Simulator
3-	VHDL	schreib eine ALU in VHDL schreib eine CU in VHDL schreib einen Zähler in VHDL
4-	Program-mierung	programmiere einen ASCII-zu-Morsecode-Wandler in 'C' programmiere ein ANN in 'C' schreib ein 'C'-Programm, das 'Rauschen' erzeugt programmiere die N-dimensionale Skalar-Multiplikation (MAC) der Vektoren a und b in 'C'

5-	Zahlen-dar-stellung	was machen "-(~x)"? und "~(-x)"?
		was steht nach
		`int cEOF=-1;`
		`unsigned char c1= cEOF;`
		`int c2=c1;`
		in der Variablen 'c2' für eine Zahl (Bitkombination)?
		Was sind Einerkomplement und Zweierkomplement>
		Was ist eine normalisierte, binäre Fließkommazahl?
		Was ist eine gepackte (packed) BCD-Zahl?
		Wie wird die Zahl 47 (in 'C') als ASCII-String gespeichert?
		Was kannst Du in 'Dic'?

für fehlende Angaben sind sinnvolle Annahmen zu treffen		

50 Dic4-17Sep'15 sem./Pe4

3. DIGITALE SYSTEME UND COMPUTERSYSTEME semestriert

BGBl. II - Ausgegeben am 17. September 2015 - Nr. 262 (cloud.htlinn.ac.at @Ui8)

III. Jahrgang:

Bereich	Bildungs- und Lehraufgabe	Lehrstoff	KUYU
7.Semester – Kompetenzmodul 7:			
Entwurf digitaler Systeme	- den Unterschied zwischen einer Struktur- und einer Verhaltensbeschreibungssprache erklären.	Spezifikation und Darstellung von digitalen Systemen.	
Computer architekturen	- die Funktionsweise von I/O-Komponenten erklären.	Peripheriekomponenten.	
Embedded Systems	- Programme für einen Mikrocontroller in einer höheren Programmiersprache entwickeln, testen und dokumentieren.	Entwicklung von Mikrocontrollerprogrammen, Interrupt-Service-Routinen.	
Signalverarbeitung	- digitale Signale analysieren und Basisoperationen der digitalen Signalverarbeitung anwenden.	Digitale Signale im Zeit- und Frequenzbereich, Grundoperatoren der digitalen Signalverarbeitung.	
8.Semester – Kompetenzmodul 8:			
Entwurf digitaler Systeme	-das Prinzip des hierarchischen Systementwurfs anwenden.	Partitionierung digitaler Systeme, Anwendung von Komponenten und Bibliotheksmodulen.	
Computer architekturen	- das Hardware-Software-Interface von Standardschnittstellen erklären.	Prozessorschnittstellen.	
Embedded Systems	- Interface-Programme für einen Mikrocontroller erstellen und testen.	Entwicklung von Mikrocontroller-Programmen mit Peripheriefunktionen.	
Signalverarbeitung	- digitale Systeme zur Signalerzeugung und Filterung beschreiben sowie deren Funktionsweise erklären.	Linear-zeitinvariante zeitdiskrete Systeme und deren Beschreibung im Zeit- und Frequenzbereich.	

KU: Zahlendarstellung Schaltnetze SeqLog FSM VHDL DE0

- 21.09.2020 3) Jahresübersicht,Wdhlg. Automaten 4) ÜbungAutomatenentwurf
- 28.09.2020 5) Zustandsautomaten in C, Übung zu Zustandsautomaten (Vorbereitung Test nächste Woche) 6) Übung systematischer Automatenentwurf
- 05.10.2020 7) 1. Test (Automaten;25min); Realisierung von Schaltnetzen und Schaltwerken mittels Festwertspeicher, Test Automaten 8) Realisierung von Schaltnetzen mittels Festwertspeicher 9) Realisierung von Schaltwerken (Automaten) mittels Festwertspeicher
- 12.10.2020 10) uP/uC 101 11) uP/uC 101 - a CLIL activity
- 19.10.2020 12) VHDL Auffrischung / Wiederholung Simulation mit ModelSim 13) VHDL Simulation mit ModelSim
- 09.11.2020 14) Einführung uP-Technik 15) Einführung uP-Technik
- 16.11.2020 16) Festigung Grundkenntnis uP/uC CLIL: "Hello, world" from scratch on a 6502 How do CPUs read machine code? 17) Assembly language vs. machine code
- 23.11.2020 18) uP: Befehlsabarbeitung, Speichermodell, Befehlssatz 19) uP: Softcore 6502 - Quartus Projekt, TB-Erstellung
- 30.11.2020 20) Analyse Reset-Sequenz 6502 ROM-Anbindung an Softcore 6502 21) Analyse Reset-Sequenz 6502 ROM-Anbindung an Softcore 6502 22) Erweiterung 6502-Übungssystem um ROM-Speicher Erstellung erstes Programm mit korrekter Reset-Einsprungs-Adresse
- 14.12.2020 23) erstes Assemblerprogramm auf SIM6502-Umgebung, interne Analyse ModelSim 24) Durchsprache Musterlösung ASL, Aufgabe Einbindung 8bit Register für parallel Ausgabe Adresse 0x6000, Übung Seite 47
- 21.12.2020 25) Musterlösung derHausaufgabe; zwei Lösungsvarianten wurden ausführlich behandelt 26) Programmverzweigungsbefehle - JUMP Übung Seite 47 (Abgabe nicht gefordert) (vom 14.12.20)
- 11.01.2021 27) Assembler - Grundlagen und Funktionsweise in der Sim6502-Umgebung 28) Assembler; mit praktischer Übung in der Sim6502 Umgebung - CLIL
- 18.01.2021 29) RAM unterschied ROM, Klassifizierung von RAM, Funktionsweisen, Kenndaten, Besonderheiten VHDL-Integration Sim6502-Umgebung 30) RAM, VHDL-MUX für mehrere 8bit Datenquellen (q_ROM, q_RAM, NOP auf DI)
- 25.01.2021 31) CPU_DATA_IN_MUX; RAM and bus timing 32) CPU_DATA_IN_MUX; RAM and bus timing
- 01.02.2021 33) Wdhlg. RAM, Besprechung Musterlösung CPUDataIn-MUX 34) Wdhlg. RAM, Besprechung Musterlösung CPUDataIn-MUX 35) RAM-Beschreibung in VHDL (Anlegen eines neuen Datentyps, ARRAY)
- 15.02.2021 36) RAM Implementierung in VHDL 37) RAM Implementierung in VHDL 38) Programmerstellung für RAM-Test; Speicherabbilder in ModelSIM
- 22.02.2021 39) Gruppe2: RAM Beschreibung mittels VHDL, Einbau RAM in 6502-Umgebung 40) Gruppe2: RAM Beschreibung mittels VHDL, Einbau RAM in 6502-Umgebung
- 01.03.2021 41) Wdhlg.Grundfunktion Stack, prinzipielle Funktionsweise Erstellen Testprogramm mit PHA / PLA und Simulation in ModelSim42) Die Rolle des Stack bei Unterprogramm-Aufrufen Simulation jsr und rts in unserer Sim6502-Umgebung
- 08.03.2021 43) Wdhlg. RAM, RAM-Zugriff, Bedeutung von globalen und lokalen Variablen, Grundfunktion Stack Push/Pull Akku-Operation in der 6502 SimUmgebung Initialisierung Stackpointer auf Adresse 01FF 44) Wdhlg. Ablauf Unterprogrammaufruf e, Analyse Sim6502-Umgebung
- 15.03.2021 45) Test (25min); Besprechung Musterlösung, Test - uP Grdlg. 46) Nachbesprechung HWE - Besprechung Projekt "Leistungsendstufe" (Stundentausch mit Do, 11.03.21) Modifikation vereinfachtes Transistor-ESB 47) modifiziertes vereinfachtes Transistor-ESB - Herleitung der Zusammenhänge rbe, rce, ...
- 22.03.2021 48) Gruppe Präsenzunterricht: VHDL - Unterschied Struktur- und Verhaltensbeschreibung anhand einer Automatenaufgabe Gruppe FernUnterricht: PSoC UART 49) Gruppe Präsenzunterricht: File-I/O in VHDL (testbench - Einlesen von Testvektoren und Protokollierung Resultat in output-Datei) Gruppe FernUnterricht: PSoC UART
- 12.04.2021 50) 2. Test (2. Gruppe) Serielle Schnittstelle - Unterschied zur parallel Schnittstelle, Klassifizierungen, Test für 2. Gruppe 51) Serielle Schnittstelle UART, Übertragungsprotokoll, Pegel 52) Serielle Schnittstellen -Übertragungsprotokoll UART, USB-CDC
- 19.04.2021 53) Gruppe Präsenzunterricht: VHDL Statemachine (type-Deklarierung der Zustände, ...), Übung Quartus Gruppe Distancelearning: PSoC ADC 54) Präsenzgruppe: VHDL Datei I/O DistanceGruppe:Übungsauftrag PSoC ADC
- 26.04.2021 55) Wunschprüfung Benjamin Sparer I/O Pins - Arten, Unterschiede, Einsatzbereiche Konfigurationsmodi Konfigurationsänderung während dem Betrieb Gruppe Distance-Learning: Arbeitsauftrag "ADC einmal anders" 56) PSoC I/O
- 03.05.2021 57) VHDL TextI/O - Ausgabe in Textdatei uC Peripherie - Unterschiede I/O Pins FernUnterricht: Arbeitsauftrag "ADC einmal anders" 58) Unterschiede SIO, GPIO, USB-IO FernUnterricht: ADC einmal anders
- 10.05.2021 59) Durchsprache PWM-Projektausarbeitung PSoC Projekt "Helligkeitsmessung mit LED" 60) Besprechung Musterlösung "Digitale Sinusgenerierung mittels VHDL"
- 17.05.2021 61) GruppeA:Helligkeitsmessung LED GruppeB: Pin-Verhalten 62) GruppeA:Helligkeitsmessung LED" GruppeB: Pin-Verhalten 63) GruppeA:Helligkeitsmessung LED GruppeB: Pin-Verhalten
- 31.05.2021 64) Möglichkeit der Testverbesserung für alle anderen / sonst: Interrupts 65) Interrupts

- 07.06.2021 66) Schnittstellen - Klassifizierungen, Vor-/Nachteile, Wdhlg. UART/RS232 (Pegel, Übertragungsprotokoll, ...) 67) I2C
- 14.06.2021 68) I2C-Bus - Start-/Stopbedingungen, Buskommunikation im Detail PSoC: Realisierung I2C Slave (I2C Write - Ansteuerung LEDs) 69) PSoC I2C Slave Read (Temperatursensor) BCP I2C
- 21.06.2021 70) Wunschprüfung:Matthias Riepler SPI, CAN 71) SPI, CAN
- 28.06.2021 72) Wunschprüfungen Johannes Brunner, Christopher Mantovani, Marco Mair, Simon Philipp SPI, CAN Bus 73) CAN-Bus DMA (CLIL)

- 14.09.2020 1) Schulbeginn - WH
- 28.09.2020 2) Vorstellung von Logo! um Automaten zu testen (Praxisbezug Steuerungstechnik)
- 12.10.2020 3) Automaten mit Festwertspeicher - PSoC Übung Würfel4 und Würfel5
- 16.11.2020 4) Rechnen in VHDL - Theorie sowie 2 Übungsbeispiele - N-bit Addierer, 4bit-Zähler
- 30.11.2020 5) VHDL & ROM-Übung: Generierung digitaler Sinusverlauf mittels Zähler und LUT
- 14.12.2020 6) Vorstellung SID, Anbindung SID an 6502 Softcore-Umgebung, Programmerstellung für Ausgabe eines Tones Fertigstellen des Arbeitsauftrages (vom 30.11.20)
- 11.01.2021 7) VHDL Sinusgenerator - Codieren eines Adress-und Korrekturnetzwerkes
- 25.01.2021 8) Erweiterung dig.Sinuserzeugung um PWM-Einheit (VHDL)
- 15.02.2021 9) ModelSim - Simulation RAM Implementierung
- 01.03.2021 10) Simulation Stack in ModelSim
- 15.03.2021 11) HWE (Stundentausch mit Do, 11.03.21) - EGS - Berechnung Vu und Vi mit erweitertem ESB Fertigstellen des Arbeitsauftrages (vom 11.01.21)
- 15.03.2021 12) Arbeitsauftrag: PSoC ADC - Übungen mit SAR- und Sigma-Delta-Umsetzer
- 03.05.2021 13) Übung GPIO-Pins PSoC Creator FernUnterricht:Arbeitsauftrag ADC einmal anders
- 17.05.2021 14) Frequency Measurement with PSoC Timer/PWM Excercise - CLIL
- 17.05.2021 15) Frequency Measurement with PSoC Timer/PWM Excercise - CLIL
- 31.05.2021 16) Fertigstellung "Frequencymeasurement with PSoC" - Anwendung Timer/Counter/PWM in der Praxis
- 14.06.2021 17) PSoC Übung: I2C Master
- 28.06.2021 18) DMA

- 21.09.2020 1) Übung Automaten (mit Simulation in Logo!)
- 05.10.2020 2) PSoC: Übung Automat mit registered LUT (2bit Vor- /Rückwärtszähler), Würfel mit Register LUT, Zustandsdiagramm-Entwurf in PSoC Creator
- 19.10.2020 3) PSoC: Fertigstellen / Abschluss Übung Würfel 4 und Würfel 5; neu Würfel 6 (PSoC Creator und HDL)
- 09.11.2020 4) Quartus: Anlegen eines Speichers, Generierung eines digitalen Sinusverlaufs
- 23.11.2020 5) Rechnen in VHDL
- 21.12.2020 6) SID: Grundprinzip, Anbindung VHDL-Nachbau des MOS6581 in VHDL an unsere sim6502-Umgebung, Beschreibung von Registern, Ausgabe eines einfachen Tones mittels 1. Oszillators
- 18.01.2021 7) Erweiterung der Sim6502 VHDL Umgebung um Komponente, die die 4x7Segment-Anzeigen des DE0-Boards ansteuert
- 01.02.2021 8) Erweiterung der parallel Output-Einheit zu einer PIO-Komponente.
- 22.02.2021 9) Übung: Portieren der Sim6502-Übungsumgebung auf das DE0-Board
- 08.03.2021 10) Warum uP/uC Programme immer eine Endlosschleife benötigen - Probleme bei fehlerhaften Stack-Operationen
- 22.03.2021 11) Übung VHDL Automaten
- 12.04.2021 12) Übungsanleitung - UART (PSoC)
- 26.04.2021 13) Übungsprojekt Helligkeitsmessung mittels LED
- 10.05.2021 14) Online:messtechnische Untersuchung I/O-Pin Verhalten Präsenz:Helligkeitsmessung mittels LED
- 07.06.2021 15) Timer/Counter/PWM Übung:Entfernungsmessung per Ultraschall (Modul HC-SR04 Anbindung an PSoC5LP)
- 21.06.2021 16) Übung Timer - Abstandsmessung mittels UltraschallmodulHC-SR04
- 05.07.2021 Storniert

YU: uC pSoC

- 21.09.2020 4) vhdl_einführung
- 24.09.2020 5) VHDL: Einführung, full_adder
- 28.09.2020 6) vhdl theorie
- 01.10.2020 7) vhdl theorie
- 05.10.2020 8)
- 08.10.2020 9) Stillbeschäftigung - Supplierung Fehleintrag
- 12.10.2020 10) hdl grundlagen
- 15.10.2020 11) vhdl: befehle, modelle
- 19.10.2020 12)
- 22.10.2020 Storniert
- 05.11.2020 13) vhdl_grundlagen_wiederholung
- 09.11.2020 14) vhdl: gundlagen, sinulatioo, entwurf
- 12.11.2020 15) hdl entwurf
- 16.11.2020 16) kombinat.Grundschaltungen in vhdl
- 19.11.2020 17) VHDL - KL, Rechenschaltungen
- 23.11.2020 18) Zahlen, Rechnen in VHDL
- 26.11.2020 19) adderer/subtrahierer für festkomma
- 30.11.2020 20)
- 03.12.2020 21) vhdl multipier
- 10.12.2020 22) ripple carry adder
- 14.12.2020 23)
- 17.12.2020 24)
- 21.12.2020 25)
- 07.01.2021 26) pld
- 11.01.2021 27) pld
- 14.01.2021 28) cpld
- 18.01.2021 29) PLD - CPLD
- 21.01.2021 30)
- 25.01.2021 31) SL in HDL
- 28.01.2021 32) vhdl - seq. Logik
- 01.02.2021 33) hdl - sqeunetielle logik
- 04.02.2021 34) fsm in hdl
- 15.02.2021 35) vhdl behavior
- 18.02.2021 36)
- 22.02.2021 37) hdl theorie (fragen)
- 25.02.2021 38) fsm in hdl
- 01.03.2021 39)
- 04.03.2021 40) vhdl zusammenfassende Übungsstunde
- 08.03.2021 41) psoc wiederholug, programm
- 11.03.2021 42) psoc:wiederholung (programm, theorie)
- 15.03.2021 43) uC - wh
- 18.03.2021 44) psoc

◦22.03.2021 45)
◦25.03.2021 46) uC
◦08.04.2021 47) psoc dma
◦12.04.2021 48) uC
◦15.04.2021 49) uC - dma in Buffer,glue_3
◦19.04.2021 50) wiederholung cortexm3
◦22.04.2021 51) Wiederholung CortexM3/PSoC5LP
◦26.04.2021 52)
◦29.04.2021 53)
◦03.05.2021 54) dsv on psoc
◦06.05.2021 55) psoc state machine
◦10.05.2021 56)
◦17.05.2021 57) psoc: adc,dac,interrupt ◦20.05.2021 58) prüfungen, programmentwicklung , adc
◦27.05.2021 59) programmierung, dac, dac
◦31.05.2021 60) dsv on psoc, adc, dac
◦07.06.2021 61) psoc
◦10.06.2021 62) glue_4
◦14.06.2021 63) dsv psoc
◦17.06.2021 64)
◦21.06.2021 65)
◦24.06.2021 66)
◦28.06.2021 67)
◦01.07.2021 68)
◦05.07.2021 69)
◦08.07.2021 70)

◦14.09.2020 1)
◦14.09.2020 2)
◦28.09.2020 3) mpx_data_flow:entwurf, tb
◦12.10.2020 4) fa_structure
◦16.11.2020 5) vhdl entwurf:Befehlssatz anhand full_adder
◦30.11.2020 6) vhdl, kl, prioritätsencoder
◦14.12.2020 7) rechenschaltungen
◦11.01.2021 8) vhdl - mult/Add
◦25.01.2021 9) RCA, DFF in VHDL
◦15.02.2021 10)
◦01.03.2021 11) hdl fsm sequencedecoder
◦15.03.2021 12) hdl- atm
◦19.04.2021 13) vhdl atm
◦03.05.2021 14)
◦17.05.2021 15) hdl atm
◦31.05.2021 16)
◦14.06.2021 17) hdl - de0
◦28.06.2021 18) hdl de0
◦28.06.2021 19) hdl de0

◦21.09.2020 1) vhdl einführung
◦05.10.2020 2) Kaskodeschaltung, Diff-Verst. (HF Verhalten) - ganze Klasse!; 2. Stunde Schulsprecheransprachen
◦19.10.2020 3)
◦09.11.2020 4)vhdl_verhaltensbeschr eibung (full_adder)
◦23.11.2020 5) vhdl - kl - prioritätsencoder
◦21.12.2020 6) Rechenschaltungen
◦18.01.2021 7) vhdl rechenschaltungen
◦01.02.2021 8) hdl - synchrone grundschaltungen
◦22.02.2021 9) fsm in hdl (squence_dek)
◦08.03.2021 10) hdl: sequence_detector
◦22.03.2021 11) hdl - atm
◦12.04.2021 12)
◦26.04.2021 13)
◦10.05.2021 14)atm atm_hdl atm
◦07.06.2021 15)
◦21.06.2021 16)
◦05.07.2021 17) hdl

YW: dB Amp EGS OP Rect Wien

◦14.09.2020 1) ◦18.09.2020 Storniert ◦21.09.2020 2) WH 3. Klasse ◦25.09.2020 3)
◦28.09.2020 4) ◦02.10.2020 5) ◦05.10.2020 6) ◦09.10.2020 7)
◦12.10.2020 Storniert ◦16.10.2020 8) ◦19.10.2020 9) ◦23.10.2020 10)
◦06.11.2020 11) ◦09.11.2020 12) ◦13.11.2020 13) ◦16.11.2020 14)
◦20.11.2020 15) ◦23.11.2020 16) ◦27.11.2020 17) ◦30.11.2020 18)
◦04.12.2020 19) ◦11.12.2020 20) ◦14.12.2020 21) ◦18.12.2020 22)
◦21.12.2020 23) ◦08.01.2021 24) ◦11.01.2021 25) ◦15.01.2021 26)
◦18.01.2021 27) ◦22.01.2021 28) ◦25.01.2021 29) ◦29.01.2021 30)
◦01.02.2021 31) ◦05.02.2021 32) ◦15.02.2021 33) ◦19.02.2021 34)
◦22.02.2021 35) ◦26.02.2021 36) ◦01.03.2021 37) ◦05.03.2021 38)
◦08.03.2021 39) ◦12.03.2021 40) ◦15.03.2021 41) ◦22.03.2021 42)
◦26.03.2021 43) ◦09.04.2021 44) ◦12.04.2021 45) ◦16.04.2021 46)
◦19.04.2021 47) ◦23.04.2021 48) ◦26.04.2021 49) ◦30.04.2021 50)
◦03.05.2021 51) ◦07.05.2021 52) ◦10.05.2021 53) ◦17.05.2021 54)
◦21.05.2021 55) ◦28.05.2021 56) ◦31.05.2021 57) ◦07.06.2021 58)
◦11.06.2021 59) ◦14.06.2021 60) ◦18.06.2021 61) ◦21.06.2021 62)
◦25.06.2021 63) ◦28.06.2021 64) ◦02.07.2021 65) ◦05.07.2021 66)
◦09.07.2021 67)

◦14.09.2020 1)
◦28.09.2020 2) ◦12.10.2020 Storniert ◦16.11.2020 3) ◦30.11.2020 4) ◦14.12.2020 5)
◦11.01.2021 6) ◦25.01.2021 7) ◦15.02.2021 8) ◦01.03.2021 9) ◦15.03.2021 10)
◦19.04.2021 11) ◦03.05.2021 12) ◦17.05.2021 13) ◦31.05.2021 14) ◦14.06.2021 15)
◦28.06.2021 16)

◦21.09.2020 1) ◦05.10.2020 2) ◦19.10.2020 3) ◦09.11.2020 4) ◦23.11.2020 5)
◦21.12.2020 6) ◦18.01.2021 7) ◦01.02.2021 8) ◦22.02.2021 9) ◦08.03.2021 10)
◦22.03.2021 11) ◦12.04.2021 12) ◦26.04.2021 13) X ◦10.05.2021 14) ◦07.06.2021 15)
◦21.06.2021 16) ◦05.07.2021 17)

51 uC, SoC, ES, -Definitionen

51.1 Microcontroller

From Wikipedia, the free encyclopedia

A microcontroller (MCU for microcontroller unit) is a **small computer on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip**. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

In modern terminology, a microcontroller is similar to, but less sophisticated than, a system on a chip (SoC). An SoC may include a microcontroller as one of its components, but usually integrates it with advanced peripherals like a graphics processing unit (GPU), a Wi-Fi module, or one or more coprocessors.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems. In the context of the internet of things, microcontrollers are an economical and popular means of data collection, sensing and actuating the physical world as edge devices.

Some microcontrollers may use four-bit words and operate at frequencies as low as 4 kHz for low power consumption (single-digit milliwatts or microwatts). They generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nanowatts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

51.2 Singlechip Computer

From Wikipedia, the free encyclopedia

Mit **Singlechip (auch Mikrocontroller, μC, MCU)** werden elektronische Bausteine bezeichnet, die sowohl CPU, ROM und RAM, als auch Ein/Ausgabe Ports in einem integrierten Schaltkreis zusammenfassen. Typische, in Schachcomputern verwendete, Beispiele sind die Chips der Intel 80C50, Hitachi H8 und 6301Y Baureihe. Ein Nachteil dieser kostengünstigen Bauweise ist, dass spätere Programmänderungen - durch den ansonsten einfachen Austausch des ROM oder EPROM - unmöglich sind.

(die Aussage isa Shas! - XH)

51.3 System on a chip

From Wikipedia, the free encyclopedia

The **Raspberry Pi uses a system on a chip as an almost fully contained microcomputer**. This SoC does not contain any kind of data storage, which is common for a microprocessor SoC.

A system on a chip is an integrated circuit (also known as a "chip") that integrates all or most components of a computer or other electronic system. These components almost always include a central processing unit (CPU), memory, input/output ports and secondary storage, often alongside other components such as radio modems and a graphics processing unit (GPU) – all on a single substrate or microchip. It may contain digital, analog, mixed-signal, and often radio frequency signal processing functions (otherwise it is considered only an application processor).

Higher-performance SoCs are often paired with dedicated and physically separate memory and secondary storage (almost always LPDDR and eUFS or eMMC, respectively) chips, that may be layered on top of the SoC in what's known as a package on package (PoP) configuration, or be placed close to the SoC. Additionally, SoCs may use separate wireless modems.

SoCs are in contrast to the common traditional motherboard-based PC architecture, which separates components based on function and connects them through a central interfacing circuit board. Whereas a motherboard houses and connects detachable or replaceable components, SoCs integrate all of these components into a single integrated circuit. An SoC will typically integrate a CPU, graphics and memory interfaces, hard-disk and USB connectivity, random-access and read-only memories and secondary storage and/or their controllers on a single circuit die, whereas a motherboard would connect these modules as discrete components or expansion cards.

An SoC integrates a microcontroller, microprocessor or perhaps several processor cores with peripherals like a GPU, Wi-Fi and cellular network radio modems, and/or one or more coprocessors. Similar to how a microcontroller integrates a microprocessor with peripheral circuits and memory, an SoC can be seen as integrating a microcontroller with even more advanced peripherals. For an overview of integrating system components, see system integration.

More tightly integrated computer system designs improve performance and reduce power consumption as well as semiconductor die area than multi-chip designs with equivalent functionality. This comes at the cost of reduced replaceability of components. By definition, SoC designs are fully or nearly fully integrated across different component modules. For these reasons, there has been a general trend towards tighter integration of components in the computer hardware industry, in part due to the influence of SoCs and lessons learned from the mobile and embedded computing markets. SoCs can be viewed as part of a larger trend towards embedded computing and hardware acceleration.

SoCs are very common in the mobile computing (such as in smartphones and tablet computers) and edge computing markets. They are also commonly used in embedded systems such as WiFi routers and the Internet of Things.

51.4 Embedded system

From Wikipedia, the free encyclopedia

An **embedded system is a computer system** – a combination of a computer processor, computer memory, and input/output peripheral devices – that has a dedicated function within a larger mechanical or electronic system. It is embedded as part of a complete device often including electrical or electronic hardware and mechanical parts. Because an embedded system typically controls physical operations of the machine that it is embedded within, it often has real-time computing constraints. Embedded systems control many devices in common use today. In 2009 it was estimated that ninety-eight percent of all microprocessors manufactured were used in embedded systems.

Modern embedded systems are often based on microcontrollers (i.e. microprocessors with integrated memory and peripheral interfaces), but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialized in a certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the digital signal processor (DSP).

Since the **embedded system is dedicated to specific tasks**, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic light controllers, programmable logic controllers, and large complex systems like hybrid vehicles, medical imaging systems, and avionics. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large equipment rack.

"One of the first recognizably modern embedded systems was the Apollo Guidance Computer,[citation needed] developed ca. 1965"

51.5 Zusammenfassung

* Soc == uC == SingleChipComputer
* 'embedded system' == zweckgewidmet

52 LSV Dic3 YU+ZI 2018

Koordinatoren: **YU+ZI** Imsterbg.2018

Bereich	Lehrstoff	Bildungs- und Lehraufgabe
5.*Semester – – – Kompetenzmodul5* :		
Entwurf digitaler Systeme	Synchrone und asynchrone **Schaltwerke.** (asynchron wurde kritisch hinterfragt) -Zähler	-die verschiedenen Kategorien von Schaltwerken erklären und in Form von Diagrammen spezifizieren.
Computerarchitekturen	Aufbau und Arbeitsweise eines **Mikrocontrollers.** (die nächsten 2 Jahre ATMEL, dann Cortex)	-das Prinzip einer Mikrocontrollerarchitektur und die wesentlichen Schritte der Befehlsausführung erklären.
Embedded Systems	Grundbegriffe von **Hardwarebeschreibungssprachen** und Simulationstools. (-> geteilter Unterricht)	-kombinatorische Systeme mithilfe einer Hardwarebeschreibungssprache spezifizieren und simulieren.
Signalverarbeitung	—	—
Realisierung und Test von Systemen	—	—
6.*Semester – – – Kompetenzmodul6* :		
Entwurf digitaler Systeme	Architekturen von digitalen **Logikbausteinen.** (->aus HWE übernommen – PLDs werden aus HWE herausgenommen)	-Schaltwerke entwerfen und in programmierbaren Logikbausteinen implementieren. (-> geteilter Unterricht)
Computerarchitekturen	Befehlsarchitektur und Software-Tools.	-das Programmiermodell eines **Mikrocontrollers** erklären und einfache Programme entwickeln. (->geteilter Unterricht)
Embedded Systems	Kombinatorische Rechenschaltungen und Spezifikation mittels einer **Hardwarebeschreibungs-sprache.**	-die Vor-und Nachteile der verschiedenen **Zahlendarstellungen** im Dualsystem und die Basisalgorithmen der **Dualarithmetik** erklären. (-> 1.Klasse, 4.Klasse Signalverarbeitung)
Signalverarbeitung	—	—
Realisierung und Test von Systemen	—	—

Sep-Okt:	Digitale Systeme	synchrone und asynchrone **Schaltwerke**
Nov-Dez:	Computer Architekturen	**Microcontroller**: Aufbau und Arbeitsweise
Jan-Feb:	Embedded Systems	Grundbegriffe von **HDL**, Simulationstools
Feb	Digitale Systeme	(synchrone) **Schaltwerke** entwerfen, in PLD implementieren
Mar	Digitale Systeme	Architekturen digitaler **Logikbausteine**
Apr	Computer Architekturen	Programmiermodell eines **Microcontrollers** erklaeren, einfache Programme entwickeln
Mai	Computer Architekturen	+Embedded Systems Programmiermodell eines **Microcontrollers** erklaeren, einfache Programme entwickeln Vor- und Nachteile der verschiedenen **Zahlendarstellungen** im Dualsystem und die Basisalgorithmen der **Dualarithmetik** erklaeren
Juni	Embedded Systems	kombinatorische Rechenschaltungen und Spezifikation mittels einer **Hardwarebeschreibungssprache**

53 Dic3-17Sep'15 sem./Pe4

3. DIGITALE SYSTEME UND COMPUTERSYSTEME semestriert

BGBl. II - Ausgegeben am 17. September 2015 - Nr. 262 (cloud.htlinn.ac.at @Ui8)

III. Jahrgang:

Bereich	Bildungs- und Lehraufgabe	Lehrstoff	KUYU
5.Semester – Kompetenzmodul 5:			
Entwurf digitaler Systeme	-die verschiedenen Kategorien von **Schaltwerken** erklären und in Form von Diagrammen spezifizieren.	Synchrone und asynchrone **Schaltwerke**.	
Computer architekturen	-das Prinzip einer **Mikrocontroller**architektur und die wesentlichen Schritte der Befehlsausführung erklären.	Aufbau und Arbeitsweise eines **Mikrocontrollers**.	
Embedded Systems	-kombinatorische Systeme mithilfe einer **Hardwarebeschreibungssprache** spezifizieren und simulieren.	Grundbegriffe von **Hardwarebeschreibungssprachen** und Simulationstools.	
6.Semester – Kompetenzmodul 6:			
Entwurf digitaler Systeme	-Schaltwerke entwerfen und in programmierbaren **Logikbausteinen** implementieren.	Architekturen von digitalen **Logikbausteinen**.	
Computer architekturen	-das Programmiermodell eines **Mikrocontrollers** erklären und einfache Programme entwickeln.	Befehlsarchitektur und Software-Tools.	
Embedded Systems	-die Vor-und Nachteile der verschiedenen Zahlendarstellungen im Dualsystem und die Basisalgorithmen der **Dualarithmetik** erklären.	Kombinatorische **Rechenschaltungen** und Spezifikation mittels einer Hardwarebeschreibungssprache.	

KU: Zahlendarstellung Schaltnetze SeqLog FSM VHDL DE0
YU: uC pSoC

KU:√Grundklassifikation von Signalen √**Zahlendarstellung** und Grundrechnungsarten im Binärsystem
√allgemein gültige Darstellung von Zahlen √Binär- Oktal- Hexadezimalsystem, Umwandlung von Zahlen
√Umrechnung Zahlensysteme, Konvertierungsfehler √Arithmetik im Dualsystem
√2er Komplement - negative Zahlen im Festkommaformat √Gleitkommadarstellung - Einführung
√Zahlensysteme - IEEE Format √Demo Ganz- / Gleitkommaeinsatz in der Programmierung;
√Arithmetik im Gleitkommasystem √spezielle **Schaltnetze** - MUX, Kodierer, Decodierer, ... √Kodierer, Dekodierer, DeMUX
√**VHDL** - RTL-Beschreibung DEMUX, Vektoren √HA, VA, Umschaltung Addition / Subtraktion, dig. √Komparatoren
√15) Installation von Quartus; geschichtlicher Überblick "programmierbare Logik"
√16) Beginn VHDL - Aufbau Entity und Architecture √17) Einführung VHDL - Entity und Architecture
√18) 1. Test (30min), Test √19) Rückgabe 1. Test; Durchsprache & Musterlösung 1.Test
√20) Aufbau entity & architecture; std_logic_vector √21) VHDL std_ulogic Datentyp; Anwendung HA
√22) Beispiele std_ulogic_Datentyp, signal √23) Wdhlg. 1 Test (25min); Klärung der Testfragen, Wdhlg. 1 Test
√24) Rückgabe Test-Wdhlg; Durchsprache der Lösungen √25) VHDL- Umsetzung 1-aus-4 Codeumsetzer mit **DE0** HW
√26) VHDL - Signale & Hierarchie √27) VHDL - Hierarchie (Übung 4bit VA) √28) DE0 Übung: 4bit Addierer
√29) VHDL-Projekt: 4bit Addierer √30) VHDL - Signalflussrichtung, Verknüpfungsoperator &
√31) VHDL: when-else Bedingung, Anwendung Prioritätsdekoder
√32) VHDL with-select, 7-Segment-Decoder √33) VHDL - Ansteuerung 7-Segment-Anzeige
√34) VHDL - Ansteuerung 7-Segment-Anzeige, 4bit-Addierer √35) VHDL - Abschlussprojekt 74LS85
√36) Übungsstunde VHDL - Beispielprojekt; Möglichkeit der Fragenstellung zur Testvorbereitung
√37) 2. Test (25min) - Nachbesprechung 2. Test, 2. Test (VHDL) √38) Beginn **sequentielle Logik** √39) FF: RS, D
√40) Zweiflankensteuerung, JK-FF, T-FF √41) JK-FF, T-FF, Anwendungen FF
√42) Einsatzbereiche von FFs, asynchrone/synchrone FF-Schaltungen √43) Logikfamilien √44) D-Latch in VHDL
√45) Latch in VHDL √46) Latches in VHDL √47) Beginn **Automaten** - Zustandsgraph
√48) Beginn Automaten - Zustandsgraph, Ad-hoc DesignMuster (gegenseitige Verriegelung,Reihenfolgeschaltung, ...)
√49) Zustandsgraph: Erläuterung einer Schritt-für-Schritt Entwicklung anhand des "Quizz-Master-Beispiels"
√50) Übung Zustandsgraph Impulsschaltung für 2 Meldeleuchten"; Beginn Prozess in VHDL
√51) VHDL - Prozess: if-elsif-else-end if Struktur √52) MUX realisiert mi if-elsif-else-end if Struktur; casein VHDL
√53) VHDL - case Struktur √54) VHDL - Schaltwerke, Register und Taktteiler √55) VHDL - 2bit Binärzähler
√56) VHDL - 2bit Zähler √57) VHDL - for-Schleife √58) for-Schleife in VHDL √59) Ad-hoc Entwurf
√60) Wunschprüfung Bastani Sam; Ad-hoc Entwurf "Quizzmaster"
√61) Nachtest Schüler die beim 2. Test gefehlt haben, Wunschprüfung Eric Beer (15min mündlich); Automatenmodelle
√62) Automatenmodelle (Mealy, Moore, vorauslaufender Moore), Vorgangsweise system. Automatenentwurf, Wunschprü
√63) Ad-hoc-Entwurf (Quizzmaster), Nachtesttermin 2.Test für fehlende Schüler (20min)
√64) Wunschprüfung Brkic, Montavani (je 15min); system. Entwurf
√65) Wunschprüfung Sarac Yunus (15min); systematischer Schaltungsentwurf
√66) Automaten in VHDL, Grundaufbau uC √67) Rechnen in VHDL; Automaten in VHDL; uC Aufbau

09.09.2019 1) Allgemeines zu **HWE**, Projektablauf; Beurteilung; Rückblick bisheriger Projekte; Beginn Vorbesprechung **Rechteck-Dreieck-Oszillator**
16.09.2019 1) Vorbesprechung Rechteck-Dreieck-Oszillator -Wdhlg. inv. Integrierer, nichtinv. Schmitt-Trigger
23.09.2019 2) Vorbesprechung Rechteck-Dreieck-Oszillator
30.09.2019 2) Projekt1 - Rechteck Dreieck Oszillator
07.10.2019 3) 1. Projekt - Rechteck Dreieck Oszillator; Generierung Puls-Pause
14.10.2019 3) 1_Projekt Rechteck-Dreieck-Oszillator Altium-Simulation, Steckbrett-Entwurf
21.10.2019 4) 1_Projekt Rechteck_Dreieck_Oszillator - Simulation, Steckbrettaufbau
11.11.2019 4) Steckbrettaufbau Rechteck-Dreieck-Oszillator, Beginn Dokumentation
18.11.2019 5) Fertigstellung und Abgabe "Rechteck-Dreieck-Oszillator"
25.11.2019 5) Aufbau & Test Rechteck-Dreieck-Oszillator; Abschluss Dokumentation
02.12.2019 6) 2.Projekt: **PSoC Würfel - UDB**
09.12.2019 6) 2. Projekt - PSoC Würfel
16.12.2019 7) Fertigstellung & Abgabe PSoC Würfelprojekt
13.01.2020 8) PSoC: Realisierung digitaler Logik mit MUX; LUT im Einsatz
27.01.2020 9) Ansteuerung PSoC Pin in C, Übungsanleitung Würfel 9
20.01.2020 7) Kurzprojekt: Ansteuerung 7-Segment-Anzeige PSoC
03.02.2020 8) Vorbesprechung / Beginn 5. Projekt (PSoC Spiel)
17.02.2020 9) PSoC Spiel (Programmieraufgabe) - 5.Projekt
24.02.2020 10) 5.Projekt - PSoC Spiel
02.03.2020 10) Fertigstellung PSoC Spiel - Abgabe 5. Projekt
09.03.2020 11) 5. Projekt: Altium - MAX7219
16.03.2020 11) Fertigstellung MAX7219 8-digit-Anzeigenplatine in Altium
23.03.2020 12) **FET-Projekt**
30.03.2020 12) **FET-Übungsprojekt**
20.04.2020 13) **Analogprojekt**
27.04.2020 13) **Analogprojekt - Transistor**
04.05.2020 14) **Transistorprojekt**
11.05.2020 14) **HWE-Analogprojekt**, ab 08:30 Letto Schulung (Stillbeschäftigung der Schüler)
18.05.2020 15) **HWE-Analogprojekt**
25.05.2020 15) **HWE-Analogprojekt**
08.06.2020 16) PCB **einstufiger Transistorverstärker**
15.06.2020 16) PCB einstufiger Transistorverstärker
22.06.2020 17) FET **Konstantstromquelle**
29.06.2020 17) FET Konstantstromquelle
06.07.2020 18) DIC-Projekt gem. Unterrichtsplanung alle Inhalte vermittelt - nichts offen

YU:√ 3) wh kl, sl√ 4) sl √ 5) sl mealy moore √ 8) glue1 , **fsm** √ 9) GLUE1_DIC √ 10) fsm √ 11) fsm, ca start √ 13) **uc** √ 15) uc √ 16) smü, uc, GLUE2_DIC √ 17) glue_zurueck, besprechung, noten √ 19) prüfungen, uc √ 20) ca_**psoc** √ 21) psoc programm √ 22) programm blinking led und gpio √ 23) psoc programm, cortex √ 24) psoc_cortex √ 27) psoc software pwm √ 29) fu: cortex, GLUE3_DIC √ 31) fu: uc - programmiermodell √ 32) arbeitsauftrag √ 33) peripherie √ 35) GLUE4_DIC √ 36) psoc √ 37) prüfungen, psoc_timer_(adc) √ 38) psoc peripherie

11.09.2019 1)
18.09.2019 –
09.10.2019 2)
16.10.2019 2) hierachisches design
23.10.2019 3) **dreieck-rechteck** altium
06.11.2019 4) **dreieck-rechteck,** altium
13.11.2019 3)
20.11.2019 5) **dreieck-rechteck**
27.11.2019 4) dreieck rechteck aufbau
04.12.2019 6) schaltungsaufbau, messung
04.12.2019 7) schaltungsaufbau, messung
11.12.2019 5) **dr-re**
18.12.2019 8)
18.12.2019 9) **dr-re**
08.01.2020 6) fsm
15.01.2020 10) fsm
22.01.2020 7) fsm
29.01.2020 11) psoc-fsm
05.02.2020 8) fsm psoc
19.02.2020 9) edge_detector
26.02.2020 12)
04.03.2020 10) **bio_amp: diff_amp - vb**
11.03.2020 13)
18.03.2020 11)
25.03.2020 14) **aktives filter**
01.04.2020 12) **filter**
15.04.2020 13) **fu instrumentenverstarker**
22.04.2020 15) **fu:instrumentenverstaerker**
29.04.2020 14)
06.05.2020 16) **fu: iv**
13.05.2020 15) arbeitsauftrag
20.05.2020 17) Platine
27.05.2020 16)
10.06.2020 17)
17.06.2020 18)
24.06.2020 18) **bio_amp**
01.07.2020 19)
08.07.2020 Storniert

54 Imsterberg'18 "10Uhr45" SfM/SgE/SgH

```
Ged^/chtnisprotokoll ^...ber die Besprechung der Lehrinhalte DIC

Zeit: 10Uhr45
Ort: AV B^...ro
Teilnehmer: Christoph Sch^¶nherr, Alfred Stumpfel, Michael Kupfner, Felix Klingler, Georg
    Steinwender,
Helmut Stecher

Es wurde auf Grundlage des bestehenden LP und des im Herbst einzuf^...hrenden semestrierten
Lehrplanes die Inhalte koordiniert.
Die Expertengruppe ist einstimmig zum Ergebnis gekommen, dass f^...r die semestrierten Lehrinhalte
die Stundenverschiebung von der 3. Klasse in die 1. Klasse in DIC nicht zielf^...hrend ist, daher wird
     diese
Verschiebung nur noch im Schuljahr 2015/16 durchgef^...hrt und dann zur^...ckgenommen.
Die Lehrinhalte wurden Klasse f^...r Klasse besprochen, die ^Änderungen wurden in die
"Lehrstoffverteilung neuí Þ eingepflegt und ist f^...r alle DIC-Lehrer ab dem Schuljahr 15/16
    verpflichtend.
Es gibt einige ^Þberschneidungen mit anderen F^/chern und diese wurden von Seite der DIC Inhalte
bereinigt. Die DIC Lehrer und Lehrerinnen m^...ssen sich darauf verlassen k^¶nnen, dass Inhalte, die
    f^...r
diesen Unterricht notwendig sind, auch unterrichtet werden. Die Koordination wird von AV Stecher
^...bernommen und die Inhalte werden eingefordert.

Digitale Systeme und Computersysteme:
Im Bereich Entwurf digitaler Systeme
k^¶nnen die Absolventinnen und Absolventen die grundlegenden Verfahren des digitalen
Systementwurfs anwenden, Digitalbausteine ausw^/hlen und den Entwurf dokumentieren. Sie k^¶nnen
digitale Systeme durch Simulation verifizieren und bewerten, unter Verwendung von
Entwicklungsplattformen implementieren, in Betrieb nehmen und testen.
Im Bereich Computerarchitekturen
kennen die Absolventinnen und Absolventen die Basisarchitekturen und Kenngr^¶szen moderner
Computer und k^¶nnen eine f^...r die jeweilige Anwendung geeignete Computerarchitektur ausw^/hlen.
Sie k^¶nnen Computerarchitekturen analysieren sowie deren Eignung f^...r spezielle Anwendungsf^/lle
bewerten und vergleichen.
Im Bereich Embedded Systems
k^¶nnen die Absolventinnen und Absolventen Embedded Systems unter Verwendung von
Entwicklungsplattformen als Hardware Software Co-Design realisieren sowie f^...r die jeweilige
Anwendung geeignete programmierbare Logikbausteine ausw^/hlen,
mithilfe von Hardwarebeschreibungen konfigurieren und testen.
Im Bereich Signalverarbeitung
kennen die Absolventinnen und Absolventen die grundlegenden Methoden der digitalen
Signalverarbeitung und die Architekturen moderner Signalprozessoren. Sie k^¶nnen f^...r die jeweilige
Anwendung geeignete Signalverarbeitungsalgorithmen ausw^/hlen und parametrieren sowie
Methoden der Signalverarbeitung durch Simulation analysieren und bewerten sowie Algorithmen der
Signalverarbeitung implementieren, testen und optimieren.
Im Bereich Realisierung und Test von Systemen
k^¶nnen die Absolventinnen und Absolventen Prototypen digitaler Systeme fertigen, in Betrieb
nehmen, unter Verwendung von Software-Tools und Messger^/ten testen bzw. Fehler lokalisieren und
beheben.

3. DIGITALE SYSTEME UND COMPUTERSYSTEME (semestriert - UNVER^ÄNDERT)
III. Jahrgang:
5. Semester í Ó Kompetenzmodul 5:
Bildungs-und Lehraufgabe:
Die Sch^...lerinnen und Sch^...ler k^¶nnen im
Bereich Entwurf digitaler Systeme
-die verschiedenen Kategorien von Schaltwerken erkl^/ren und in Form von Diagrammen spezifizieren.
Bereich Computerarchitekturen
-das Prinzip einer Mikrocontrollerarchitektur und die wesentlichen Schritte der Befehlsausf^...hrung
    erkl^/ren.
Bereich Embedded Systems
-kombinatorische Systeme mithilfe einer Hardwarebeschreibungssprache spezifizieren und simulieren.
Lehrstoff:
Bereich Entwurf digitaler Systeme:
Synchrone und asynchrone Schaltwerke.
Bereich Computerarchitekturen:
Aufbau und Arbeitsweise eines Mikrocontrollers.
Bereich Embedded Systems:
Grundbegriffe von Hardwarebeschreibungssprachen und Simulationstools.
6. Semester í Ó Kompetenzmodul 6:
Bildungs-und Lehraufgabe:
```

```
Die Sch^...lerinnen und Sch^...ler k^¶nnen im
Bereich Entwurf digitaler Systeme
-Schaltwerke entwerfen und in programmierbaren Logikbausteinen implementieren.
Bereich Computerarchitekturen
-das Programmiermodell eines Mikrocontrollers erkl^/ren und einfache Programme entwickeln.
Bereich Embedded Systems
-die Vor-und Nachteile der verschiedenen Zahlendarstellungen im Dualsystem und die Basisalgorithmen
    der
Dualarithmetik erkl^/ren.
Lehrstoff:
Bereich Entwurf digitaler Systeme:
Architekturen von digitalen Logikbausteinen.
Bereich Computerarchitekturen:
Befehlsarchitektur und Software-Tools.
Bereich Embedded Systems:
Kombinatorische Rechenschaltungen und Spezifikation mittels einer Hardwarebeschreibungssprache.

IV. Jahrgang:
7. Semester í Ó Kompetenzmodul 7:
Bildungs-und Lehraufgabe:

Die Sch^...lerinnen und Sch^...ler k^¶nnen im
Bereich Entwurf digitaler Systeme
-den Unterschied zwischen einer Struktur-und einer Verhaltensbeschreibungssprache erkl^/ren.
Bereich Computerarchitekturen
-die Funktionsweise von I/O-Komponenten erkl^/ren.
Bereich Embedded Systems
-Programme f^...r einen Mikrocontroller in einer h^¶heren Programmiersprache entwickeln, testen und
dokumentieren.
Bereich Signalverarbeitung
-digitale Signale analysieren und Basisoperationen der digitalen Signalverarbeitung anwenden.
Lehrstoff:
Bereich Entwurf digitaler Systeme:
Spezifikation und Darstellung von digitalen Systemen.
Bereich Computerarchitekturen:
Peripheriekomponenten.
Bereich Embedded Systems:
Entwicklung von Mikrocontrollerprogrammen, Interrupt-Service-Routinen.
Bereich Signalverarbeitung:
Digitale Signale im Zeit-und Frequenzbereich, Grundoperatoren der digitalen Signalverarbeitung.
8. Semester í Ó Kompetenzmodul 8:
Bildungs-und Lehraufgabe:
Die Sch^...lerinnen und Sch^...ler k^¶nnen im
Bereich Entwurf digitaler Systeme
-das Prinzip des hierarchischen Systementwurfs anwenden.
Bereich Computerarchitekturen
-das Hardware-Software-Interface von Standardschnittstellen erkl^/ren.
Bereich Embedded Systems
-Interface-Programme f^...r einen Mikrocontroller erstellen und testen.
Bereich Signalverarbeitung
-digitale Systeme zur Signalerzeugung und Filterung beschreiben und deren Funktionsweise erkl^/ren.
Lehrstoff:
Bereich Entwurf digitaler Systeme:
Partitionierung digitaler Systeme, Anwendung von Komponenten und Bibliotheksmodulen.
Bereich Computerarchitekturen:
Prozessorschnittstellen.
Bereich Embedded Systems:
Entwicklung von Mikrocontroller-Programmen mit Peripheriefunktionen.

Bereich Signalverarbeitung
Linear-zeitinvariante zeitdiskrete Systeme und deren Beschreibung im Zeit-und Frequenzbereich.
V. Jahrgang -Kompetenzmodul 9:
9. Semester:
Bildungs-und Lehraufgabe:
Die Sch^...lerinnen und Sch^...ler k^¶nnen im
Bereich Entwurf digitaler Systeme
-komplexe digitale Systeme in Form eines Top-Down-Design entwickeln.
Bereich Computerarchitekturen
-Architekturen zur Optimierung der Leistungsf^/higkeit spezieller Anwendungen beschreiben und deren
Funktionsweise erkl^/ren.
Bereich Embedded Systems
-die Aufgaben und Funktionsweise von Echtzeitbetriebssystemen erkl^/ren.
Bereich Signalverarbeitung
```

```
-Signalprozessor-Architekturen erklˆ/ren und Basisalgorithmen der digitalen Signalverarbeitung
    implementieren.
Lehrstoff:
Bereich Entwurf digitaler Systeme:
Realisierung eines digitalen Systems ausgehend von einer Verhaltensbeschreibung.
Bereich Computerarchitekturen:
Anwendungsspezifische Prozessorarchitekturen.
Bereich Embedded Systems:
Betriebssysteme fˆ...r Embedded Systems, Echtzeitverarbeitung.
Bereich Signalverarbeitung:
Ausgewˆ/hlte Beispiele der digitalen Signalverarbeitung und deren Realisierung mittels
    Signalprozessoren.
10. Semester:
Bildungs-und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler kˆ¶nnen im
Bereich Entwurf digitaler Systeme
-komplexe digitale Systeme mit Hilfe von Softwaretools optimieren.
Bereich Computerarchitekturen
-Architekturen fˆ...r spezielle Anwendungen anhand einer Spezifikation auswˆ/hlen.
Bereich Embedded Systems
-ein Echtzeitbetriebssystem fˆ...r spezielle Anwendungen konfigurieren.
Bereich Signalverarbeitung
-Implementierungen von Algorithmen analysieren und testen.
Lehrstoff:
Bereich Entwurf digitaler Systeme:
-Analysetools.
Bereich Computerarchitekturen:

-Anwendungsspezifische Prozessorarchitekturen.
Bereich Embedded Systems:
-Echtzeitbetriebssysteme.
Bereich Signalverarbeitung:
-Analyse-und Testverfahren.

Gegenˆ...berstellung

III. Jahrgang:

alt

neu

Bildungs- und Lehraufgabe ˆ...ber alle Jahrgˆ/nge

Bildungs- und Lehraufgabe fˆ...r jeden Jahrgang
definiert

definiert
Kompetenzbereich "Entwurf digitaler Systemeí Þ:
Bildungs- und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler
- kˆ¶nnen die grundlegenden Verfahren des digitalen
Systementwurfs

anwenden,

Digitalbausteine

auswˆ/hlen und den Entwurf dokumentieren;
- kˆ¶nnen digitale Systeme durch Simulation
verifizieren und bewerten, unter Verwendung von
Entwicklungsplattformen implementieren, in Betrieb
nehmen und testen.
Kompetenzbereich "Computerarchitekturení Þ:
Bildungs- und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler
- kennen die Basisarchitekturen und Kenngrˆ¶szen
moderner Computer und kˆ¶nnen eine fˆ...r die
jeweilige Anwendung geeignete Computerarchitektur
auswˆ/hlen;
- kˆ¶nnen Computerarchitekturen analysieren sowie
deren Eignung fˆ...r spezielle Anwendungsfˆ/lle
bewerten und vergleichen (V).
```

```
Kompetenzbereich "Embedded Systemsí Þ:
Bildungs- und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler
- kˆ¶nnen Embedded Systems unter Verwendung von
Entwicklungsplattformen als Hardware Software CoDesign realisieren;
- kˆ¶nnen fˆ...r die jeweilige Anwendung geeignete
programmierbare Logikbausteine auswˆ/hlen, mithilfe
von Hardwarebeschreibungen konfigurieren und
testen.
Kompetenzbereich "Realisierung und Test von
Systemení Þ:
Bildungs- und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler kˆ¶nnen Prototypen
digitaler Systeme fertigen, in Betrieb nehmen, unter
Verwendung von Software-Tools und Messgerˆ/ten
testen bzw. Fehler lokalisieren und beheben.

5. Semester í Ó Kompetenzmodul 5:
Bildungs-und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler kˆ¶nnen im
Bereich Entwurf digitaler Systeme
-die verschiedenen Kategorien von Schaltwerken

erklˆ/ren und in Form von Diagrammen spezifizieren.
Bereich Computerarchitekturen
-das Prinzip einer Mikrocontrollerarchitektur und die
wesentlichen Schritte der Befehlsausfˆ...hrung erklˆ/ren.
Bereich Embedded Systems
-kombinatorische

Systeme

Hardwarebeschreibungssprache

mithilfe

einer

spezifizieren

und

simulieren.
Kompetenzbereich "Entwurf digitaler Systemeí Þ:
Lehrstoff:
Zahlendarstellung und Codierung; Entwurf von
Schaltwerken; programmierbare Logikbausteine.
Kompetenzbereich "Computerarchitekturení Þ:
Lehrstoff:
Basisarchitekturen.
Kompetenzbereich "Embedded Systemsí Þ:
Lehrstoff:
Prozessoren, Einfˆ...hrung in die
Mikrocontrollerprogrammierung.
Kompetenzbereich "Realisierung und Test von
Systemení Þ:
Lehrstoff:
Digitale Systeme:
Aufbau, Test und Fehlersuche bzw. -behebung.

Lehrstoff:
Bereich Entwurf digitaler Systeme:
Synchrone und asynchrone Schaltwerke. (asynchron
wurde kritisch hinterfragt) -õÂ© Zˆ/hler
Bereich Computerarchitekturen:
Aufbau und Arbeitsweise eines Mikrocontrollers.

(die nˆ/chsten 2 Jahre ATMEL, dann Cortex)
Bereich Embedded Systems:
Grundbegriffe von Hardwarebeschreibungssprachen
und Simulationstools. (-> geteilter Unterricht )
6. Semester - Kompetenzmodul 6:
Bildungs-und Lehraufgabe:
```

```
Die Schˆ...lerinnen und Schˆ...ler kˆ¶nnen im
Bereich Entwurf digitaler Systeme
-Schaltwerke entwerfen und in programmierbaren
Logikbausteinen implementieren. (-> geteilter
Unterricht,)
Bereich Computerarchitekturen
-das Programmiermodell eines Mikrocontrollers
erklˆ/ren und einfache Programme entwickeln. (íÅÒ
geteilter Unterricht )
Bereich Embedded Systems
-die Vor-und Nachteile der verschiedenen
Zahlendarstellungen im Dualsystem und die
Basisalgorithmen der Dualarithmetik erklˆ/ren.
(íÅÒ 1.Klasse, 4.Klasse Signalverarbeitung )
Lehrstoff:
Bereich Entwurf digitaler Systeme:
Architekturen von digitalen Logikbausteinen. . (->
aus HWE ˆ...bernommen - PLDs werden aus HWE
herausgenommen)
Bereich Computerarchitekturen:
Befehlsarchitektur und Software-Tools.

Bereich Embedded Systems:
Kombinatorische Rechenschaltungen und
Spezifikation mittels einer Hardwarebeschreibungssprache.

IV. Jahrgang
7. Semester - Kompetenzmodul 7:
Bildungs-und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler kˆ¶nnen im
Bereich Entwurf digitaler Systeme
-den Unterschied zwischen einer Struktur-und einer
Verhaltensbeschreibungssprache erklˆ/ren.
Bereich Computerarchitekturen
-die Funktionsweise von I/O-Komponenten (->
Standard-Peripherie eines uC, eventuell USB?)
erklˆ/ren.
Bereich Embedded Systems
-Programme fˆ...r einen Mikrocontroller in einer
hˆ¶heren Programmiersprache entwickeln, testen und
dokumentieren. (-> geteilter Unterricht )
Bereich Signalverarbeitung
-digitale Signale analysieren und Basisoperationen
der digitalen Signalverarbeitung anwenden.
Kompetenzbereich "Entwurf digitaler Systemeí Þ:
Hardwarebeschreibungssprachen;
Komponentenauswahl und Systemdesign.

(-> + Zahlen 6.Semester - Flieskommadarstellung )
Lehrstoff:
Bereich Entwurf digitaler Systeme:

Kompetenzbereich "Computerarchitekturení Þ:
Computersysteme.

Spezifikation und Darstellung von digitalen

Kompetenzbereich "Embedded Systemsí Þ:
Entwurf und Dokumentation von
Mikrocontrollerprogrammen, Peripheriebausteine,
Interface-Techniken.

Bereich Computerarchitekturen:

Systemen.

Peripheriekomponenten.
(->ADC(DAC/Timer/UART/SPI/dann USB statt

Kompetenzbereich "Signalverarbeitungí Þ:
Algorithmen der digitalen Signalverarbeitung:
Signalanalyse, Signalgenerierung, Filterung
```

```
UART)

Kompetenzbereich "Realisierung und Test von
Systemení Þ:
Mikrocontroller und programmierbare
Logikbausteine:
Implementierung, Test und Fehlersuche bzw. behebung.

Interrupt-Service-Routinen. (-> geteilter Unterricht )

Bereich Embedded Systems:
Entwicklung von Mikrocontrollerprogrammen,

Bereich Signalverarbeitung:
Digitale Signale im Zeit-und Frequenzbereich (-> zTransformation ?), Grundoperatoren(-> MAC
ausreichend ? ) der digitalen Signalverarbeitung.
8. Semester - Kompetenzmodul 8:
Bildungs-und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler kˆ¶nnen im

Bereich Entwurf digitaler Systeme
-das Prinzip des hierarchischen Systementwurfs
anwenden.
Bereich Computerarchitekturen
-das Hardware-Software-Interface
Standardschnittstellen erklˆ/ren.
Bereich Embedded Systems
-Interface-Programme fˆ...r einen Mikrocontroller
erstellen und testen. (-> geteilter Unterricht z.B.:
Datenˆ...bertragung, PWM-Ausgabe, Einlesen
Drehdekoder...)
Bereich Signalverarbeitung
-digitale Systeme zur Signalerzeugung und Filterung
beschreiben und deren Funktionsweise erklˆ/ren.
Lehrstoff:
Bereich Entwurf digitaler Systeme: (-> FPGA )
Partitionierung digitaler Systeme, Anwendung von
Komponenten und Bibliotheksmodulen. (->digitale
Filter FIR.. )

Bereich Computerarchitekturen:
Prozessorschnittstellen. (->
Speichererweiterung/DMA/Memoryinterface... )
Bereich Embedded Systems:
Entwicklung von Mikrocontroller-Programmen mit
Peripheriefunktionen. (-> ˆÞbermittlung )

Bereich Signalverarbeitung
Linear-zeitinvariante zeitdiskrete Systeme und deren
Beschreibung im Zeit-und Frequenzbereich.

V. Jahrgang
V. Jahrgang -Kompetenzmodul 9:
9. Semester:
Bildungs-und Lehraufgabe:
Die Schˆ...lerinnen und Schˆ...ler kˆ¶nnen im

Kompetenzbereich "Entwurf digitaler Systemeí Þ:
Beschreibung, Entwurf, Simulation und
Dokumentation komplexer digitaler Systeme; EMVkonformes Design
Kompetenzbereich "Computerarchitekturení Þ:

Bereich Entwurf digitaler Systeme
-komplexe digitale Systeme in Form eines TopDown-Design entwickeln. (-> z.B.SoftCore - uC
Implementierung in FPGA ? (Michael))
Bereich Computerarchitekturen
-Architekturen zur Optimierung der

Anwendungsspezifische Architekturen.
Kompetenzbereich "Embedded Systemsí Þ:
Betriebssysteme fˆ...r Embedded Systems;
Echtzeitverarbeitung.
Kompetenzbereich "Signalverarbeitungí Þ:
```

```
Signalprozessoren, Entwurf und Dokumentation von
Signalprozessoranwendungen
Kompetenzbereich "Realisierung und Test von
Systemení Þ:
Komplexe Systeme:
Einsatz von Betriebssystemen f˜...r Embedded
Systems; EMV-konformer Aufbau.

Leistungsf^/higkeit spezieller Anwendungen
beschreiben und deren Funktionsweise erkl^/ren.
Bereich Embedded Systems
-die Aufgaben und Funktionsweise von
Echtzeitbetriebssystemen (-> RTOS )erkl^/ren.
Bereich Signalverarbeitung
-Signalprozessor-Architekturen erkl^/ren und
Basisalgorithmen der digitalen Signalverarbeitung
implementieren. (-> kein Signalprozessor, sondern
eher z.B. LABVIEW )
Lehrstoff:
Bereich Entwurf digitaler Systeme:
Realisierung eines digitalen Systems ausgehend von
einer Verhaltensbeschreibung.
Bereich Computerarchitekturen:
Anwendungsspezifische Prozessorarchitekturen.
(-> z.B.: Prozessoren f˜...r Video/Audio -
Anwendungen, Verschl^...sselung usw, lexikalisch)
Bereich Embedded Systems:
Betriebssysteme f˜...r Embedded Systems, (-> FSST )
Echtzeitverarbeitung.
Bereich Signalverarbeitung:
Ausgew^/hlte Beispiele der digitalen
Signalverarbeitung und deren Realisierung mittels
Signalprozessoren. (-> Theorie )
10. Semester:
Bildungs-und Lehraufgabe:
Die Sch^...lerinnen und Sch^...ler k^¶nnen im
Bereich Entwurf digitaler Systeme
-komplexe digitale Systeme mit Hilfe von
Softwaretools optimieren.
Bereich Computerarchitekturen
-Architekturen f˜...r spezielle Anwendungen anhand
einer Spezifikation ausw^/hlen.
Bereich Embedded Systems
-ein Echtzeitbetriebssystem f˜...r spezielle
Anwendungen konfigurieren.
Bereich Signalverarbeitung
-Implementierungen von Algorithmen analysieren
und testen.
Lehrstoff:
Bereich Entwurf digitaler Systeme:
-Analysetools.
Bereich Computerarchitekturen:
-Anwendungsspezifische Prozessorarchitekturen.

Bereich Embedded Systems:
-Echtzeitbetriebssysteme.
Bereich Signalverarbeitung:
-Analyse-und Testverfahren.

Abschlieszend wird in der Gruppe vereinbart, dass eine abgestimmte Literatur als Arbeitsgrundlage
    f˜...r
die Lehrinhalte definiert wird. Georg Steinwender ^...bernimmt die Koordination.

AV Stecher bedankt sich bei allen f˜...r das konstruktive Gespr^/ch.
```

55 Kompetenz-Lehrplan-2011 (cloud)

55.1 3. DIGITALE SYSTEME UND COMPUTERSYSTEME III. Jahrgang:

Kompetenzbereich	Lehrstoff
„Entwurf digitaler Systeme“:	**Zahlendarstellung** und Codierung; Entwurf von **Schaltwerken**; **programmierbare Logikbausteine**.
„Computerarchitekturen“:	Basisarchitekturen.
„Embedded Systems“:	Prozessoren, Einführung in die **Mikrocontroller**programmierung.
„Signalverarbeitung“:	—
„Realisierung und Test von Systemen“:	Digitale Systeme: Aufbau, Test und Fehlersuche bzw. -behebung.

BGBl. II - Ausgegeben am 7. September 2011- Nr. 300

Anlage 1.2

LEHRPLAN DER HÖHEREN LEHRANSTALT FÜR ELEKTRONIK UND TECHNISCHE INFORMATIK

3. DIGITALE SYSTEME UND COMPUTERSYSTEME

Kompetenzbereich „Entwurf digitaler Systeme“: Bildungs- und Lehraufgabe: Die Schülerinnen und Schüler
-können die grundlegenden Verfahren des digitalen Systementwurfs anwenden,
Digitalbausteine auswählen und den
Entwurf dokumentieren;
- können digitale Systeme durch Simulation verifizieren und
bewerten,
unter Verwendung von Entwicklungsplattformen implementieren,
in Betrieb nehmen und
testen.

Lehrstoff: III. Jahrgang:
Zahlendarstellung und
Codierung;
Entwurf von Schaltwerken;
programmierbare Logikbausteine.

Kompetenzbereich „Computerarchitekturen“: Bildungs- und Lehraufgabe: Die Schülerinnen und Schüler
- kennen die Basisarchitekturen und Kenngrößen moderner Computer und können eine für die jeweilige Anwendung geeignete Computerarchitektur auswählen;
- können Computerarchitekturen analysieren sowie deren Eignung für spezielle Anwendungsfälle bewerten und vergleichen (V).

Lehrstoff: III. Jahrgang:
Basisarchitekturen.

Kompetenzbereich „Embedded Systems“: Bildungs- und Lehraufgabe: Die Schülerinnen und Schüler
- können Embedded Systems unter Verwendung von Entwicklungsplattformen als Hardware Software Co-Design realisieren;
- können für die jeweilige Anwendung geeignete programmierbare Logikbausteine auswählen, mithilfe von Hardwarebeschreibungen konfigurieren und testen.

Lehrstoff: III. Jahrgang:
Prozessoren,
Einführung in die Mikrocontrollerprogrammierung.

Kompetenzbereich „Signalverarbeitung“: Bildungs- und Lehraufgabe:

Lehrstoff: III. Jahrgang:
—

Kompetenzbereich „Realisierung und Test von Systemen“: Bildungs- und Lehraufgabe: Die Schülerinnen und Schüler
können Prototypen digitaler Systeme fertigen,
in Betrieb nehmen,
unter Verwendung von Software-Tools und
Messgeräten testen bzw.
Fehler lokalisieren und
beheben.

Lehrstoff: III. Jahrgang:
Digitale Systeme:
Aufbau,
Test und
Fehlersuche bzw.
-behebung.

DEFINITIONEN nach Chris"XH"

ACHTUNG:
Diese Definitionen entsprechen nicht bzw. nicht ganz den allgemeinen bzw. wissenschaftlichen Standpunkten oder den Inhalten von Wikipedia udergl. Meine Absicht damit ist lediglich ein leichteres Verständnis im XH-Unterricht.

SYSTEM

::=

7-Tupel (I, O, Z, A, t, fo, fz)
mit

I.....Menge aller Inputs (Input-Alfabet)
O ... Menge aller Outputs (Output-Alfabet)
Z ... Menge aller Zustände (Zustandsmenge)
A.....Menge der Anfangsbedingungen
t.....die Zeit
fo.....Output-Funktion fo: $I \times Z \times A \times t \longrightarrow O$
fz.....Zustands-Funktion fz: $I \times Z \times A \times t \longrightarrow Z$
(fo, fz: auch zeit-variante Systeme sind Systeme)

(alle Arten äußerer Einflüsse wie Störgrößen oder Parameter sind im Input zusammengefasst)

Die 5 SYSTEM-Typen

:

(1) Black-Box
(2) Netzwerk
(3) Generator
(4) Dynamik
(5) Algorithmus

SIGNAL

::= ist eine Funktion der Zeit ('X-Achse)'

Signal ::= f (t,...)

SPEKTRUM

::= ist eine Funktion der Frequenz ('X-Achse)'

Spektrum ::= f (ω,...)

(der Physiker verzeihe das ω anstelle des ν, aber hier sind wir in der Elektronik...)

DATEN

::= Merkmals-Auspraegungen, Attribut-Werte
(Einzahl: das DATUM)

NACHRICHTEN

::= transportierte Daten

+Daten
+Sender
+Empfaenger

INFORMATION

::= I(m) = ld(1/P(m))

mit
I(m) ... Informationsgehalt der Nachricht m
m ..... Message m
P(m) ... Erwartungswert der Message m
ld() ... 'logarithmus dualis', log2 zur Basis 2

WISSEN

::=

+Fakten

+Regeln

SOFTWARE

::=

alles nicht-Materielle
also
+Daten
+ProgrammCode

Wenn man ganz tief ins Detail geht, stellt man schlussendlich fest:

$E = mc^2$

Materie == Energie!

Und nach Heisenberg

$\Delta x \cdot \Delta p \approx h$

Ortsangabe ist Information, also Software, Masse ist Materie, also Hardware.
Hard- und Software können nicht (beliebig genau) unterschieden werden !!!

Landauer fand die Energie eines [*bit*] Information:

$W = k_B \cdot T \cdot ln2$

W ...Work, Arbeit = Energie
k_B ...Boltzmann-Konstante $1,38^{-23}[J/K]$
T ...abs.Temp $[K]$

Ereignis:

ist die

Änderung eines (Gesamt-) **Systemzustandes**

falls der genaue Ablauf dieser Veränderung interessiert, nennt man es einen

Vorgang

PROJEKT

::=

alles, was man vorher planen muss:
+Vorhaben
+RessourcenLimit=Kostenvoranschlag
+ZeitLimit=Liefertermin
+komplex genug
+anders/neu genug = "innovativ"

TEAM

::=

Fachleute verschiedener Profession + gemeinsames Ziel
"ExpertenTeam"

TEAM-MITARBEITER

... "was ist MEIN JOB?"

+Probleme kommunizieren
+Skills kommunizieren

EXPERTE

... "wie macht man das?"

+der "Macher"
Spezialist mit Erfahrung (von lat. "Erfahrung machen")
"aus Erfahrung wird man klug" == "aus Fehlern wird man klug"

PROJEKTLEITER

... "wer macht mir das?"

+ delegieren
+der "Planer"

STABILITY

::=

A stable system assures that every limited input signal produces a limited response.
begrenzter Input ==> begrenzter Output
zu jedem Epsilon gibt es ein Delta,
sodass fuer alle X aus Input: |X| < Epsilon ==> |f(X)| < Delta

KAUSAL ANTIZIPATIV

>--<

:

Ein System ist **kausal**, wenn es *nicht auf die Zukunft reagiert (unabhängig)*
sonst nennt man es **antizipativ**.
In order to be implementable, any time-dependent filter must be causal:
the filter response only depends on the current and past inputs.
A standard approach is to leave this requirement until the final step.
If the resulting filter is not causal, it can be made causal by introducing an
appropriate time-shift (or delay). If the filter is a part of a larger system
(which it normally is) these types of delays have to be introduced with care
since they affect the operation of the entire system.

LTI

system ::= Linear + Time Invariant
'werkt heute gleich wie gestern'

DETERMINISTISCH

:

Ein System ist *nicht deterministisch* wenn es *richtige Loesungen*
voraus ahnen kann. Manche beschreiben das auch so, als wuerde sich das System
bei jeder moeglichen Entscheidung vervielfaeltigen/klonen, und die Klone,
die einen *unguenstigen Weg* eingeschlagen haben, sterben wieder.

$1k = 2^{10} = 1024$
$1M = 2^{20} = 1048576$
$1G = 2^{30} = 1073741824$
$1T = 2^{40} = 1099511627776$
$10^x = 2^{3.32x(*)}$
zB. $2^{10} = 10^3$, $2^{20} = 10^6$,
$\sqrt{1000} = \sqrt{10^3} = 10^{1,5} = 2^5 =\sim 32,...$ $(31,6)$
(*) $ln(10)/ln(2) = 3,32 = log(10)/log(2) = log_x(10)/log_x(2)$

REGELUNGSTECHNIK:

+"control engineering", "control theory"
+"signal processing"

Regelkreis:

'control loop'

EigenAnstiegsZeit (Oszilloskop)

$t_{an}^2 = t_{skop}^2 + t_{sig}^2$

$t_{skop} * B_{skop} = 0.35$

t_{an} .....gemessene Anstiegszeit
t_{skop} .....EigenanstiegsZeit Oszilloskop
t_{sig} .....wahre Anstiegszeit des Signals
B_{skop} ....3dB-Eingangs-Bandbreite des Oszilloskops

Diode:

$$I_{Anode} = *e^{U_{BE}/m \cdot U_T} - 1$$ wie Ebers-Moll Formel für BJT

mit

$U_T = kT/q$ (=25,875mV @ 300K) "Temperaturspannung"
q ..... Elementarladung 1,602176487E-19 Cb
k ..... Boltzmann-Konstante 1,38065E-23 J/K = 8,61734E-5 eV/K
m ..... Emissionskoeffizient 1..2 (1N4148: 1,89)
Is ..... Sättigungssperrstrom $10^{-}12..10^{-}6A$, (Si: $10^{-}11..10^{-}9A$)

Temperaturabhängigkeit:

(Is ist stärker temperaturabhängig als e^{U_{BE}/U_T})

$$I_S(T) = I_S(T_0) \cdot e^{\left[(\frac{T}{T_0}-1)\frac{U_G(T)}{m \cdot U_T(T)}\right]}(T/T_0)^{(X_{T,I}/m)}$$

gibt bei m=2 -> ca. 0,08/K
-> mal 2 je 9K
-> mal 10 je 30K

$X_{T,I}$... 3 (*Si*)
U_G... 1,12 *Bandgap* (*Si*)
Schottky : $X_{T,I}$ ... 2; U_G ... 0,7 .. 0,8*V*

Switcher (SMPS):

$$U_{ind} = L\frac{dI}{dt}$$
$$\rightarrow$$
$$\frac{U_L}{L} = \frac{\Delta I_L}{\Delta t}$$
$$L = \frac{\Delta I_L}{\Delta t}$$

U_L, U_{ind} .....induzierte DrosselSpannung
L ..... Induktivität $N^2\frac{\mu A}{l} = N^2 A_L$
Δt .....Einschaltzeit der Spannung U_L
ΔI.....Änderung des SpeicherDrossel-Stroms in der Zeit Δt

Drossel:

$L = U_L \Delta t / \Delta I_L$

Drosseln vergleichen:

$L2 = L1 \cdot \frac{\Delta t2}{\Delta t1} = L1 \cdot \frac{f1}{f2}$

zB. L1=47uH, L2=1.1mH
$\rightarrow$
f2 = f1 * L1/L2 = 50kHz * 47u/1.1m = 2,136kHz

BJ-Transistor

BJ-Transistor :

vereinfacht:
$I_C = B \cdot I_B$
$i_C = \beta \cdot i_B$ (Kleinsignal)
$i_C ... \frac{\Delta I_C}{\Delta t}$
$i_B ... \frac{\Delta I_B}{\Delta t}$
wenn $U_{BE} > 0,7V$ und $U_{CE} > 2V$
β PwrBJT: 10 .. 50
β Kleinsignal-BJT: 100 .. 600
β Darlington: 1000 .. 30000
BJT.....Bipolar Junction Transistor
Pwr.....Power (Leistungs-)

Ebers-Moll-Modell

Ebers-Moll-Modell :

$$I_C = I_S(T) e^{\frac{U_{BE}}{U_T}}$$

$U_T = -kT/q$ (=25,875mV @ 300K) "Temperaturspannung"
q.....Elementarladung 1,602176487E-19 [Cb]
k.....Boltzmann-Konstante 1,38065E-23 [J/K] = 8,61734E-5 eV/K
T.....abs.Temperatur [K]
I_S.....ßaturation current- Sperrsättigungsstrom 10E-13 [A]
Temp.Coefficient = ca. -2.1mV/K
$U_{CE,sat} = U_T \cdot ln \frac{B_N(1+B_I)\cdot(B_I I_B + I_C)}{B_I^2(B_N I_B - I_C)}$

Sättigungsbereich

Sättigungsbereich :

$U_{CE,sat} = U_T \cdot ln \frac{B_N(1+B_I)(B_I \cdot I_B + I_C)}{B_I^2(B_N \cdot (I_B - I_C)}$

spez.:
$U_{CE,sat}(Ic = 0) = -U_T \cdot lnA_I \quad \approx 2..20mV$
inversBetrieb:
$U_{EC,sat}(I_E = 0) = -U_T \cdot lnA_N \quad \approx 0,05..0,5mV$ (!!)
A_N ..... $-I_C/I_E$
A_I ..... $-I_E/I_C$
B_N ..... $A_N/(1-A_N) = I_C/I_B$
B_I ..... $A_I/(1-A_I) = I_{E,rev}/I_{B,rev}$

Emittergrundschaltung im Arbeitspunkt

Emittergrundschaltung im Arbeitspunkt
$V_{u,C} = -R_C/(r_E + R_E) = -s' \cdot R_C$
$r_E = U_T/I_C = 1/s$
$r_{BE} = U_T/I_B = (1+\beta) \cdot r_E = \beta/s$
$r_C E = (U_y + U_{CE})/I_C$
U_y ..... Early-Spannung
$\beta = dI_C/dI_B = s \cdot r_{BE} = \frac{r_{BE}}{r_E} - 1$ ..... Basisstrom-Verstärkung
$s = \frac{1}{r_E} = \frac{I_C}{U_T}$ ..... (genau: $s = \frac{1}{r_E}\frac{\beta}{1+\beta}$)
$s' = \frac{1}{r_E + R_E} = \frac{s}{1+s*R_E}$ (E-Gegenkopplg.)
$\alpha = \frac{\beta}{1+\beta} = \frac{I_C}{I_E} < 1.0$ ..... Emitterstrom-Verstärkung

Vakuum-Impedanz

Vakuum-Impedanz :

Das Vakuum hat eine E/H-Impedanz von

$$\sqrt{\frac{\mu_0}{\epsilon_0}} = \sqrt{\frac{4\pi 10^{-7}}{8.85410^{-12}}} = 377\Omega$$

Lichtgeschwindigkeit

Lichtgeschwindigkeit :

$$c = \sqrt{\frac{1}{\mu_0 * \epsilon_0}} = 2.99796 \quad [m/s]$$

$\mu_0 = 4\pi * 10^{-7}$ $\epsilon_0 = 8.854 * 10^{-12}$

56 DIC3 Jahresleitfragen

57 Jahres- Leitfragen

57.1 Fragenkatalog DIC4a$_{xh}$ 18Feb17

1. ADC: flash-ADC, succ.Approx, single/dual slope Converter
2. AudioAmp als Spice Netlist
3. Boundary Scan, JTAG
4. CMOS Inverter, NAND, NOR
5. DAC, R-2R-DAC
6. dB- Rechnung
7. Ebers-Moll-Modell, Early-Formel, Sättigung
8. englische Fachausdrücke der Elektronik
9. FSM
10. Microcontroller- Applikation 'ElevatorControl': FSM-Entwurf
11. Microcontroller- Applikation 'ElevatorControl': Datenmodell
12. Microcontroller- Applikation 'ElevatorControl': webbasiert
13. Microcontroller- Assemblerprogramm
14. Microcontroller- C-Programm
15. Microcontroller- Datenblattlesen
16. Microcontroller- InterruptServiceRoutine
17. Microcontroller- Simulator + Debugger
18. numerische Integration (FE): Methode, Aufloesung/RoundingError tradeoff
19. Schalten von BJT u. MosFET
20. Specs: TTL u. CMOS Gatter, CAN, Ethernet, IIC, RS232, USB
21. Specs: USB PhySpecs +Classes +Transfer +Devices
22. Spice net list
23. TTL-NAND Schaltung auswendig
24. VHDL
25. Widerstandsfarbcode, E12-Reihe auswendig

57.2 Fragenkatalog DIC5a$_{xh}$ 18Feb17

1. Boundary Scan, JTAG
2. CLIL
3. Computer-Architektur, Prozessor-Architektur
4. digital Grundschaltungen, Logic-Specs, Schaltsymbole, Interfacing, Logic Analyzer
5. DFT u. Programmierung
6. DMA + SndCard 'Prozessor'
7. DS-Processor
8. ES: Definition, Anforderungen, Struktur
9. embedded ANN
10. embedded Expertensystem
11. embedded Fuzzy Control
12. embedded Grammatik, BNF, EBNF, arith.Expr
13. embedded RandomGeneration; artificial Noise (Rauschen), Dithering
14. embedded rule based system; Bsp PROLOG
15. embedded RTOS
16. FIR (u. IIR) Filter, Struktur, in FPGA, Simulation, z-Transform, Windowing
17. Halb-/Volladdierer, Addier/Subtrahierwerk
18. Image/Audio File- u. Zahlenformate IEEE-754
19. MAC Operation Hard/Software, Anwendungen
20. numerische Integration (FE Methode) in 'C' u. 'Excel'
21. OS für ES, Eigenschaften, Forderungen
22. PLL + Ne555 (=analog+digital)
23. Realisierung komplexer ES auf FPGA Basis
24. Regeneration verrauschter Signale bekannter Frequenz m. Faltung u. Mehrfachaddition
25. Signal-Korrelation u. Faltung
26. Spice net list
27. VHDL

μMat.Dic5a'2023-XH

1-Grundschaltungen GS:

Schaltnetzentwurf
Schaltwerkentwurf
Normalformen, Verfahren, Minimisierung
ADC, DAC

2-programmierbare logische Schaltungen Hdl:

MAC Hardware in VHDL
FPGA

3-Computerarchitekturen Arc:

MAC-Operation
Computerbusse
von-Neumann Architektur
Harvard Architektur
Interrupts, Pipelining, Cache-Memory, Multicore
RasPi
uC
Cypress pSOC

4-embedded systems ES:

ANN + Synapsenmatrix
Sprache+Grammatik
Expertensystem
Fuzzy Logic
RT POSIX Pgmg

5-digitale Signalverarbeitung Sig:

DCT-DiscreteCosineTransform ≡ 'Fouriers Partnervermittlung'
Fourier-Analyse/Transformation
FIR Filter
Digitalisierung, Quantisierung, Aliasing
Windowing
FE
ANN

DIC-Themenbereiche KU'2223

1. Grundschaltungen (GS)
2. Programmierbare logische Schaltungen (Hdl)
3. Computerarchitekturen (Arc)
4. Embedded Systems (ES)
5. Digitale Signalverarbeitung (Sig)

XHs allerwichtigste 3 Grundregeln:
- Alls, was der XH sag, ischa Bledsinn!
- Arbeit muß[a] Spaß[b] machen.
- XH mag keine Texte.[c]
- "geaht nit" gibts nit. (Ingenieur-Prinzip)
- nur aus Fehlern wird man klug[d]

[a]stammt noch aus alter Rechtschreibzeit
[b]Arbeit macht 'Spaß', wenn mans gern tut und gut kann; erreichtma mit Verbesserungsbestrebungen = gut-sein-wollen
[c]also Diagramme, Skizzen, Screenshots, Bilder, Tabellen udgl. statt *benutzerfreundlich, übersichtlich, selbsterklärend, irgendwie*
[d]Misserfolge geben zu denken

58 se foreword

Die Inhalte dieses Dokuments sind der Phantasie und Utopie entstammende, rein künstlerische Darstellungen bzw. *word art* ohne inhaltlichen Sinn, nicht frei von Rechten Dritter, ohne Eignung für einen bestimmten Zweck, die weder mit Wahrheit, Rechtmässigkeit, Richtigkeit, Brauchbarkeit noch mit deutscher, neuer, diskriminierender[12] oder Wiener Rechtschreibung zu tun haben, und dürfen weder weitergegeben, noch veröffentlicht, noch mechanisch oder elektronisch verarbeitet oder gespeichert werden. Für Inhalte quellverwiesener, zitierter oder verlinkter Werke/Inhalte kann keine Verantwortung/Haftung übernommen werden.

Als Vorlage verwendi die
Diplomschrift- Vorlage
von AV YH
(s. moodle2 und obelix) mit geringen Anpassungen in Schriftstil (Arial) u. Layout (A4) . . .

Dem unerreichbaren, großen
Vorbild und LaTeX-Mentor
Prof. Dr. Robert SALVADOR
gewidmet und gedankt!

58.1 der *obelix* Account

Der *obelix* ist mein jahrzehntelanger, schülergewidmeter Linuxrechner im Schulnetz. Er hat die IP `http://10.10.63.61/` und den *hostname obelix*, ist mit dieser URL aber nicht im Schul-DNS eingetragen (das wird zwar vielfach gewünscht, verweigere iXH aber bewusst, damit die Teilnehmer das Arbeiten mit IP statt URL sehen/lernen *müssen*). Das Ding ist im Unterricht und Alltag vielfach nützlich, als Webserver mit U-Material, Datenblattspeicher, Programmierwerkzeug, LAN-Experiment-Partner, Onlinedoku-Bibliothek (man-pages), WebDatenbankserver(MySQL), Schaltungssimulator(SPICE), DTP-Tool(LaTeX), Abgaben-Ordner und mein pers. zunehmend erforderliches *extended Memory*.

58.1.1 Aufbau of se user name

am *obelix* `http://10.10.63.61/` hat man einen eigenen Account;
der Benutzername (zB. "n15afffvv") besteht aus

```
  n = EL-Abt. (w=WI Abt.)
 15 = Jahr 2015 des HTL-Einstiegs
  a = a-Klasse
fff = Familienname, Buchst.1--3
 vv = Vorname, Buchst.1 u. 2
 zB. "n15amaifr", "n16cxmilu" usw.
```

(Umlaute und ß werden zweibuchstabig ersetzt ö=oe, ß=ss,..., auch *sch* durch "x"; das Standartpasswort lautet "htl" (klein!) — ändere es raschest!

[12] *[...] Das Wort Diskriminierung stammt von dem aus dem lateinischen Verb discriminare 'trennen', 'absondern', 'abgrenzen', 'unterscheiden' im Spätlateinischen abgeleiteten Verbalsubstantiv discriminatio 'Scheidung, Absonderung.' Das Verb diskriminieren wurde im 16. Jahrhundert in der wertneutralen Bedeutung 'unterscheiden', 'sondern', 'trennen' ins Deutsche entlehnt und ist dort seit dem 19. Jahrhundert kontinuierlich belegt [...]* (de.wikipedia.org 01Nov15)
'to discriminate': unterscheiden, einen Unterschied machen, unterschiedlich behandeln [...] (www.dict.cc 01Nov15)
Als Diskriminator (lat. discriminare = trennen, scheiden) werden verschiedene Geräte oder Baugruppen innerhalb von Geräten der Nachrichtentechnik, Elektronik und Messtechnik bezeichnet. Diskriminatoren dienen zur Auswertung analoger oder digitaler Signale [...] (de.wikipedia.org/wiki/Diskriminator 01Nov15)
Die Formulierung »Schüler und SchülerInnen« macht einen Unterschied, indem sie extra genannt werden, »diskriminiert« also; zudem sind Großbuchstaben mitten im Wort *nicht* deutsch! Es verbindet also »SchülerInnen« mit groben Rechtschreibfehlern. Darin ist mE. insgesamt eine ganz ganz große Beleidigung und Geringschätzung der gesamten Weiblichkeit zu sehen — mE. diskriminierende Nichtrechtschreibung.
Eine Titelschreibweise wie Mag[a] ist mW. normenwidrig, der Titel ist *nichtexistent* und vor allem ist er eine diskriminierende Frauenkennzeichnung (*'Frauen sind kennzeichnungspflichtig'*) wie »Intel inside«, vergleichbar dem »Judenstern« im dt.Reich; ist das »verbotene Wiederbetätigung« ?

58.2 remote Login zum Linux Server (zB obelix)

Wie stelle ich eine *ssh*- Remote-Login Verbindung zum *obelix* her:

vom Linux:

```
ssh n18bxxxyy@10.10.63.61
password: 'htl'
'cd public_html'
```

Achtung:
Linux ist **"case-sensitive"**
Gross/Kleinschreibung
ist **relevant!**

vom MacOS:

Systemkonsole starten
dann wie auf Linux: `"ssh n18bxxxyy@10.10.63.61"`
password: 'htl'
'cd public_html'

am Winzigweich:

`"putty 10.10.63.61"` (oder zB TeraTerm)
username: 'n18bxxxyy'
password: 'htl'
'cd public_html'

*seit Version 10.?? habe 'powershell' auch ein (reduziertes) "ssh"

(mit xxx ... erste 3 Nachnamensbuchstaben,
yy ... erste 2 Vornamensbuchstaben)

58.3 C-Programmcodeeingabe am 'obelix'

nach erfolgreichem 'remote login' mit 'ssh' oder 'putty':

im Webseitenbereich:

1. *'cd ~/public_html'*
2. *'mkdir -pv gibNichtXH/meinprojekt'*
3. *'cd gibNichtXH/meinprojekt'*
4. *'nano meinPgm.c'*
5. (C-Programmcode eintippen)
6. (abspeichern mit Strg+O)
7. 'nano' verlassen mit Strg+X

Achtung:
Linux ist
"case-sensitive"
Gross/Kleinschreibung
ist **relevant!**

ausserhalb
Webseitenbereichs:

1. *'cd ~'*
2. *'mkdir -pv gibNichtXH/meinprojekt'*
3. *'cd gibNichtXH/meinprojekt'*
4. *'nano meinPgm.c'*
5. `(C-Programmcode eintippen)`
6. `(abspeichern mit Strg+O)`
7. 'nano' verlassen mit Strg+X

Es gibt **kein**
Datei-
"WIEDERHERSTELLEN"!

58.4 C-Code XH- Abgabe ins 'gibXH' am 'obelix'

Alles, was DU ins `public_html/gibXH` ablegst, wird bewertet!

Nach erfolgreichem 'remote login' mit 'ssh' oder 'putty',
erfolgter Programmcodeeingabe und Test:

gibXH-Abgabe:

1. *'mkdir -pv ~/public_html/gibXH/meinprojekt'*
2. *'cp -v meineAbgabe ~/public_html/gibXH/meinprojekt'*

58.5 C-Compilieren am 'obelix'

nach erfolgreichem 'remote login' mit 'ssh' oder 'putty'
plus C-Programmcode eintippen:

ohne Zusatzlibrary:	*'gcc -o myBin meinPgm.c'*	
incl. 'math'-library:	*'gcc -o myBin meinPgm.c -lm'*	f. sin(), exp(), pow() udgl.
incl. 'pthread'-library:	*'gcc -o myBin meinPgm.c -lptread'*	(pthread ... Multithreading)
incl. 'rt'-library:	*'gcc -o myBin meinPgm.c -lrt'*	('rt' ... RT, real time)
incl. math u. RT:	*'gcc -o myBin meinPgm.c -lm -lrt'*	

58.6 compiliertes C-Programm ausführen am 'obelix'

nach erfolgreichem 'remote login' mit 'ssh' oder 'putty'
plus C-Programmcode Eintippen plus Compilieren:

im current directory:	*'./myBin'*
in '/pfad/zum/dir':	*'/pfad/zum/dir/myBin'*

58.7 C++ Compilieren am 'obelix'

nach erfolgreichem 'remote login' mit 'ssh' oder 'putty'
plus Programmcodeeingabe (in diesem Bsp. ins File 'meinCCpgm.C'):

ohne Zusatzlibrary:	*'g++ -o myCCbin meinCCpgm.C'*	'gcc' und 'g++' sind derselbe Compiler; das 'g++' sagt ihm lediglich, dass er C++ zu übersetzen hat und C++ Libraries einbindet.

58.8 CIFS Windows Netzlaufwerk mappen

(1) den Windows-Explorer starten
(2) Netzlaufwerk verbinden
(3) share: \\10.10.63.61\n18bxxxyy
(4) unter anderem Benutzernamen (sonst nimmt es den Win- User)
(5) Benutzername: n18bxxxyy
(6) Kennwort: htl
(7) fertigstellen

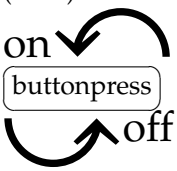
58.9 Password

Das Linux-User-Passwort ändert man mit *passwd <username>*
und das Windows-Netzlaufwerk-Password (samba) mit *smbpasswd <username>*

59 Glossar

liebe Leserin, lieber Leser, Du musst selber erkennen, welcher der Einträge ernst gemeint ist (**schmunzelgrins**).

AD activity diagram (UML)
Vorgänge (activities) in Ovalen, Zustände (state) sind Pfeile.



verwandt zum *Flussdiagramm, flow chart, work flow*

BD Blockdiagramm (nicht UML)

BTL Bridge Tied Load = Brückenschaltung (zB. H-Brücke)

CD component diagram (UML)

CLI CommandLineInterpreter

DD deployment diagram (UML)

ER entity-relation modell/method (nicht UML)

ES Embedded System

GP General Purpose (zB. GP-Processor, GP-OS)

GP-OS General Purpose OS, siehe 'MP-OS'

GUI Graphical User Interface (icons to 'klick')

KD class diagram, Klassendiagramm (UML)
(heisst in UML eigentlich *static structure diagram*

MMU Memory Management Unit, Hardware f. Virtual Memory Management (VMM), ≙ der *'Adress-Bscheisser'*

MP-OS Multi-Purpose OS, allgemein verwendbares OS ohne Spezialisierung; zB. die verbreiteten Desktop-OS von M$, Apfel, Gugel, Sisko, Opensoars.

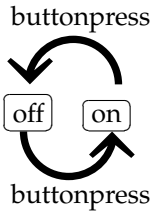
OS Operating System; Software, die bei Programmwechsel im Rechner- Arbeitsspeicher verbleibt und oft erforderliche Funktionen bereitstellt

PEMDAS Rechenregel *"Pleas Excuse My Dear Aunt Sally" steht für* 'Parethesis before Exponentiation, Multiplication, Division, Addition, Subtraction'

RT Real Time, dt. Echtzeit (im Gegensatz zu Simulationszeit: zB. 5ns Schaltungssimulation dauern 2 Sekunden Berechnungszeit; zB. Kino-Spielfilm dauert 3 Std und zeigt einen Zeitraum von 6 Jahren)

RTOS Ein OS, welches Real Time stark beachtet, indem es genaue Zeit und Timer bereitstellt sowie Task/Thread-Prioritäten und -Scheduling umsetzt. Ein RTOS ist umso RT, je rascher (Latenz, latency) sein Taskmanager (Scheduler) auf Ereignisse (Timerablauf, Inputsignale) reagiert. (ein MP-OS beachtet RT-Erfordernisse erst, wenn der Prozess sowieso *an der Reihe* ist)
Eine dazulink-Library wie zB. *'RTuinOS'* ist jedenfalls kein RTOS, weil es ja nicht einmal ein OS ist.

StD statechart diagram, Zustandsdiagramm (UML)
Vorgänge (activities) sind Pfeile, Zustände (state) in Ovalen.



verwandt zum *bubble-diagram*

SqD sequence diagram (UML)
(des mit di Objekt-Lebenslinien)

UCD use-case diagram (UML)
(des mit di Strichmandln)

UI User Interface

VFS Virtual File System

VMM VirtualMemoryManagement
= MMU+Auslagerung(swapping)

AAK alles alter Kas

Abfallflanke, fallende Flanke falling edge

Algorithmus von *'al-Chwarizmi'*, latinisiert Algorismi (geboren um 780; gestorben zwischen 835 und 850), war ein choresmischer Universalgelehrter, Mathematiker, Astronom und Geograph während der abbasidischen Blütezeit im Frühmittelalter. Er stammte zwar aus dem zentralasiatischen Choresmien, verbrachte jedoch den größten Teil seines Lebens in Bagdad und wirkte dort im "Haus der Weisheit", der berühmten Hochschule von Bagdad. Von seinem Namen leitet sich der Begriff **'Algorithmus'** ab. Chwarizmi, der sich mit Algebra als elementarer Untersuchungsform beschäftigte, gilt als einer der bedeutendsten Mathematiker. Auch leistete er bedeutende Beiträge als

Geograph und Kartograph, dies auch durch Übersetzungen aus dem Sanskrit und dem Griechischen.

Anstiegsflanke, steigende Flanke rising edge

Arbeitspunkteinstellung biasing

Ausschalt-Verzögerung turn off delay t_d, off

Auslieferungszustand klassisch: Zustand in dem eine Sache ausgeliefert wurde. Heute, im Softwarezeitalter: Zustand des dem Software-Hersteller ausgeliefert seins...

bleeder resistor Lastwiderstand zum "ausbluten" = entladen nach Trennung der Energiezufuhr

coprime relativ prim

cnt count (Zählerstand)

ctr counter (Zähler)

cvtr converter (Wandler)

Differential von the derivative of

Durchlaufzeit propagation delay (time) t_d

differenzieren, ableiten derive, differentiate

eddy current Wirbelstrom (eddy = Wirbel)

Einschalt-Verzögerung turn on delay t_d, on

inductive kickback Selbstinduktion

Lastenheft Das Lastenheft (teils auch Anforderungsspezifikation, Anforderungskatalog, Produktskizze, Kundenspezifikation oder englisch Requirements Specification genannt) beschreibt die Gesamtheit der Anforderungen des Auftraggebers an die Lieferungen und Leistungen eines Auftragnehmers. Es ist z. B. im Software-Bereich das Ergebnis einer Anforderungsanalyse und damit ein Teil des Anforderungsmanagements.

Das Lastenheft kann der Auftraggeber in einer Ausschreibung verwenden und an mehrere mögliche Auftragnehmer verschicken. Mögliche Auftragnehmer erstellen auf Grundlage des Lastenheftes ein Pflichtenheft, welches in konkreterer Form beschreibt, wie der Auftragnehmer die Anforderungen im Lastenheft zu lösen gedenkt. Der Auftraggeber wählt dann aus den Vorschlägen den für ihn geeignetsten aus. Die Anforderungen in einem Lastenheft sollten durch ihre Formulierung so allgemein wie möglich und so einschränkend wie nötig formuliert werden. Hierdurch hat der Auftragnehmer die Möglichkeit, optimale Lösungen zu erarbeiten, ohne durch zu konkrete Anforderungen in seiner Lösungskompetenz eingeschränkt zu sein.

Im Rahmen eines Werkvertrages oder Werkliefervertrages und der dazugehörenden formellen Abnahme beschreibt das Lastenheft präzise die nachprüfbaren Leistungen und Lieferungen.

Lastenheft Quelle: http://de.wikipedia.org/w/index.php?oldid=127249582 Bearbeiter: Abubiju, Afrank99, Aggi, Ahellwig, Alexander.stohr, Arx, Avron, Axel Naumann, BJ Axel, BK, Borobo, Complex, Croesch, Diba, Edoe, Enno76, Eventkultur, Filzstift, Fisfra, Fish-guts, Fladi, Flo 1, Gerbil, Gerhard de, Ghw, Giftmischer, Goliath613, HaSee, HaeB, Haeber, Hardenacke, He3nry,
Hjpospie, Holmium, Howwi, Igelball, Ines.lehmann, J.Ammon, Jü, Kaninchenohr, Kgfleischmann, Kku, Koppi2, LKD, Leiblachtaler, Logograph, Lohan, Louis Bafrance, Ma-Lik, Markobr, Marrrci, Martin1978, Meinungsfreiheit, Methossant, MyBenutzername, Niemeyerstein, Nirakka, Ocrho, Ot, Pallasathena, Peter200, Pittimann, Polluks, ProfessorX, Quedel, RacoonyRE,
Roo1812, Saehrimnir, Sascha Claus, Scooter, Se4598, Semper, Siehe-auch-Löscher, Softsheep, Sparti, Spuk968, Sterling, Tambora, Tango8, Tkarcher,
Tobias K., Ucc, Udm, Ute Erb, VÖRBY, WAH, WIKImaniac, Westiandi, WhiteCrow, Wiki-observer, Wiki4you, Xwolf, Yotwen, 177 anonyme Bearbeitungen

Pflichtenheft Das Pflichtenheft beschreibt in konkreter Form, wie der Auftragnehmer die Anforderungen des Auftraggebers zu lösen gedenkt – das sogenannte wie und womit. Der Auftraggeber beschreibt vorher im Lastenheft möglichst präzise die Gesamtheit der Forderungen – was er entwickelt oder produziert haben möchte.

Erst wenn der Auftraggeber das Pflichtenheft akzeptiert, sollte die eigentliche Umsetzungsarbeit beim Auftragnehmer beginnen.

Pflichtenheft Quelle: http://de.wikipedia.org/w/index.php?oldid=127249625 Bearbeiter: 1-1111, A.Savin, APPER, Abubiju, Aggi, Aka, Alnilam, Alraunenstern, Andre Riemann, Andreas 06, Andruil, Antimaterie, Astrofreund, Avron, BJ Axel, Batty, Benatrevqre, Bernburgerin, Bernd vdB, BuC-Projekt, Christian Specht, ChristophDemmer, D, DasBee, Dberlin, Deki, Der.Traeumer, DerHexer, Diba, Drahreg01, Dundak, Engie, Eventkultur, Exoport, Felix Stember, Fleshgrinder, Flominator, Frasta, Friedemann Lindenthal, GDK, GNosis, GS, Gerbil, HaeB, Haeber, Heidenodawas, Howwi, Hybridbus, Istandil, Iste Praetor, Jergen, Jpp, Jü, Kku, LKD, Luigi bosca, Lupussy, Ma-Lik, Marrrci, Methossant, Michaki, Mnh, Morten Haan, Mps, MyBenutzername, Mühsam, Niemeyerstein, Nina, Nocturne, O.Koslowski, Ocrho, OecherAlemanne, Ot, Pajz, PeterFrankfurt, Pittimann, Pmatu, Ra'ike, Regi51, RichiH, Roo1812, STBR, Saehrimnir, Saenger84, Saibo, Sallynase, Schandi, Schusch, Sechmet, Sinn, Softsheep, Sparti, Spuk968, Staro1, Stefanobasta, Stern, Tambora, Terrestria, The-Me, Thomas Springer, Thomasxb, Thornard, Tivi, Tofra, Totschläger, Traxer, UlrichJ, Umweltschützen, Uwe Keim, WAH, WIKImaniac, Wahldresdner, Westiandi, WhiteCrow, Wiki-observer, Xorx, YMS, YourEyesOnly, Zahnradzacken, Zeit ist unendlich, 225 anonyme Bearbeitungen

Betriebswirtschaft keine Ahnung, aber sicherlich keine wissenschaftlich professionelle Form von organisierter Geldgier.

benutzerfreundlich inhaltslose Floskel ohne jeden Aussagewert. *Benutzerfreundlich keit* ist zwar gut, aber man weiss nicht, was es ist. Es gibt keine Maßeinheit (Liter?) und keine Messmittel — Benutzerfreundlichkeitsmessgeräte sind unbekannt!

übersichtlich saubermännische Schönrede des eigenen Sauhaufens.

selbsterklärend schwächstmögliche Ausrede für die eigene Unfähigkeit zu Planung, Protokollierung, Berichterstattung und Unkenntnis jeglicher Entwurfs- u. Darstellungsmittel.

irgendwie meta-sprachliche Variable, die zum Ausdruck bringt, dass man keine Ahnung hat.
bei Medizinern auch

ma kunnt probieren . . .

miassma schauen . . .

etwas auch Sprachvariable, vgl. *Ding, Zuig, Plunder, Kram* usw.

pro lateinisch 'für', 'nach vorn'/'voraus' zB. programm: für später beschrieben, pro-jektor: nach-vorne-werf-Gerät das pro-cedere: das Vorgehen, pro-cess: voran schreiten progress: voraus-schreitend (fortgeschritten)

post lateinisch nachher, im Nachhinein, zB. post mortem: nach dem Tod, post medridiem: nach Mittag

pre lateinisch 'vorher' zB. pre-justiz: Vorurteil, pre-potent: überheblich

Halbleiter halbierte Leiter;
auch: Unzureichend fähige Person in leitender Funktion
in Chemie/Physik: Chemische Elemente in der Haupt-Diagonale des Periodensystems der Elemente von H bis Te (H, B, Si, Ge, As, Sb, Se, Te); sie können sich sowohl als Metalle als auch Nichtmetalle *"benehmen"* (zB. H als H_3O^+ oder OH^-, C als Diamant oder Graphit).

Scheißhausparole gerne geglaubtes, aber unwahres Gerücht. Den Ausdruck hat mir ein mehrjährig im Ural kriegsgefangener Nachbar vor 25 Jahren beigebracht; sie nannten das Gerücht *morgen kommen wir nach Hause* eine "Scheißhausparole", mglw. deshalb, weil nur im Klo heimliches Weitersagen möglich war.

Datenkapsel data encapsulation

Datenbanknormalisierung data base normalization, schrittweises DB-Entwurfs-Verfahren

Ding In der allgemeinen Technik und besonders der Elektronik unverzichtbares *Zuig* zur Verwendung für *Eppas*. Metasprachliche Variable also.

Dings
1. Ungenaue (populärwissenschaftliche) Beschreibung eines *Ding* (s.o.)
2. Person ohne Identität (*Herr/Frau Dings, du weißt schon. . .*)

Bei Ertönen des Wortes *Dings* wissen alle sofort "*ah, das ist ein gut ausgebildeter, kompetenter Techniker, der kennt die Fachausdrücke*"

dB-Ding tatsächlich für das *Spektroskop* gebrauchter Fachausdruck eines Maturanten 2012/13

DA Diplomarbeit (bei uns zur Erlangung der techn. Berufsreife und des Titels "Ingenieur")

da lokalitätsbezogene Präzisierung eines *Dings* (ein *Dings**da***)

DS Diplomschrift - das schriftliche Dokument zu einer DA

Zuig Zeig

Zeig Zeug

Zeug auch metasprachliche Variable (*Wildcard*)

Eppas etwas

Diode weibliche Form von Ding mit zwei Anschlüssen.

Triode weibliche Form von Ding mit drei Anschlüssen.

Tetrode weibliche Form von Ding mit 4 Anschlüssen.

Pentode weibliche Form von Ding mit 5 Anschlüssen.

Hexode weibliche Form von Ding mit 6 Anschlüssen.

Heptode weibliche Form von Ding mit 7 Anschlüssen.

Oktode weibliche Form von Ding mit 8 Anschlüssen.

ampelifier control circuit for traffic lights

Elektrohnik Stromsucherei ohne Oszilloskop

Elektroniker ohne Oszilloskop elektrischer Ochs ohne Aussicht.

Elektronikerin undiskriminiertes Gegenteil von Elektroniker

IC Integrated Circuit

IP intellectual property

IP intrusion protection

IP internet protocol

IoT Internet-of-Things

IoE Internet-of-Everything (im Ernst - is ka Scherz!)

VHDL *VLSI Hardware Description Language*, aber auch *very high speed Hardware Description Language* oder:
very high speed integrated circuit hardware description language
Anscheinend einer der Fälle, wo die Abkürzung geläufig, aber keine Bedeutung bekannt, wie etwa auch bei *GNU, PHP, . . .*

VLSI very large scale integration

FPGA Field Programmable Gate Array
super Ding! Vor allem, wenn man es selber programmieren kann.
Zurzeit werden der Reihe nach Scheckkartengrosse (ca. 80mm X 60mm) Kleinstcomputer-Boards mit in FPGA geflashten ARM-CPUs und Linux-OS entwickelt, etwa *RaspberryPi*, *BeagleBone* oder *redPitaya*.
Besonders rechenintensive Aufgaben löst man vorteilhaft als ins FPGA *geflashte*, hardwaremäßige Rechnerstruktur.

geistige Fußgänger Denkgeschwindigkeit << 5km/h

Beeinträchtigung ≠ Eintracht

schreckhaft ≠ schrecklich

Dame ≠ dämlich ≠ damisch

Heim ≠ heimlich ≠ unheimlich

wie sagen kleine griech. Mathematiker, wenn sie aufs Klo müssen?:
- "α, α!"

Milestone (mile stone)
Termin
++plus++
Produkt
++plus++
Testplan

Es genügt nicht, mit ein oder zwei Worten nur einen unklaren Fertigungsgrad zu einem Test-Termin zu umschreiben, und sich dann niemand weiter darum kümmert. (Das wird in Fachkreisen "*Qualität hineinprüfen*" genannt)

Nein.

Es ist festzulegen, wer die Einhaltung feststellt, exakt wie die Prüfung zu erfolgen hat (idealerweise mit Messgeräten gemäß Testplan), was mit den Mess-/Prüfergebnissen geschieht, und wie es dann weiter geht (bes. bei Fehlerfeststellungen).
Diplomarbeitskandidaten melden gern *ich bin schon fast fertig* (wenn sie 29 Seiten Quatschtext zammkopiert haben und meinen, 30 zu brauchen). Was heißt das? Wieviel Prozent ist *fertig*? → Definitiv heißt das gar nichts! Die einzige quantitative Definition von *fast* ist mir aus der Mathematik mit *fast alle = es gibt eine endliche Anzahl von Ausnahmen* bekannt.

Serverraumgestaltung server room design

Specs specifications, geforderte bzw. gewährleistete Spezifikationen; meist Kenndaten, Leistungswerte o.dergl. . . . *meet the specs*. . .

uC micro controller; Intel hatse zerscht *single chip computer* genannt. Deutsche Schreibweise: Mikrokontroller (bei Siemens gelesen; klingt mE. wie *Speckkneydl*)

Arduino is ka uC; es is a ganzes Modul und ausserdem ham die nuijen ARM-Dinger drauf mit so Linux-OS, richtige *cruncher* also.

UndiXH wollt zerscht a selbergmachtes uC-Board; ez weissi nimmer. . .

number cruncher rechenleistungsfähige Prozessoren oder Computer

7seg Sieben-Segment-Anzeige (die haben incl. Dezimalpunkt 8 Segmente :-)) (Bezugsquelle: Vor Jahren haze der Neuhold mal billig verramscht; iseh grad, dasser den Krempel allmno 10 Stk um 2.50€ una vierstellige um 50 Cent verklopft; und 2x16 nachschmeiss-LCD's; wenn Ihr zagzag a Sammelbestellung organisierts, hapzase next week)
(se ham nix organisiert)

coarse heißt *grob*; grob einstellen (*coarse tuning*)

fine ist dann die *Feineinstellung* (*fine tuning*); der fein-einstell-Regler heißt auch *vernier*

Ersatzschaltbild zusammenfantasierte Schaltung zur Berechnungs-Vereinfachung: Was die Signal-Verarbeitung beeinflusst, wird durch *lineare* Bauteile ersetzt (wir behaupten: "*Signale sind so klein, dass der benutzte Teil auch nichtlinearer Bauteilkennlinien so gut wie gerade, also linear, wirken*")
Alles Andere lässt man eiskalt weg!

nested loop verschachtelte Schleife

Sitzredakteur wenn Medien mit widerrechtlich erworbenem Material viel Geld verdienen, aber trotzem kaum gestraft werden wollen, dann bezahlen sie einen Arbeitslosen oder Sandler dafür, dass er offiziell als Redakteur auftritt und dann eben *einsitzen* geht. So nennt man ihn "Sitz- Redakteur". fällt mir grad ein, weils im BC bringen, dassma em Schumi die Krankenakte gestohlen hat.

Schneewittchen engl. *'snow white'*

ab und zu engl. *'every then and now'*

ewig engl. *'forever and 3 days'*

DFT discrete fourier transform

FFT fast fourier transform, runtime optimized DFT algorithm (for 2^n sized signal sample blocks only)

DSP digital signal processing

GP general purpose

MP multi processing

PDF portable document format, (Programmiersprache!)

PLC programmable logic control

PLC powerline communication, HF via unabgeschirmte, unsymmetrische Stromleitungen mit naturgemäß immenser Störproblematik

PoE Power over Ethernet

PoL Point of Load - Spannungs-/Stromregler werden nahe zum Verbraucher (load) angeordnet

PS power supply - Stromversorgung

PS PostScript Drucker-Sprache (Programmiersprache!)

SM stepping motor - Schrittmotor

SM Synapsen-Matrix

SMP symmetrical multiprocessing

SMPS switched mode power supply - Schaltnetzteil

SPS engl.'PLC' programmable logic control (speicherprogrammierbare Steuerung)

Tastverhältnis duty cycle (bei PWM pulse width modulation, zB.f.→SMPS)

Aufwand etwas Aufwendiges (oder Aufwändiges?)

Gemse gemsige (oder gämsige? oder gamsige?) Gams

Betreuling betreute Person oder PersonIn

Prüflinge zu prüfende Personen, PersonInnen oder Dinge

Kandidatling Kandidat oder KandidatIn

Tischling Tischler oder TischlerIn

Frischling schweinisches Baby

verschachtelte Schleife nested loop

Verzögerungszeit delay time, Gatterlaufzeit: propagation delay

C und kleinweich trenn sofort das SHARP vom C . . .

some BJT
```
Ein Widerstand am E-Anschluss
spart dir den Verbrenn-Verdruss
```

kurz und schluss Die Story mit der Batterie . . .

Kurzfassung:

Man kann in der Ausbildung nichts
kontraproduktiveres
tun, als Singen und Summen
im und um den Unterricht zu
unterbinden, weil
das die Hirnentwicklung blockiert

60 Musik und Gehirnentwicklung

- ⊕ Musik führt zu **Gehirnwachstum**
- ⊕ in Musikerhirnen ist **mehr graue Substanz** vorhanden
- ⊕ Musizieren trainiert höchste Konzentration: Hören, Sehen und Bewegung finden gleichzeitig statt, wodurch **neue Nervenverbindungen** geschaffen werden
- ⊕ **Singen ist das intensivste Hirntraining** überhaupt
- ⊕ Singen trainiert das **2-te Sprachzentrum** (Broca-Areal) zusätzlich zum Wernicke-Zentrum
- ⊕ absichtsloses, unbekümmertes Singen und Summen hat den größten Nutzeffekt für die **Entwicklung von Kindergehirnen**
- ⊕ Musik verbessert **Durchblutung, Immunsystem und Schmerzempfinden!**
- ⊕ Bereits ab der 22. Schwangerschaftswoche ist die Hörschnecke bei Ungeborenen vollständig ausgebildet, sodass schon Ungeborene Musikstücke wiedererkennen und darauf reagieren
- ⊕ in Musikerhirnen ist die Verbindung zwischen rechter und linker Gehirnhälfte, das sogenannte Corpus callosum, deutlich kräftiger ausgebildet. Die **Gehirnhälften arbeiten stärker zusammen**

Dieses sollte ein Beitrag zum Thema *Musik und Gehirn* werden, scheint nun aber auszuufern. Anekdote: Vor gut einem Vierteljahrhundert hatte iXH mal auf einer Almhütte eine Busreisegruppe mit so kulturlosem Kommerzschmarren, den unsere südschwedischen Nachbarn ahnungsloserweise *Tirolerlieder* nennen, zu unterhalten. Ziemlich bald fiel ein Pensionist auf, der von all diesem Gejammer alle Strophen mit fehlerfreiem Text vorbrüllte. In a Pause hunami zu dene gsetzt, aber de Olle hat nix mit mir gredet! Schon fiel mir seine 'sie ins Wort und meinte, Ihr Gemahl könne seit einem Schlaganfall nimma sprechen. iXH denk mir *was redet die Tante, der sing ja wie Zeisig* – se muass meine Gedanken diagnostiziert haben, denn glei hatse gmeint *das Sprechzentrum hat man ja doppelt!* Verstanden hat mi der Olle, aber gredet hat nur Oma. Gsungen hat dann wieder nur Opa. Ob der mit mir reden konnte, wenn er es xungen hätt, oder nur was auswendi glerntes aussa bring, weissi nit.

Ez schreibi scho tagelang da dran umma und kimm nu schleppend weiter. Ins Hirn geschissen haz mir des Thema, weil des *Resistor-Keyboard* aus der *Krawallgarage* wieder auftaucht is und ez bei der 4cHEL ganz hinten aufan Tisch steaht (liegt)

Quellen:

Max-Planck-Institute für empirische Ästhetik in Frankfurt am Main (MPI EA) und für Kognitions- und Neurowissenschaften in Leipzig (MPI CBS)

Prof. Dr. Gerald Hüther, Leiter der Zentralstelle für Neurobiologische Präventionsforschung der Universität Göttingen und Mannheim/Heidelberg

Prof. Dr. Eckart Altenmüller, Professor für Musikphysiologie und Musikermedizin an der Hochschule für Musik, Theater und Medien Hannover

Bernd u. Daniela Willimek, Hochschule für Musik Karlsruhe u. Universität Rostok

Vinoo Alluri, Studienleiter Universität Iyväskylä, Finnland

Gottfried Schlaug, Harvard Medical School, Boston

Josep Marco-Pallarés Universität Barcelona

Psychologin Frances Rauscher, University of Wisconsin Oshkosh

Physiker und Neurobiologe Gordon Shaw, University of California

Psychologen George Caldwell und Leigh Riby, Glasgow Caledonian University

Musikpsychologe Stefan Koelsch, Freie Universität Berlin

Isabell Peretz, Université de Montréal, Kanada

Jaak Panksepp, Emeritus Bowling Green State University, Ohio

Julia Grieser-Painter, Oregon Health & Science University

- Bildgebende Verfahren geben Einblicke, was im Gehirn beim Musikhören/machen vor sich geht. Die Bilder lassen erahnen, wie viele Hirnareale beim Musizieren beteiligt sind.
- Musik in der Schule - macht Musik schlau?
  Eine der aktuellsten Studien veröffentlichte der Neurowissenschaftler Sam Norman-Haignere gemeinsam mit seinen Kollegen des Massachusetts Institute of Technology im Fachmagazin "Cell Biology". Dass Team konnte nachweisen, dass es in unserem Hörkortex Nervenzellen gibt, die einzig und allein auf Gesang spezialisiert sind. Sie reagieren weder auf gesprochene Sprache noch auf Instrumentalmusik. Das hingegen leisten Neuronen in unmittelbarer Nachbarschaft. Dieser Arbeitsteilung kamen die Forschenden erst auf die Spur, als sie feiner auflösende Untersuchungsverfahren anwenden konnten. Denn mit einer herkömmlichen Magnetresonanztomografie (MRT) konnten sie zwar sehen, dass die Areale der Hörrinde arbeiten, dass die dicht beieinanderliegenden Zellgruppen aktiv sind, jedoch nicht genau, welche bei welcher Art von Reiz. Ähnlich wie Bildende Kunst oder Sport wird immer wieder der Nutzen des Musikunterrichts in der Schule angezweifelt – jedoch völlig zu Unrecht. Denn der Musikunterricht leistet einen großen Beitrag in der sozialen Entwicklung von Kindern. Die sog. Bastian Studie bestätigt, dass sich die Sozialkompetenz der Schüler durch das gemeinsame Musizieren merklich steigern ließ. Die Langzeitstudie, die an mehreren Berliner Grundschulen durchgeführt und im Jahr 2000 veröffentlicht wurde, stellte außerdem weitere positive Effekte fest:
  - Ausgrenzung von Schülern nahm merklich ab
  - Ablehnung durch Mitschüler reduzierte sich drastisch
  - Generell ruhigeres und aggressionsfreieres Klima

Zurückzuführen ist dies unter anderem darauf, dass beim gemeinsamen Musizieren viel genauer aufeinander gehört und eingegangen werden muss. Eine verbesserte Wahrnehmung des Stimmklangs führt z.B. auch dazu, dass die Kinder die Stimmung ihrer Mitschüler schneller erfassen und darauf reagieren konnten. Auch die Ausschüttung von Oxytocin während des gemeinsamen Musizierens, dem Bindungshormon, fördert den Zusammenhalt der Schüler und sorgt für ein besseres Klima innerhalb der Klasse. Zudem hat das Musizieren einen direkten Einfluss auf die Motivation und die Konzentration der Kinder, da durch den schönen Klang sofort ein belohnendes Ergebnis entsteht. So kommt es auch zu einer vermehrten Ausschüttung von Dopamin, dem Belohnungshormon. Die Kinder sind glücklicher und motivierter.

- Mythos Mozart-Effekt
  Immer wieder wird behauptet, dass Musik klug macht. In den 90er-Jahren sorgte eine Studie für Aufsehen, die den sogenannten Mozart-Effekt postulierte: Das Hören von Mozart-Musik führte gemäss der Studie zu einem besseren Resultat im Intelligenztest. Heute geht man allerdings davon aus, dass die Verbesserung nicht direkt mit den Klängen Mozarts zu tun hatte, sondern durch den Konzentration und Wohlbefinden steigernden Präsenz-Effekt ausgelöst wurde, der auch bei anderen Musikgenres eintritt, sofern es sich um einen bevorzugten Musikstil der betreffenden Person handelt. Musik macht also nicht generell schlau. Sie aktiviert aber durchaus das Gehirn auf vielfältige Weise. Das gelte insbesondere für die Werke des Salzburger Wunderkinds Wolfgang Amadeus Mozart, vermeldeten die Psychologin Frances Rauscher von der University of Wisconsin Oshkosh und der mittlerweile emeritierte Physiker und Neurobiologe Gordon Shaw von der University of California im Jahr 1993. Sie ließen Studenten einen Intelligenztest absolvieren, bei dem es galt, Aufgaben zum räumlichen Denken zu knacken. Ein Teil der Probanden bekam davor eine Mozart-Sonate zu hören, ein Teil Entspannungsmusik und die dritte Gruppe verbrachte die zehnminütige Vorbereitungszeit in absoluter Stille. Die Mozart-Hörer erreichten durchschnittlich 8 bis 9 IQ-Punkte mehr im Intelligenztest. "Mozart-Musik kann das Gehirn aufwärmen", spekulierte Shaw damals. "Wir vermuten, dass differenzierte Musik komplexe Denkvorgänge erleichtert." Monotone Musik könne dagegen das Umgekehrte bewirken. "Heute wissen wir, dass die Leistungssteigerung der Probanden wenig mit der speziellen Wirkung von Mozarts Kompositionen zu tun hatte", sagt Eckart Altenmüller, Professor für Musikphysiologie und Musikermedizin an der Hochschule für Musik, Theater und Medien Hannover. "Es handelt sich vielmehr um einen Präferenzeffekt, der das Wohlbefinden steigert und die Aufmerksamkeit erhöht." 2006 fanden die Psychologen George Caldwell und Leigh Riby von der Glasgow Caledonian University heraus, dass sich mit Rockmusik ebenso wie mit Klassik die Konzentrationsfähigkeit steigern lässt – vorausgesetzt es handelt sich um den bevorzugten Musikstil der Probanden.
- Musizieren ändert das Hirn dauerhaft
  Alle Neuverknüpfungen, die zwischen den Nervenzellen im Hirn entstehen, bleiben erhalten. Man kann sagen, dass Musik einen Trainingseffekt fürs Gedächtnis hat. Dies hat man auch bei hochbetagten Profi-Musikern festgestellt. Hirnareale, die normalerweise im Alter abbauen, sind bei den ehemaligen Profis noch immer stark ausgeprägt.
- Musiker haben mehr graue Substanz
  Mithilfe der Schnittbilder des menschlichen Gehirns zeigte sich, dass in Musikerhirnen die Verbindung zwischen rechter und linker Gehirnhälfte, das sogenannte Corpus callosum, deutlich kräftiger ausgebildet ist. Und es ist mehr graue Substanz in Regionen vorhanden, die für die Motorik, die auditive und die räumlich-visuelle Wahrnehmung zuständig sind. Forscher der Universität Jena haben in Zusammenarbeit mit Gottfried Schlaug von der Harvard Medical School in Boston herausgefunden, dass sich die Gehirne von Berufsmusikern auffällig von jenen der Nichtmusiker unterscheiden. Bereiche, die für das Hören, das räumliche Sehen und das Umsetzen von Bewegung zuständig sind, waren bei Musikern deutlich vergrößert. Wahrscheinlich, weil Musiker in ihrem Spiel nicht nur vorausdenken und die passenden Bewegungen zur Musik ausführen müssen, sondern gleichzeitig auch überprüfen sollen, ob sie richtig gespielt haben.
- "Es ist eigenartig, aber aus neurowissenschaftlicher Sicht spricht alles dafür, dass die nutzloseste Leistung, zu der Menschen befähigt sind – und das ist unzweifelhaft das unbekümmerte, absichtslose Singen – den größten Nutzeffekt für die Entwicklung von Kindergehirnen hat."
  Prof. Dr. Gerald Hüther, Leiter der Zentralstelle für Neurobiologische Präventionsforschung der Universität Göttingen und Mannheim/Heidelberg Dass Musik einen großen Nutzen hat, wird nicht nur von den Neurobiolog*innen belegt, auch andere Wissenschaftler*innen und Musikpädagog*innen beschreiben die Wichtigkeit von Musik im Alltag. Gleichzeitig mehren sich die Aussagen darüber, dass in Familien und anderswo seltener gesungen würde als früher, dass Erzieher*innen seltener Instrumente spielen könnten. In der Ausbildung zur Erzieher*in gebe es so wenige Unterrichtseinheiten zu Musik, dass diese nicht einmal ausreichten, um ein Liedrepertoire für wichtige Feste im Jahreslauf aufzubauen (Zimmer 2019). Darüber hinaus sei bedenkenswert, dass ein einmaliges Modul keinerlei Regelmäßigkeit biete. Regelmäßigkeit jedoch ist beim Singen und bei Musik allgemein überaus sinnvoll. Und schließlich stehen an manchen Orten Statements im Raum, über die sich nachzudenken lohnt: "Hauptsache, es wird gesungen. Wie gesungen wird, das spielt doch keine Rolle.", "Ich kann nicht singen, deshalb spiele ich den Kindern die richtigen Melodien von einer CD vor". Diese beiden Statements würde ich in einem ersten Schritt gern näher beleuchten: "Hauptsache, es wird gesungen. Wie gesungen wird, das spielt doch keine Rolle." Die Sängerin Catherine Veillerobe schließt sich dieser These grundsätzlich an, auch wenn sie diese in Bezug auf die Stimmlage differenziert. Über Pädagog*innen, die Singimpulse an die Kinder herantrügen, könne man sich grundsätzlich freuen. Vornehmlich hebt sie im Interview mit Jasmin Zimmer hervor, dass mit der ästhetisch-musischen Bildung viele Kompetenzen gestärkt würden – soziale, kommunikative, das Empathie-Empfinden. Als Beispiel bezieht sie sich auf Erkenntnisse aus der Sprachentwicklungsforschung, nach denen rhythmisch dargebrachte Impulse Kinder stärker stimulierten als rein sprachliche und so die Sprachentwicklung mitunter besser anregten. Verbunden mit Bewegung und Tanz werde "das Singen zu einer ganzheitlichen Betätigung", es entstehe ein Flow, der "das Wohlbefinden des Menschen erheblich" steigere (Veillerobe, in: Zimmer 2019). Etwas komplexer wird es bei der Betrachtung der Stimmhöhe. Ideal – so die Musikerpädagogin – sei es, wenn sich die Stimmen anglichen: die Stimmhöhe der Erzieherin solle sich möglichst an die der Kinder (hohe Stimmlage) angleichen. "Gesunde, ausgeglichene Kinderstimmen bewegen sich hauptsächlich im Kopfstimmbereich. Rein physiologisch, betrachtet man die Resonanzräume aber auch die Körperproportionen – der Kopf ist im Verhältnis zum Körper viel größer als bei Erwachsenen – sind Kinder beim Singen in diesem Kopfstimmbereich natürlicherweise zu Hause" (Veillerobe, in: Zimmer 2019). Bei Erwachsenen kommen je nach Stimmfach mehr Bruststimmanteile dazu. Da Kinder von Vorbildern lernen, ist es wichtig, sie beim Finden ihrer Kopfstimmigkeit zu unterstützen und eine kindgerechte Stimmlage anzubieten. So kann die Stimmentwicklung von Kindern gut unterstützt werden. Nun scheint es manchem Erwachsenen etwas peinlich zu sein, mit den Kindern in hohen Stimmlagen zu singen oder höher, als man vielleicht sonst singen würde. Aber – so die Überzeugung der Sängerin – es kommt auf die eigene Haltung und das angestrebte Ziel an. Nach Erfahrung der Spezialistin singen Kinder häufiger und lieber mit, wenn das Vorbild in hoher Stimmlage singt. Und vielleicht animiert man so auch Kinder zum Mitmachen, die eher selten einen Ton von sich geben. So gesehen könnte man auch sagen: der Erfolg wird den aktiven Sänger*innen Recht geben. Wer ist nicht schon einmal bei einem Konzert in der Kita, das beim Sommerfest geboten wird, gerührt dagestanden und hat sich einfach gefreut, mit welchem Stolz und welcher Inbrunst die Kinder ihr Können präsentieren. Wen packte es nicht, wenn die Kinder beim täglichen Singkreis engagiert mitklatschen, hüpfen, ein Lied oder Passagen davon mitsingen. "Ich kann nicht singen, deshalb spiele ich den Kindern die richtigen Melodien von einer CD vor." Damit komme ich zum zweiten Statement, das nicht selten zu hören ist. Die einfachste Antwort darauf könnte lauten: Kann ich nicht, gibt´s nicht. Mit

den dazu nötigen Organen kann jeder Mensch singen; man muss es ausprobieren und in gewisser Weise üben. Auch andere Fertigkeiten erlernt man durchs Tun und durch vielfache Wiederholung, und so verhält es sich auch beim Singen. Mit etwas Mut und gemeinsam mit einer geübten Kolleg*in wird sich die Freude am Singen einstellen. Auf Perfektion kommt es hier gewiss nicht an, "vielmehr auf die Lust am Tönen, am Ausprobieren." Dabei können wir Erwachsenen erleben, "wie unsere eigenes Instrument der Stimme wieder zum Klingen kommt – und welche Wirkungen Singen nicht nur auf die Kinder, sondern auch auf uns selbst hat" (Kreusch-Jacob 2018, 35). Beim Abspielen einer CD fehlt darüber hinaus "der menschliche Aspekt, auch die emotionale Anbindung an das Gegenüber" (Veillerobe, in: Zimmer 2019). Und diese Anbindung spielt, wie bei vielen Lernprozessen, eine bedeutende Rolle. Das singende Vorbild regt das Kind ungleich stärker an mitzusingen, als es eine CD vermag. Das Auflegen der CD kann einfach sein, aber man könnte entgegen halten: Was wir selbst zu produzieren imstande sind, das sollten wir auch zum Anregen gelungener Bildungsprozesse tun. Verstehen Sie mich bitte im besten Sinne, das Abspielen einer CD soll nicht verteufelt werden: Wenn ansteht, ein bestimmtes Lied – vielleicht aus "Peter und der Wolf" – vorzuspielen, so steht dem nichts im Wege. In dem hier dargestellten Zusammenhang geht es um das Vorbild, das wir als Pädagog*innen sein können, und um ein Lebensgefühl, das sich mit dem Gesang transportiert. Die Opernsängerin und Musikpädagogin Catherine Veillerobe drückt es so aus. "Ein singender Mensch vermittelt nicht nur den reinen Klang, sondern transportiert auch immer tiefgreifende Gefühle und ganze Lebenswelten, die ihn bestimmen." Und daraus folgt für sie ganz einfach: "Singen macht glücklich!" (Veillerobe, in: Zimmer 2019).

- Gehirn verarbeitet jede Musik anders Unser Gehirn verrät, welche Musik wir hören. Welche Musik wir hören, verrät das Muster unserer Hirnaktivität. Ob und wie sich das Muster der Aktivität auch zwischen verschiedenen Musikgenres unterscheidet, hat im August 2013 ein Forscherteam um den Studienleiter Vinoo Alluri von der Universität von Iyväskylä in Finnland untersucht. Für ihre Studie spielten sie Probanden mehrere unterschiedliche Musikstücke vor, darunter Ausschnitte aus einem Vivaldi-Konzert, ein Jazzstück von Miles Davis, Blues, einen argentinischen Tango und ein Stück von den Beatles. Während die Teilnehmer der Musik lauschten, zeichneten die Forscher ihre Hirnaktivität mittels der fMRT auf. Wie erwartet, gab es einige Areale, die von allen Musikarten aktiviert wurden: Bereiche in der Hörrinde, im Emotionen verarbeitenden limbischen System und im motorischen Kortex. Aber es gab auch Unterschiede: Besonders komplexe Musikstücke lösten eine höhere Aktivität im rechten Schläfenlappen aus. Und noch etwas wurde deutlich: Bei Liedern mit Text, beispielsweise Popsongs, verschob sich die Aktivität von der linken überwiegend in die rechte Hirnhälfte.
- Musizieren/hören schüttet Endorphine aus
  Musizieren und Musikhören lösen im Gehirn dieselben Effekte aus wie Essen, Sport, Sex oder Drogen. Es kommt zu einer Ausschüttung von Endorphinen, unseren körpereigenen Glückshormonen, und zu einer Verringerung des Stresshormons Cortisol. Zudem wird vermehrt der Neurotransmitter Dopamin ausgeschüttet, der eine wichtige Rolle im Belohnungssystems unseres Gehirns spielt und motivierend wirkt. Musik, die unserer Stimmung entspricht, hebt unser Wohlbefinden. Mit anderen Personen Musik zu machen oder ein Konzert zu geniessen stimuliert auch die Ausschüttung von Oxytocin. Oxytocin ist als Bindungshormon bekannt, da es Vertrauen und Sympathie zwischen Personen fördert. All diese Veränderungen unserer Hirnchemie blockieren Schmerz, bauen Stress ab und lösen positive Emotionen aus – Musik macht glücklich. Das gilt übrigens nicht nur für fröhliche Musik: Fühlen wir uns traurig oder wütend, verbessert das Hören von Musik mit derselben Stimmung ebenfalls unser Wohlbefinden. Zu sehen, was beim Musikhören passiert wurde erst durch bildgebende Verfahren wie die funktionelle Magnetresonanztomografie (fMRT) möglich. Hört ein Mensch Musik, werden die Strukturen zuerst im Hirnstamm verarbeitet. Auf dieser Ebene ist die Musik noch nicht ins Bewusstsein gedrungen. Das geschieht erst, wenn die Reize das Hörzentrum, den sogenannten Hörkortex, erreichen. Erst dort werden Instrumente oder Stimmen unterschieden. Es ist schon länger bekannt, dass Musiker andere Gehirnstrukturen haben als Nichtmusiker. Neu ist lediglich, dass bei Jazzpianisten andere Hirnprozesse ablaufen als bei klassischen Pianisten, selbst wenn sie das gleiche Musikstück spielen. "Miles Davis ist nicht Mozart", betonen die Leipziger Forscher des Max-Planck-Instituts (MPI) für Kognitions- und Neurowissenschaften in ihrer Studie vom Januar 2018. Der Grund: Die beiden Musikstile fordern den Musikern unterschiedliche Fähigkeiten ab.
- Musiker >< Nichtmusiker
  Auch, wenn Musik an sich nicht schlau macht, gibt es einige Wirkungen von Musik, die Einfluss auf unser Hirn haben. Einen Effekt, den jeder kennt, ist, dass durch sie Emotionen ausgelöst werden können. So bringt uns zum Beispiel das erste Weihnachtslied im Dezember in Weihnachtsstimmung oder die Filmmusik eines Horrorfilms versetzt uns in Angst und baut Spannung auf. Zwar könnte man vermuten, dass es ein direktes Musikzentrum gibt, aber ein solches gibt es im Hirn nicht. Viel mehr wirkt sich Musik auf unterschiedliche Hirnareale aus. So zum Beispiel auf den auditorischen Kortex, auf eines von zwei Spracharealen, auf motorische und visuelle Areale, auf das limbische System, das für Emotionen zuständig ist und auf das Belohnungssystem. Beim Musik Hören und insbesondere auch beim Musizieren hat unser Hirn jede Menge zu tun. Es muss eine große Fülle von Informationen verarbeiten, wie die Tonhöhen, Melodien und Rhythmen aber auch gleichzeitige Töne als Akkord wahrzunehmen oder zu erkennen welches Instrument im Raum wo spielt, erfordert einiges an kognitiver Leistung. Diese große Anzahl an Vergleichen und Messungen im Hirn teilen sich die linke und rechte Hirnhälfte. Doch wo liegen jetzt die Unterschiede zwischen Musikern und Nichtmusikern? Es wurde nachgewiesen, dass der Corpus Collasum, also der Hirnbalken, der für den Austausch von Informationen zwischen den beiden Hirnhälften zuständig ist, bei Profi-Musikern stärker ausgebildet ist. Des Weiteren wurde auch entdeckt, dass Musiker mehr graue Substanz, also Nervenzellkörper, in den Bereichen des Gehirns haben, die für das räumliche Sehen, Hören und die Motorik zuständig sind. Also die Areale, die die Aktivitäten der Hände, mit denen des Hörens und Analysierens verknüpfen, sind besonders ausgeprägt.
  Es bleibt zu sagen, dass keine allgemeingültige Aussage zur richtigen Musik oder zum richtigen Umgang mit Musik getroffen werden kann, denn jeder nimmt Musik etwas anders wahr. Zur besseren Konzentration hingegen eignet sich häufig jedoch eher instrumentelle Musik, da uns die sprachlichen Reize häufig ablenken. Zum Sport eignet sich schnelle und treibende Musik. Grundsätzlich sollte aber jeder die Musik hören, die ihm am besten gefällt. Denn damit kann das generelle Wohlbefinden und die Motivation gesteigert werden. Funfact: Man geht davon aus, dass die Aufteilung der Aufgaben, die beim Laien linke und rechte Hirnhälfte übernehmen, beim Profi-Musiker genau umgedreht sind.
- Musik lässt manche Menschen kalt
  Während Musikliebhaber in den höchsten Tönen von ihren schönsten Konzerten schwärmen, lässt das andere völlig kalt. Neurowissenschaftler um Josep Marco-Pallarés von der Universität Barcelona haben im März 2014 herausgefunden, dass einige Menschen völlig immun gegen jede Wirkung von Musik sind. Die Forscher sprechen von Anhedonie - der Unfähigkeit, Freude zu empfinden. In Tests erkannten die Teilnehmer zwar, ob Musik fröhlich oder traurig war, aber sie ließen sich von den Gefühlen nicht anstecken. Die Forscher gehen davon aus, dass ihr Belohnungssystem im Gehirn anders arbeitet. Denn die Studienteilnehmer waren durchaus zur Freude fähig, beispielsweise, wenn sie in einem Spiel Geld gewinnen konnten. Nur Musik hatte bei ihnen keine Auswirkung.
- Hirn-Wirkungen: Die äußerste Schicht (Hirnrinde) ist in erster Linie für bewusste Empfindungen und Handlungen zuständig.
  Musizieren erfordert höchste Konzentration: Hören, Sehen und Bewegung finden gleichzeitig statt. Musik ist mehr als akustisches Signal. Sie aktiviert weite Bereiche des Gehirns, weckt Assoziationen und Emotionen und wurzelt möglicherweise in einer Art vorsprachlichen Kommunikation (*Prof. Dr. Eckart Altenmüller*). Während wir Musik unbeschwert genießen, arbeitet unser Gehirn auf Hochtouren. Auch die Finger eines Pianisten, die scheinbar einfach über

die Tasten gleiten, werden durch komplizierte Prozesse im Kopf gesteuert. Dank immer ausgefeilterer Verfahren konnten Wissenschaftler weltweit bereits viele Geheimnisse in diesem Zusammenhang lüften. Doch was geschieht überhaupt, wenn Musik an unser Ohr dringt? Zunächst einmal besteht sie aus Klängen oder, physikalisch ausgedrückt, aus Schwingungen, die durch feine Veränderungen des Luftdrucks entstehen. Im Ohr wird der mechanische Reiz in ein neuronales Signal verwandelt, das über mehrere Umschaltstationen letztlich die Hörrinde im Schläfenlappen erreicht Hören: Vom einfachen Wackeln zur wunderbaren Vielfalt der Klänge. Bereits auf dem Weg dorthin wird das akustische Signal mehrfach analysiert und vorsortiert. So differenziert der Cochleariskern zwischen einzelnen, gleichbleibenden Tönen und einem akustischen Muster. Und am Ende der Hörbahn, in der primären Hörrinde, unterscheidet die Heschel'sche Querwindung zwischen reinen Tönen und komplexen Hörreizen wie Mehrklängen und Klangfarben. Im weiteren Verlauf fächert sich die Musikverarbeitung im Gehirn immer weiter auf. Kein Wunder also, dass die Suche nach einem Musikzentrum im Denkorgan zum Scheitern verurteilt war. Neben den bereits erwähnten Arealen im motorischen Cortex, kommen etwa die visuellen Zentren ins Spiel. "Wenn wir beispielsweise einem Streichquartett lauschen, sehen wir vor unserem inneren Auge die Geiger und Cellisten musizieren", erklärt Altenmüller. "Und wir verknüpfen mit dem Höreindruck eine kulturelle und historische Prägung, die mit dieser Art von Musik im Zusammenhang steht." Zudem tritt das limbische System in Aktion. Es bewertet etwa, ob uns Musik gefällt oder nicht. So agiert der Gyrus cinguli, wenn eine Melodie als angenehm empfunden wird. Dissonante, als unangenehm erfahrene Klänge regen dagegen den Gyrus parahippocampalis an. Auch das Belohnungssystem trägt seinen Teil zum Musikempfinden bei. Es wird – ähnlich wie beim Sex, Essen oder Drogenkonsum – aktiv und zeichnet für den einen oder anderen wohligen Schauer verantwortlich. Und das alles geschieht vor dem Hintergrund persönlicher Vorlieben und Erfahrungen sowie der kulturellen Prägung. Streng genommen ist also Musik in jedem individuellen Gehirn ein wenig anders repräsentiert. Überholt ist auch die Vorstellung, dass Musikverarbeitung eine Sache der rechten Hirnhälfte sei. Tatsächlich sind beide Hemisphären beteiligt, wenn auch mit unterschiedlichen Aufgaben. Bereits 1990 folgerte Isabell Peretz von der Université de Montréal in Kanada aus der Untersuchung von Patienten mit einseitigen Hirnschäden, dass die rechte Hirnhälfte Musik eher ganzheitlich verarbeitet, die linke dagegen analytisch. Doch auch diese Hypothese scheint sich nicht zu bestätigen. Eckart Altenmüller und seine Kollegin Maria Schuppert von der Hochschule für Musik Detmold untersuchten im Jahr 2000 Patienten mit rechts- oder linksseitigen Schlaganfällen. Das Ausfallmuster war recht heterogen. Betroffene mit Schäden in der linken Hälfte hatten entweder Schwierigkeiten mit Rhythmen oder Tonfolgen. War die rechte Seite in Mitleidenschaft gezogen, haperte es entweder an Kontur und Tonfolge oder an Takt und Rhythmus. Demnach, so vermuteten die Forscher, existiert bei der Musikverarbeitung eine Art Hierarchie: Während die rechte Hirnhälfte die Grobstruktur herausarbeitet, übernimmt die linke Hemisphäre die Feinanalyse. Es gibt nicht das eine Musikzentrum im Hirn. Musik aktiviert die unterschiedlichsten Hirnregionen gleichzeitig. Denn Musik zu machen beansprucht ein kompliziertes Zusammenspiel sehr verschiedener Fähigkeiten: den Hörsinn, den Sehsinn, den Tastsinn, die Feinmotorik. Neuere Untersuchungen haben gezeigt, dass bei der Verarbeitung von Musik sogar das Broca-Areal beteiligt ist, eines der beiden Sprachzentren, mit Auswirkungen auf kognitive und emotionale Entwicklung. (Es gibt das Broca- und das Wernicke-Sprachzentrum: Ohne Broca kannst nix reden; ohne Wernicke verstehst die Sprache nimmer und kannst nur Schwachsinn lallen)

Musik hören tut Körper und Geist gut. Sie lässt uns im Takt wippen, mit den Füßen tippeln, sie motiviert uns beim Sport und beeinflusst unsere Stimmung positiv an schlechten Tagen. Spätestens seit die Neurowissenschaften genauer auf die Auswirkungen von Musik in den verschiedenen Phasen des Lebens schauen und vielfältige positive Bezüge finden, ist klar: Musik gehört von Anfang an dazu. Bereits im Mutterleib hat das Ungeborene Hörerfahrungen Bereits ab der 22. Schwangerschaftswoche ist die Hörschnecke bei Ungeborenen vollständig ausgebildet. Es verwundert also nicht, dass schon Ungeborene Musikstücke wiedererkennen und darauf reagieren, wie zum Beispiel durch Bewegungen oder Tritte im Mutterleib und nach der Geburt mit erhöhter Aufmerksamkeit. Die Informationen aus dem Mutterleib sind wichtig und haben einen praktischen Nutzen nach der Geburt (vgl. Pauli 2009), nämlich Wiedererkennung. Außerdem stellten Forscher fest, dass sich der Herzschlag von Ungeborenen bei sanften Klängen, wie zum Beispiel bei klassischer Musik, beruhigte und Bewegungen entspannter wurden. Das gegenteilige Verhalten konnte bei anregender Musik festgestellt werden. Kinder können also bereits im Mutterleib durch Musik geprägt werden, wobei werdende Mütter nicht unbedingt nach einer Prägung die zuhörende Musikrichtung aussuchen sollten, sondern viel lieber nach eigenen Präferenzen. Denn das tut Kind und Mutter gut.

Herzschlag, Atemfrequenz und Muskelspannung je nach Musikrichtung. Die Klänge wirken sich nach Forschungsergebnissen auf unsere Nebenniere und die Hypophyse aus. Musizieren und Musikhören hat folglich Einfluss auf den Hormonhaushalt. Deshalb hört man beim Sport gerne schnelle und aggressive Musik. Das dabei ausgeschüttete Adrenalin treibt zu Höchstleistungen an. Das Gegenteil passiert bei ruhiger Musik. Das Stresshormon Cortisol wird verringert, Noradrenalin wirkt Stress reduzierend und sogar Schmerz dämpfend. Deshalb wird Musik auch in der Schmerztherapie unterstützend eingesetzt. Musik und Musizieren haben den Weg in die Medizin gefunden. Durch Musizieren werden neue Nervenverbindungen geschaffen. Dies kann positiv in der Rehabilitation von Schlaganfallpatienten und in der Altersheilkunde eingesetzt werden. Musik hat nicht nur chemische Effekte auf unser Gehirn, sondern auch strukturelle. Musikalische Reize sorgen dafür, dass sich die Nervenzellen in unserem Gehirn neu verschalten und sich die Hirnareale so besser miteinander vernetzen. Die Fähigkeit des Gehirns, sich auf diese Weise zu verändern, wird als Neuroplastizität bezeichnet. Gehirne von Berufsmusikern zeigen im Vergleich zu Nichtmusikern einige Unterschiede: So ist die Verbindung zwischen den beiden Hirnhälften, das Corpus Callosum, deutlich stärker ausgebildet. Dies legt nahe, dass die beiden Hirnhälften besser miteinander kommunizieren können. Auch haben Musikerhirne mehr graue Substanz in Bereichen, die für das Hören, räumliches Sehen und die Motorik zuständig sind.

Es gibt immer mehr Untersuchungen, die zeigen, dass Musik nicht nur auf unser emotionales, sondern auch auf unser körperliches Wohlbefinden einen positiven Effekt hat. So fanden Wissenschaftler beim Hören von Musik eine Steigerung von Antikörpern und Zellen, die uns gegen Bakterien schützen. Der Einsatz von Musik erzielte ausserdem positive Effekte in der Therapie verschiedenster Erkrankungen, von Depressionen und Schlaflosigkeit über ADHS und Schizophrenie bis hin zu Parkinson, Demenz und Schlaganfällen. Es gibt also kaum einen einfacheren, billigeren und vor allem genüsslicheren Weg, unserem Körper und Geist etwas Gutes zu tun, als immer wieder einmal die Kopfhörer aufzusetzen. Deine Workout-Playlist motiviert dich nicht nur und bespaßt dein Hirn, während du anstrengende Arbeit verrichtest, sondern erhöht auch deine Schmerztoleranzgrenze und ist damit die gängigste Form, Schmerzen besser auszuhalten oder zu mindern, funktioniert nur etwas anders als die Ibuprofen, die wir kennen. Musik ist nämlich nichts, was wir aktiv zu uns nehmen, um unseren Körper zu heilen (auch wenn viele jetzt das Gegenteil behaupten werden). Musik minimiert Schmerzen, indem es unsere Konzentration verlagert. Wenn du gestresst bist, sorgt dein Gehirn dafür, dass genug „Feel Good"-Hormone deine Negativität verlagern. Du fühlst dich stärker und je stärker du dich fühlst, desto mehr Schmerzen kannst du in dem Moment tolerieren. Die Forschung hat gezeigt, dass das Hören von Musik Angstzustände, Blutdruck und Schmerzen verringern sowie die Schlafqualität, die Stimmung, die geistige Wachheit und das Gedächtnis verbessern kann. Möchtest du deinen Körper straffen, gehst du ins Fitnessstudio. Möchtest du aber dein Gehirn trainieren, solltest du Musik hören/machen. Musik bewegt. Diesen Reiz, diese Emotion macht sich auch die Medizin mehr und mehr zu Nutze, zum Beispiel um die gesamte Körperspannung hinabzusetzen. Die passende Musik kann die Ausschüttung des Stresshormons Cortisol dämpfen und

gleichzeitig die Aufmerksamkeit nach Außen hinwenden - weg vom Schmerz, hin zur Musik. Der Schmerzeindruck entsteht im limbischen System, dort, wo die Emotionen sitzen. Musik belegt die gleiche Stelle und kann somit als eine Art Filter gegen Schmerzen fungieren. Mit der Lieblingsmusik können die Schmerzen buchstäblich überspielt werden. Eckart Altenmüller. Beim Musikmachen, aber auch beim Hören, vollbringt das Gehirn Höchstleistungen. Hirnregionen verbinden sich und das kommt besonders Schlaganfallpatienten zu Gute. Vor allem im Bereich Gedächtnistraining, Aufmerksamkeit und Motorik führt die Therapie mit Musik oft zu schnelleren Erfolgen. Nicht zu vergessen: die Stimmung der Betroffenen. Musizieren belebt und schafft bestenfalls positive Emotionen. Das klappt natürlich nur mit individualisierter Lieblingsmusik. Einfach zu sagen: 'Mozart macht gesund' funktioniert nicht (Eckart Altenmüller). Was kennt der Patient, welche Musik mag der Patient wirklich? Erst die persönliche Musikauswahl schafft gute Stimmung sowie Motivation und damit einen verbesserten Lerneffekt.

Musik hat die ungewöhnliche Kraft blockierte Erinnerungen an die Oberfläche zu bringen. Ein bestimmter Song kann dich an ein tolles Erlebnis vor über 20 Jahren erinnern, aber auch Schmerzen hervorrufen, die du einmal gefühlt hast. Musik kann traurige Erinnerungen aber auch tolle wecken. Musik wird daher häufig auch zur Bekämpfung von Alzheimer eingesetzt, um verlorengegangene Erinnerungen wiederzuerwecken. Musik ist kein Heilmittel gegen diese neurologische Störung, aber eine große Erleichterung für die Betroffenen.

Verspürst du manchmal eine Gänsehaut, während du ein (klassisches) Musikstück hörst, oder während jemand eine hohe Note ansingt? Wenn du diese Frage bejahen kannst, fühlst du Musik anders, als die meisten Menschen. Dein Gehirn schüttet mehrere starke Chemikalien aus, die deinen Herzschlag erhöhen, deine Temperatur steigern und deine Haut leitfähiger für Elektrizität machen, so dass dein Nervensystem, wenn es ein elektrisches Signal aussendet, winzige Muskeln am Ansatz deines Haares auf deiner Haut stimuliert. Du bekommst eine Gänsehaut. Eigentlich sollte eine Gänsehaut nicht Anlass zur Freude sein, denn Gänsehaut ist schließlich ein Anzeichen für Nervosität oder Kälte, doch die musikalische Gänsehaut wird durch deine positiven Emotionen zum Musikstück geformt. Eine Studie im Jahr 2016 fand heraus, dass Menschen, die solch eine Gänsehaut empfinden, wohl ein besseres Gehör haben und mit einem tieferen musikalischen Verständnis gesegnet sind. Kannst du dich dazu zählen?

- Musik als Suchtmittel
  Hast du dich schon einmal gefragt, warum wir uns in bestimmte Songs verlieben und ein und dasselbe Lied am liebsten den ganzen Tag für immer und ewig hören möchten? Musik gibt uns Energie und Kraft, Inspiration und Freude. Studien zufolge hat Musik denselben Einfluss auf dein Belohnungssystem, wie abhängigmachende Drogen. Verantwortlich dafür ist das Dopamin, welches während des Musikhörens ausgeschüttet wird – du willst immer mehr davon! Aber keine Sorge, die Musikabhängigkeit ist wohl eine der gesündesten,
- Berufsmusiker haben andere Hirne als Laien
  Forscher der Universität Jena haben in Zusammenarbeit mit Gottfried Schlaug von der Harvard Medical School in Boston herausgefunden, dass sich die Gehirne von professionellen Musikern auffällig von jenen der Nichtmusiker unterscheiden. Gehirnareale, die für das Hören, das räumliche Sehen und das Umsetzen von Bewegung zuständig sind, waren bei Musikern deutlich vergrößert. Wahrscheinlich, weil Musiker in ihrem Spiel nicht nur vorausdenken und die passenden Bewegungen zur Musik ausführen müssen, sondern gleichzeitig auch überprüfen sollen, ob sie richtig gespielt haben.
- Das Gehirn verrät, ob und welche Musik wir hören
  Im Gehirn zu sehen, was beim Musikhören passiert, wurde erst möglich durch bildgebende Verfahren wie die funktionelle Magnetresonanztomografie (fMRT). Dabei lässt sich erkennen, welche Art von Musik wir hören, ob wir Berufsmusiker sind oder ob uns Musik gar nicht anspricht. Bei Jazzpianisten laufen andere Hirnprozesse ab als bei klassischen Pianisten. Das haben Leipziger Forscher des Max-Planck-Instituts für Kognitions- und Neurowissenschaften in ihrer Studie vom Januar 2018 festgestellt. Der Grund: Die beiden Musikstile fordern den Musikern unterschiedliche Fähigkeiten ab. Welche Musik wir hören, verrät das Muster unserer Hirnaktivität. Die Aufgabe von Klassikpianisten ist es, ein Stück einfühlsam zu interpretieren. Sie konzentrieren sich beim Spielen besonders darauf, wie sie ein Stück spielen. "Dadurch scheinen sich unterschiedliche Abläufe im Gehirn etabliert zu haben, die während des Klavierspielens ablaufen und den Wechsel in einen anderen Musikstil erschweren", so Daniela Sammler, Neurowissenschaftlerin und Leiterin der Studie. Bei Jazzpianisten geht es vor allem darum, eine Melodie fantasievoll zu interpretieren und zu variieren. Deshalb reagieren Jazzpianisten normalerweise schneller auf eine unvorhergesehene musikalische Situation und können ihr Klavierspiel trotzdem leicht fortführen. "Als wir sie während einer logischen Abfolge von Akkorden plötzlich einen harmonisch unerwarteten Akkord spielen ließen, begann ihr Gehirn viel früher, die Handlung umzuplanen als das Gehirn klassischer Pianisten", erklärt Roberta Bianco, Erstautorin der Studie. Wenn es aber darum geht, ungewöhnliche Fingersätze zu nutzen, hatten in der Leipziger Studie die klassischen Pianisten die Nase vorn. Wie sich das Muster der Aktivität auch zwischen verschiedenen Musikgenres unterscheidet, hat vor einigen Jahren ein Forscherteam um den Studienleiter Vinoo Alluri von der Universität von Iyväskylä in Finnland untersucht. Für ihre Studie spielten sie Probanden unterschiedliche Musikstücke vor, darunter Ausschnitte aus einem Vivaldi-Konzert, ein Jazzstück von Miles Davis, Blues, einen argentinischen Tango und ein Stück von den Beatles. Während die Teilnehmer der Musik zuhörten, zeichneten die Forscher die Aktivität ihres Gehirns mit Hilfe der Magnetresonanztomografie auf. Wie vermutet, gab es einige Gehirnareale, die von allen Musikgenres aktiviert wurden, aber es gab auch Unterschiede: Besonders komplexe Musikstücke lösten eine höhere Aktivität im rechten Schläfenlappen aus. Und bei Liedern mit Text, beispielsweise Popsongs, verschob sich die Aktivität von der linken überwiegend in die rechte Hirnhälfte. Musik macht also nicht generell schlau – auch nicht klassische. Aber sie regt das Gehirn in vielfältiger Weise an. Wie das genau aussieht, hängt von der persönlichen Biografie ab, ist also individuell verschieden. "Wenn jemand zum Beispiel Klavierunterricht hatte und dann einem Klavierkonzert von Beethoven lauscht, werden die Areale im motorischen Cortex aktiv, die die Hand und die Fingerbewegung repräsentieren", erklärt Altenmüller, der selbst Flöte und Medizin studiert hat. Das Denkorgan spielt im Geiste mit – auch wenn die betreffende Person die Finger gar nicht bewegt. Bei Profis sind die Aktivierungsmuster beim Spielen und Zuhören sogar annähernd identisch. Ähnliches passiert auch, wenn der Musikhörer selbst gar kein Instrument spielt, wie Experimente des Musikpsychologen Stefan Koelsch von der Freien Universität Berlin aus dem Jahr 2006 nahelegen. Bei Probanden, die als angenehm empfundene Musik im funktionellen Magnetresonanztomografen hörten, waren Bereiche des Rolandischen Operculums aktiv – eine Region, die unter anderem Kehlkopf und Stimmbänder repräsentiert. Möglicherweise hatte das Gehirn im Geiste "mitgesummt".

  Es ist schon länger bekannt, dass Musiker andere Gehirnstrukturen haben als Nichtmusiker. Zudem gibt es Auffälligkeiten im Hirn von Klassik- oder Jazzpianisten.
- Eine Form der Kommunikation
  Es geht aber um mehr als reines Hörvergnügen. "Die Musik drückt das aus, was nicht gesagt werden kann und worüber zu schweigen unmöglich ist", sagte einst der französische Schriftsteller Victor Hugo. Dass Musik eine so emotionale Angelegenheit ist, könnte ganz in den Ursprüngen der menschlichen Evolution begründet sein, möglicherweise als eine Art vorsprachliche Kommunikation. So vermutet der Emotionsforscher Jaak Panksepp, Emeritus an der Bowling Green State University in Ohio, dass frühe Hominiden mit Hilfe melodischer Rufe in Kontakt zueinander blieben – etwa wenn eine Mutter außer Sichtweite ihres Sprösslings nach Nahrung suchte. Das Gehirn scheint Musik zudem ganz ähnlich zu verarbeiten wie Sprache – nämlich nach syntaktischen Regeln. Es analysiert Töne, Intervalle und Akkorde und stellt sie in einen Zusammenhang. Bereits 2002 erkannte die Neuropsychologin Angela Friederici vom Max-Planck-Institut für Kognitions– und Neurowissenschaften in Leipzig, dass dabei unter anderem das Broca-Areal und der vordere Teil des Gyrus temporalis su-

perior aktiv sind Hören: Vom Laut zum Wort. Beide sind auch für die syntaktische Verarbeitung von Sprache von Bedeutung. Und Stefan Koelsch stellte 2005 fest, dass musikalische Regelverstöße im Denkorgan zu Irritation führen. So machte sich in seinen Experimenten beispielsweise ein überraschender, als falsch empfundener Schlussakkord durch auffällige Muster im EEG bemerkbar. Die Theorie der nonverbalen Kommunikation wird dadurch untermauert, dass wir verschiedenen Klängen automatisch eine Bedeutung beimessen. So empfinden wir Töne etwa als hohl, rau, spitz oder hell. Ob Klang und Bedeutung für einen Hörer zusammenpassen, lässt sich anhand des elektrischen Hirnpotenzials verfolgen, genauer gesagt: Anhand der so genannten N400-Welle. Sie zeigt sich bei der Sprachverarbeitung Sekundenbruchteile nachdem ein Wort erklingt und ist umso größer, je weniger es in den bisherigen Zusammenhang passt. Folgt beispielsweise auf den Satz "Der Junge singt ein Lied" das Wort "Musik", so ist die Welle kleiner, als wenn als Nächstes das unpassende Wort "Stift" kommt, wie Koelsch in einer Übersichtsarbeit zur Musikverarbeitung erklärt. Ähnlich verhalte es sich, wenn man eine sphärenhaft klingende Mozart-Symphonie mit dem Wort "Engel" assoziieren würde, im Experiment dagegen "Flegel" erklingt. Dieser Effekt stellt sich bereits bei kurzen, isolierten Klängen ein, wie Koelsch gemeinsam mit Julia Grieser-Painter von der Oregon Health & Science University im Jahr 2011 erkannte.

"Das kann so wahrscheinlich keine andere Spezies", sagt Stefan Koelsch in einem Audiobeitrag auf `dasGehirn.info Hirnforschung und Musik mit Stefan Koelsch`. Musik und Musikalität sind demnach zutiefst menschlich.

- Musik und Emotionen
  Das größte Problem bei der Beantwortung der Frage, wie Musik Emotionen erzeugt, dürfte die Tatsache sein, dass sich Zuordnungen von musikalischen Elementen und Emotionen nie ganz eindeutig festlegen lassen. Die Lösung dieses Problems ist die Strebetendenz-Theorie. Sie sagt, dass Musik überhaupt keine Emotionen vermitteln kann, sondern nur Willensvorgänge, mit denen sich der Musikhörer identifiziert. Beim Vorgang der Identifikation werden die Willensvorgänge dann mit Emotionen gefärbt. Das gleiche passiert auch, wenn wir einen spannenden Film anschauen und uns mit den Willensvorgängen unserer Lieblingsfigur identifizieren. Auch hier erzeugt erst der Vorgang der Identifikation Emotionen. Weil dieser Umweg der Emotionen über Willensvorgänge nicht erkannt wurde, scheiterten auch alle musikpsychologischen und neurologischen Versuche, die Frage nach der Ursache der Emotionen in der Musik zu beantworten. Man könnte dabei an einen Menschen denken , der einen Fernsehapparat aufschraubt und darin mit einer Lupe nach den Emotionen sucht, die er zuvor beim Ansehen eines Films empfunden hatte. Doch wie kann Musik Willensvorgänge vermitteln? Diese Willensvorgänge haben etwas mit dem zu tun, was alte Musiktheoretiker mit "Vorhalt", "Leitton" oder "Strebetendenz" bezeichnet haben. Wenn wir diese musikalischen Erscheinungen gedanklich in ihr Gegenteil umkehren (der Ton strebt fort - ich will, dass der Ton bleibt), dann haben wir in etwa den Willensinhalt gefunden, mit dem sich der Musikhörer identifiziert. In der Praxis wird dann alles noch etwas komplizierter, so dass sich auch differenziertere Willensvorgänge musikalisch darstellen lassen.

  Zum Thema "Musik und Emotionen" möchte ich auf den fünfteiligen Artikel "Warum klingt Moll traurig? Die Strebetendenz-Theorie erklärt das Gefühl in der Musik" verweisen. Er wurde im Online-Magazin "musik heute" publiziert und kann unter folgendem Link kostenlos heruntergeladen werden: http://www.musik-heute.de/tags/strebetendenz-theorie/ Bernd Willimek
  Bernd Willimek 20.01.2017
  Da ich mehrfach gebeten wurde, das Prinzip der Strebetendenz-Theorie auf eine Weise darzustellen, so dass sie auch ein Laie mühelos nachvollziehen kann, füge ich dem obenstehenden Artikel eine solche Erklärung bei. Sie ist unter folgendem Link kostenlos abrufbar:
  www.willimekmusic.de/erklaerung-strebetendenz-theorie.pdf
  Zur Strebetendenz-Theorie gibt es jetzt einen Wikipedia-Artikel:
  www.de.wikipedia.org/wiki/Strebetendenz-Theorie

- Elektrokortikographie
  Während das MRT die Gehirnaktivität anhand des Blutflusses nur von außen scannt und damit eine für die Bedürfnisse der Forscher eher grobe Abbildung des Geschehens liefert, bietet die Elektrokortikographie (ECoG) einen tieferen Einblick in die elektrische Aktivität der Nervenzellen. Dazu müssen jedoch Elektroden im Schädel angebracht werden, was einen operativen Eingriff erfordert und daher beim Menschen nicht ohne weiteres möglich ist. Bei Epilepsie-Patienten jedoch wird das Verfahren eingesetzt, um festzustellen, wo die Anfälle ihren Ursprung haben. 15 Betroffene erklärten sich bereit, darüber hinaus auch an der Musikstudie teilzunehmen. So konnten die Forscher eine detaillierte Arbeitsteilung der Hörzellen nachweisen. Sie vermuten, dass die auf Gesang spezialisierten Neuronen auf die Interaktion von Tonhöhe und Wörtern reagieren, bevor sie diese Informationen zur Verarbeitung an das Gehirn weiterleitet.
- Wie kommt eine musikalische Idee aufs Klavier?
  Dieser Frage gingen Forschende an den Max-Planck-Instituten für empirische Ästhetik in Frankfurt am Main (MPI EA) und für Kognitions- und Neurowissenschaften in Leipzig (MPI CBS) nach. In zwei aktuellen Studien zeigen sie, in welcher Hirnregion aus einer musikalischen Idee beim Solospiel eine Fingerbewegung wird, und dass es in Duetten auf die gemeinsame "Wellenlänge" der Gehirne ankommt. Alle Schritte, die für das Solospiel nötig sind, müssen Musiker mit ihrem Partner synchronisieren, wenn sie mit ihm im Gleichklang spielen wollen. Wie das genau funktioniert, haben die WissenschaftlerInnen in einer weiteren Studie untersucht. Wenn Menschen ihre Handlungen aufeinander abstimmen, beispielsweise beim gemeinsamen Musizieren oder Singen, synchronisieren sich ihre Hirnwellen. erklärt Daniela Sammler, Leiterin der Forschungsteams. Ein Grund für diese Synchronisation ist, dass die Beteiligten zur gleichen Zeit ähnliche Dinge hören. Doch wie verhalten sich diese Wellen, wenn sich die Musiker miteinander abstimmen müssen, um harmonisch miteinander zu spielen? Um das herauszufinden, wurden 14 Pianistenpaare mittels Elektroenzephalographie (EEG) untersucht, während sie kurze Duette spielten. Alle Stücke enthielten in der Mitte eine musikalische Pause ohne Ton, danach sollten die Musiker in einem anderen Tempo weiterspielen. Jeweils ein Partner sollte schneller spielen, der andere langsamer. Diese Manipulation machte tatsächlich einen Unterschied für die Synchronizität der beiden Gehirne. Planten beide PianistInnen dasselbe Tempo, war sie hoch. Waren die Tempi verschieden, war sie niedrig. Diese Ergebnisse legen nahe, dass die Synchronisation der Hirnwellen zwischen MusikerInnen nicht nur ein Nebenprodukt ist, das durch gemeinsame Höreindrücke und die Musik selbst ausgelöst wird, sondern tatsächlich ein Mechanismus, durch den sie ihr Spiel miteinander koordinieren. Zusammengenommen stellen die Studien einen wichtigen Beleg für die komplexe Koordinationsleistung zwischen Hirn und Hand sowie zwischen EnsemblespielerInnen beim Musizieren dar.
- Für die Wissenschaft: Klavierspielen im MRT
  Beim Musizieren planen PianistInnen immer zwei Dinge parallel: Sie müssen koordinieren, was gespielt wird, also welcher Ton oder Akkord folgen soll, und wie dieser gespielt wird, das heißt, welche Finger genau den Anschlag ausführen. Ein Forscherteam des MPI EA und des MPI CBS hat mithilfe einer funktionellen Magnetresonanztomografie untersucht, wo genau diese Planungsschritte im Gehirn stattfinden. Dazu beobachten sie die Hirnaktivität die StudienteilnehmerInnen in einer engen Röhre. Im Magnetressonanztomografen wird die Hirnaktivität beobachtet. Dort Klavier zu spielen, scheint unmöglich. Deshalb wurde in Zusammenarbeit der Blüthner Pianofortemanufaktur ein Spezialklavier mit 27 Tasten entwickelt, das über eine Lichtleitung die Tastendrücke des Pianisten registriert. Als die StudienteilnehmerInnen die bildlich vorgegebenen Akkordfolgen spielten, zeigte sich, dass die beiden Planungsschritte "Was" und "Wie" auch zwei unterschiedliche Hirnnetzwerke aktivieren. Besonders auffällig war, dass sich beide Netzwerke über eine frontale Hirnregion erstrecken, die auch an der Planung sämtlicher Alltagshandlungen beteiligt ist: den linken lateralen Präfrontalkortex. Er arbeitet in Stufen, erklärt Erstautorin Roberta Bianco: Während der vordere Teil eher abstrakte Planungsschritte umsetzt, werden die Abläufe zum hinteren Teil der Region hin immer feingliedriger. Die Planung wird also immer

konkreter, es erfolgt eine Übersetzung vom Was zum Wie. Im Fall dieser Studie entspricht dies der Übersetzung einer musikalischen Idee in die Fingerbewegungen auf dem Klavier. Die Wissenschaftler:innen haben damit den Präfrontalkortex als zentrale Schlüsselregion identifiziert, die musikalische Kompositionen und Fingerbewegungen bei einer Solo-Performance koordiniert.

. Was macht Musik mit dem Gehirn?
Musik wirkt sich vielfältig auf unser Leben aus: Wir hören Musik zu unserer Unterhaltung, wählen je nach Stimmungslage fröhliche oder traurige Stücke aus, wir bewegen uns oder tanzen dazu, wir klatschen, schnippen und summen mit, wenn uns ein Musikstück begeistert. In den ersten 15 Lebensmonaten explodiert die Anzahl der Synapsen in den Gehirnen der sehr jungen Kinder; gleichzeitig werden zwischen den Hirnhälften Verbindungen geschaffen. Unser Gehirn bzw. die Synapsen werden einerseits ausgebaut, andererseits findet eine Art Selektion statt: die Wege, die häufig beansprucht werden, verstetigen sich. Wege, die selten genutzt werden, verkümmern. Ein Beispiel dazu: Bieten wir Kindern Aktivitäten wie Bewegen zur Musik an, werden unterschiedliche Hirnareale gleichzeitig aktiviert. Erlebt ein Kind mehrfach die Stimulation verschiedener Hirnareale durch ganzheitliche Angebote, bilden sich hier entsprechende Gedächtnisspuren. Bei vergleichbaren Aufgaben aktiviert unser Gehirn diese Wege wieder; die vormals angesprochenen verschiedenen Hirnareale werden zur Bewältigung der Aufgabe genutzt. Wir gelangen so gewissermaßen leichter zu einer Lösung. Und jetzt kommt es noch besser: Auch bei neuen und unbekannten Aufgaben, die sich uns stellen, nutzt unser Gehirn diese verstetigten Wege. Weil es sich als nützlich herausgestellt hat, aktiviert das Gehirn gleichzeitig mehrere Areale und verwendet beide Hirnhälften zur Verarbeitung von Sinneseindrücken. Und das kann zu besseren Lösungen in den genannten Entwicklungsbereichen führen (vgl. Hirler 2018, S. 6). Wie praktisch ist das denn? Auch wenn noch nicht abschließend geklärt ist, ob es sich um mehr oder weniger starke Korrelationen handelt oder ob kausale Zusammenhänge angenommen werden dürfen: In unseren element-i Kinderhäusern können wir musikalische Erfahrungen ermöglichen: ganzheitlich und bildungsbereichsübergreifend, vielfältig an unterschiedlichen Stationen des Tagesablaufs, für ein Miteinander, das Verbundenheit in der Gruppe fördert. Um das Bild von Gerald Hüther zu bemühen: Geben Sie den Kindern mit Musik, Tanz, Bewegung und Rhythmik das Kraftfutter, das ihre Gehirne zur Entfaltung brauchen.

. Was wird denn mit Bezug zu Musik erforscht?
In zahlreichen Artikeln wird von so genannten Transfereffekten gesprochen, die in nahe und weite Effekte unterschieden werden, oder von kreuzmodalen Einflüssen, die man genauer beleuchtet. Kurz gesagt, erforschen Wissenschaftler*innen, "... inwiefern musikalische Tätigkeiten nicht nur kognitive Fähigkeiten stimulieren, sondern auch zu einer Verbesserung der Fähigkeiten in einer Vielzahl von außermusikalischen Bereichen führen" (Zhang 2015, S. 5). Der Bezug von musiknahen Kompetenzen erschließt sich unmittelbar: Wer sich mit Musik befasst, sie hört, fühlt, sich nach ihrem Rhythmus bewegt, vielleicht im Grundschulalter ein Instrument erlernt, hat nachweisbar bessere feinmotorische Kompetenzen. Derjenige erkennt Stücke und Töne schneller und genauer wieder als Menschen, die kaum Bezüge zur Musik erlebt hatten (vgl. Zhang 2016, S. 6). Das leuchtet ein, das bestätigt uns unsere Lebenserfahrung ohne Frage. Zu nahezu allen Entwicklungsbereichen sind Forschungen durchgeführt worden: Musik hat positive Auswirkungen auf die sprachliche Entwicklung, auf die Entwicklung des Wortschatzes und später aufs Leseverständnis. Umgang mit Musik verbessert die Gedächtnisleistung, sogar das räumliche Denken, erhöht die Kreativität in Bezug auf Lösungen. Dass die sozial-emotionale Entwicklung durch aktives Musizieren beeinflusst wird, ist nicht nur eine Hypothese. Nein, auch dieser Bezug ist belegt. Vierjährige Kinder sind nach gemeinsamem Musizieren hilfsbereiter und kooperativer. In einer Studie wird der Schluss gezogen, dass durch das gemeinsame musikalische Erlebnis wie Singen intrinsische Wünsche, Emotionen und Erfahrungen in einer Gruppe teilen zu wollen, aufrechterhalten werden (Zhang 2015, S. 13f. bezieht sich auf eine Studie von Kirschner und Tomasello). Das sind wunderbare Nachrichten.

61 Schüler-Lehrer-Verhältnis

Für Motivation, Wohlbefinden und Leistung von Schulkindern spielt die persönliche Beziehung zu ihrer Lehrkraft eine große Rolle. Der Schlüssel ist emotionale Unterstützung, erklärt Bildungsforscherin Ann-Kathrin Jaekel.

Welche Auswirkungen hat eine gute Lehrer-Schüler-Beziehung? Was gehört zu einer guten Lehrer-Schüler-Beziehung? Wie lässt sich eine gute Lehrer-Schüler-Beziehung aufbauen?

Schülerinnen und Schüler haben das Bedürfnis nach persönlichem Kontakt zueinander und zu ihren Lehrkräften – das hat die Corona-Pandemie noch einmal ganz deutlich gezeigt. Als die Schulen geschlossen waren und die Kinder im sogenannten Distanzunterricht lernen mussten, hatten Videokonferenzen oder auch persönliche Treffen mit Lehrerinnen und Lehrern aus Schülersicht großen Einfluss auf die Unterrichtsqualität und auch auf ihre Freude am Lernen und ihre Anstrengungsbereitschaft. Das ist das Ergebnis einer Studie an der Universität Tübingen, die sich auf das Frühjahr 2020 bezieht. Insbesondere von den Lehrerinnen und Lehrern selbst erstellte Lernvideos wurden gut bewertet. "Da haben die Schülerinnen und Schüler das Gefühl, dass sich jemand Mühe gegeben und Zeit für sie genommen hat", sagt Studienautorin Ann-Kathrin Jaekel. Profi-Videos von externen Lernplattformen waren dagegen weniger beliebt.

Angesichts der vielen unterschiedlichen Unterrichtsmethoden in der ersten Phase des Distanzunterrichts seien diese Ergebnisse nicht unbedingt erwartbar gewesen, sagt Jaekel. Sie bestätigen aber, was bereits über die generelle Bedeutung der Lehrer-Schüler-Beziehung für die Unterrichtsqualität bekannt war – auch ganz ohne Pandemie. Nicht zuletzt in der umfangreichen Metaanalyse "Visible Learning" des neuseeländischen Bildungsforschers John Hattie steht das Verhältnis zwischen Lehrkraft und Schülerin oder Schüler weit oben auf der Rangliste der Einflussfaktoren für den schulischen Lernerfolg.

Welche Auswirkungen hat eine gute Lehrer-Schüler-Beziehung?
Die Effekte der Lehrer-Schüler-Beziehung sind dabei vielfältig. Durch Studien gut belegt ist eine grundsätzlich positive Wirkung auf Motivation, Emotionen, Lernverhalten und dadurch auch auf die Leistung von Schülerinnen und Schülern. Bei einem guten Verhältnis zu ihren Lehrerinnen und Lehrern brechen diese auch seltener die Schule ab. Und da auch Lehrkräfte ein Bedürfnis nach persönlicher Bindung haben, sorgt eine gute Beziehung zu Schülerinnen und Schülern auch bei ihnen für mehr Motivation und Freude am Beruf – auch dazu gibt es mittlerweile immer mehr Untersuchungen.

Was gehört zu einer guten Lehrer-Schüler-Beziehung?
Doch was genau ist mit einer guten Lehrer-Schüler-Beziehung eigentlich gemeint? Viele Faktoren spielen hier eine Rolle.

Nähe, Wärme, Wertschätzung sind Schlagworte aus der Forschung. Es geht um das Gefühl sozialer Eingebundenheit durch sichere, positive Bindungen, die das Lernen begünstigen. "Man braucht keinen Kumpel als Lehrer, aber jemanden, der einen unterstützt und begleitet", sagt Bildungsforscherin Jaekel. "Wichtig ist zum Beispiel wie die Lehrkraft damit umgeht, wenn jemand einen Fehler macht. Putzt sie ihn runter oder lässt sie gar zu, dass er ausgelacht wird? Oder ist sie sensibel und ansprechbar, wenn jemand Hilfe braucht?"

"Man braucht keinen Kumpel als Lehrer, aber jemanden, der einen unterstützt und begleitet."

Bildungsforscherin Ann-Kathrin Jaekel

Letztlich zahlt im Unterrichtsalltag alles auf eine gute Lehrer-Schüler-Beziehung ein, was aus dem Modell der drei Basisdimensionen guten Unterrichts als konstruktive Unterstützung bekannt ist: eine gute Feedback- und Fehlerkultur, respektvoller und wertschätzender Umgang miteinander, Ansprechbarkeit, Geduld, Empathie, individuelle Hilfe bei Verständnisproblemen. Auch Fairness spielt eine Rolle, denn das Lehrer-Schüler-Verhältnis bleibt ein hierarchisches. "Die Schülerinnen und Schüler sind in der schwächeren Position in dem Sinne, dass sie kaum Handlungsspielraum haben, wenn sie sich unfair behandelt fühlen", betont Jaekel.

Wie lässt sich eine gute Lehrer-Schüler-Beziehung aufbauen?

Der Aufbau einer guten Lehrer-Schüler-Beziehung ist eine große Herausforderung für Lehrkräfte und setzt viel sozialemotionale Kompetenz voraus. Lehrerinnen und Lehrer sollten aktiv daran arbeiten und dafür auch ihre eigene Bindungsgeschichte und Verhaltensweise reflektieren, so die Empfehlung aus der Unterrichtsforschung. Wenn sie beispielsweise mit Bloßstellen oder Sarkasmus auf unerwünschtes Verhalten von Kindern reagieren, könnte das darauf zurückzuführen sein, dass sie selbst in der Vergangenheit unsichere Bindungen erlebt haben und etwa mit Ablehnung nicht gut umgehen können. "Auch Humor oder Sarkasmus kann Kinder verunsichern, wenn sie ihn nicht verstehen", so Jaekel, "dessen muss man sich bewusst sein".

Die Bildungsforscherin weist zudem darauf hin, dass die Lehrer-Schüler-Beziehung nicht losgelöst von weiteren Merkmalen guten Unterrichts zu sehen ist. Ein gutes Klassenmanagement etwa werde als Voraussetzung dafür betrachtet, dass die Schülerinnen und Schüler konzentriert arbeiten können und sich überhaupt erst ein gutes, unterstützendes Klassenklima herausbilden kann. "Ein Grundstock an Disziplin und Störungsfreiheit und die klare Kommunikation von Aufgaben und Regeln helfen dabei, eine gute Lehrer-Schüler-Beziehung aufzubauen, weil die Lehrkraft dann nicht die ganze Zeit damit beschäftigt ist zu verhindern, dass ihr die Schülerinnen und Schüler auf der Nase herumtanzen", erklärt Jaekel. Störungen des Unterrichts und undisziplinertes Verhalten der Schüler sind Studien zufolge auch Stressfaktoren für die Lehrkraft selbst. Sie können bei ihr zu emotionaler Erschöpfung und Demotivierung führen, was wiederum einer positiven Lehrer-Schüler-Beziehung im Wege steht.

Ann-Kathrin Jaekel

Zur Person

Ann-Kathrin Jaekel ist wissenschaftliche Mitarbeiterin am Hector-Institut für Empirische Bildungsforschung an der Universität Tübingen. Ihr Forschungsschwerpunkt ist das Thema Unterrichtsqualität. Sie hat Lehramt und Bildungswissenschaften studiert und mit einer Arbeit zur Nutzung von Schülerurteilen zur Erfassung von Unterrichtsqualität in Forschung und Schulpraxis promoviert.

Weiterführende Literatur

Aldrup, Karen, Klusmann, Uta, Lüdtke, Oliver, Göllner, Richard, Trautwein, Ulrich (2018). Student misbehavior and teacher well-being: Testing the mediating role of the teacher-student relationship. Learning and Instruction. Hagenauer, Gerda, Raufelder, Diana (2021). Lehrer-Schüler-Beziehung. Aus: T. Hascher et al. (Hrsg.), Handbuch Schulforschung, Springer Fachmedien. Jaekel, A.-K., Scheiter, K., & Göllner, R. (2021). Distance Teaching During the COVID-19 Crisis: Social Connectedness Matters Most for Teaching Quality and Students' Learning. AERA Open.

Über die Autorin

Nina Voigt
Redaktion Online-Magazin schulmanagement

Mit Freude unterrichten

Die Beziehung zu den Schüler/innen ist wichtig, aber. . .

Die Hattie-Studie (2007) hat bestätigt, was zuvor bereits viele andere Untersuchungen zeigen konnten: Der Lernerfolg steht und fällt mit der Lehrperson. Dabei ist zentral, dass es ihr gelingt, zu den Schüler/innen eine gute Beziehung aufzubauen. Wenn wir mit Lehrkräften über die Bedeutung der Beziehung sprechen, ernten wir stets Kopfnicken. Alle sind sich bewusst, wie entscheidend dieser Faktor ist. In den Pausen oder beim Mittagessen kommen jedoch stets die gleichen, etwas schüchtern vorgetragenen Einwände einzelner Lehrer/innen: «Ich habe 25 Lernende in der Klasse – ich weiß nicht, wie ich zu allen eine Beziehung aufbauen soll? Mir fehlt die Zeit dafür.» «Klar ist die Beziehung wichtig, aber ich bin Fachlehrerin und sehe die Schüler/innen nur ein paar Stunden pro Woche. Da ist es kaum möglich, eine Beziehung aufzubauen, oder nicht?» Wir möchten diese Gedanken gerne entkräften, denn: Sie stimmen nicht und führen dazu, dass Sie Ihre Position und Ihren Einfluss als Lehrer/in unnötig schwächen und den wichtigsten Faktor für den Lernerfolg dem Zufall überlassen. Ihre Schüler/innen bauen in jedem Fall eine Beziehung zu Ihnen auf Wenn Sie Fachlehrerin sind, in Teilzeit arbeiten oder viele Schüler/innen haben, kann es gut sein, dass Sie nicht alle Schüler/innen auf einer persönlichen Ebene kennenlernen und eine Beziehung zu ihnen aufbauen können. Einige werden Ihnen bereits in den ersten Stunden auffallen, von anderen wissen Sie vielleicht nach einem Jahr lediglich den Namen. Der Denkfehler passiert da, wo Sie von sich auf die Schüler/innen schließen! Diese bauen in jedem Fall eine Beziehung zu Ihnen auf. Ihre Aufmerksamkeit verteilt sich auf 20 oder 25 Schüler/innen. Diese aber können sich ganz auf Sie konzentrieren. Sie können den Unterschied sofort nachvollziehen, wenn Sie auf der «Schülerseite» stehen. Denken Sie an die letzten Lehrerfortbildungen zurück: Die Person vorne ist nicht in der Lage, sich von allen Lehrpersonen ein Bild zu machen. Vielleicht spürt sie da und dort Widerstand oder Zustimmung. Ganz anders sieht es bei Ihnen und Ihren Kolleginnen aus. Sie wissen nach ein paar Lektionen, ob Sie die Referierende mögen, ob Sie sie als kompetent und glaubwürdig empfinden. Sie würden wahrscheinlich sogar sagen können, ob Sie die Dozentin oder den Dozenten als wertschätzend, einfühlsam oder im Gegenteil als blasiert oder distanziert empfanden. Wenn Sie die letzten drei oder vier Weiterbildungen gedanklich Revue passieren lassen und sich die Dozierenden in Erinnerung rufen: Könnten Sie sagen, mit welchen Sie gerne ein persönliches Gespräch über Ihren Unterricht führen würden? Bei wem Sie sich so wohl fühlten, dass Sie einem Unterrichtsbesuch nicht abgeneigt wären? Wem Sie ein konstruktives Feedback zutrauen würden? Und wem Sie keine Minute freiwillig Ihr Gehör schenken möchten? Wahrscheinlich schon. Sie können noch einen Schritt weitergehen. Denken Sie an ein Interview oder ein Buch, das Sie gelesen haben oder ein längeres Statement eines Talkshowgasts. Sie werden sofort bemerken, dass Sie diese Person zwar nicht persönlich kennen, aber sagen können, ob sie Ihnen sympathisch ist oder nicht und ob Sie – unabhängig von den Inhalten – von ihr lernen möchten. Sie können sich nicht entscheiden, ob Sie eine Beziehung zu Ihren Schüler/innen aufbauen – sondern nur, ob Sie diese aktiv mitgestalten oder dem Zufall überlassen. Womit wir bei der zweiten Frage sind.

Soll ich als Lehrperson die Beziehung zu den Schülerinnen bewusst gestalten?

Wenn wir über diesen Punkt sprechen, begegnen uns in Fortbildungen oft weitere Bedenken. Lehrpersonen sagen beispielsweise: «Muss ich mich jetzt bei den Schüler/innen

einschleimen?» «Ich finde, man muss authentisch bleiben.» «Ich will die Schüler nicht für jeden Mist loben müssen!» Aber geht es wirklich darum, wenn wir eine Beziehung zur Klasse aufbauen möchten? Ums schleimen, sich verstellen und loben? Schüler/innen wollen ganz andere Dinge von ihren Lehrkräften. Sie wollen genau das Gleiche wie Sie als Lehrer/in in einer Fortbildung. Sie möchten jemanden dort vorne, der oder die: echt ist
einen wertschätzenden Umgang pflegt
sich einfühlen kann
die Klasse ernst nimmt, an deren Wohlbefinden interessiert ist und sie mitreden, vielleicht sogar mitentscheiden lässt
den Schüler/innen etwas zutraut
die Schüler/innen sieht, dort abholt, wo sie stehen, und Forderungen stellt, die die einzelnen Kinder oder Jugendlichen mit Anstrengung erfüllen können
Respekt nicht nur einfordert, sondern auch zeigt
Humor hat
fair ist.
Sie müssen kein quasi-therapeutisches Gespräch mit den Schüler/innen führen, um für einen guten Draht zu sorgen. Es sind die kleinen Dinge im Alltag, die ins Gewicht fallen. Und Sie müssen nicht das gesamte Repertoire der Beziehungsgestaltung beherrschen: Sie können sich auf das fokussieren, was Ihnen leicht fällt und sich natürlich anfühlt – und auf diese Aspekte bewusst mehr Wert legen.

Mit kleinen, alltäglichen Handlungen eine Beziehung aufbauen und stärken
Herr Dachs Nehmen Sie sich einen kurzen Moment Zeit und denken Sie an Ihre Schulzeit zurück. Zu welchen Lehrpersonen sind Sie gerne in den Unterricht gegangen? Wieso? Was hat diese ausgezeichnet? Wenn wir diese Frage in Lehrerkollegien stellen, werden meist die oben aufgezählten Punkte genannt. Sogar wenn wir fragen, bei welchen Lehrpersonen sie viel gelernt haben und wieso, wird kaum auf die Didaktik eingegangen (sie hat viel Gruppenarbeit gemacht etc.), sondern immer wieder auf die Beziehung verwiesen. Es lohnt sich, dieser Fragen nachzugehen. Sie rufen sich damit lebendige und vertraute Beispiele in Erinnerung, von denen Sie sich etwas abschauen können. Mir, Fabian, fallen dazu viele Beispiele aus meiner eigenen Schulzeit ein. Du bist willkommen. Ich sehe dich und nehme dich an, wie du bist. Ich war ein sehr verträumter Schüler. Aufgrund mangelnder Schulreife besuchte ich ein zusätzliches Jahr den Kindergarten. Trotzdem ist mir von der Kindergärtnerin nichts anderes in Erinnerung geblieben als ihr strenger Blick und die Frage: «Fabian! Hörst du zu?!» Ganz anders war meine Erst- und Zweitklasslehrerin. Ich war so blockiert, dass ich mich im ersten Schuljahr weigerte, zu lesen. Sie ließ mir Zeit. Wenn ich in Tagträume abdriftete, sagte sie in leisem, warmem Ton meinen Namen – ohne Kritik. Wenn ich aufpasste, lächelte sie mich an. Ich fühlte mich wohl und konnte mich mehr und mehr entspannen und mich dem Unterricht zuwenden. Ihre Haltung war klar, nämlich:
Ich sehe dich
Hier bist du sicher und darfst dich entspannen
Du machst das in deinem Tempo – ich freue mich, wenn du aufpasst und sehe es dir nach, wenn es mal nicht so ist
Zeig mal, was hast du da gemacht?
Während sich mein Viertklasslehrer an meinen Aufsätzen und meinen "komischen Fantasien" störte, mochte mein Fünftklasslehrer genau dies. Im Kunstunterrichts bei ihm zeichneten meine Mitschüler Blumen, Vasen und Häuser. Mein bester Freund und ich brachten hunderte von winzigen Strichmännchen zu Papier, die allerlei komische und witzige Dinge erlebten. Er ärgerte sich nicht darüber, dass wir «den Auftrag falschen verstanden» hätten und ließ sich geduldig und belustigt die Zeichnungen erklären. Meine Aufsätze erzielten aufgrund der vielen Rechtschreibfehler zwar keine Glanznoten, aber er gestand mir, dass er sie zu Hause in einem Ordner sammelt, weil er sie originell findet. Bei ihm hatte ich immer das Gefühl, dass es wichtigeres gibt als Noten und Anpassung. Zum Beispiel Kreativität und Humor.
Mir gefällt dein Einsatz!
Als ich aufs Gymnasium wechselte, war ich der Einzige, der kein Instrument spielte. Bereits die erste Prüfung geriet deshalb zum Desaster. Der Musiklehrer ließ ein Klassikstück ertönen, wozu wir im Anschluss Fragen beantworten sollten: zu Rhythmus, Melodie, Instrumenten. Alle erzielten in diesem Test die Note "gut" oder "sehr gut". Ich hatte eine "Mangelhaft" und wurde von zwei Klassenkameraden deswegen aufgezogen. Völlig frustriert ging ich nach Hause. Auf dem Esstisch lag ein Brief – für mich! Darin stand: Lieber Fabian, es hat mir so leidgetan, dass ich dir so eine schlechte Note geben musste. Ich möchte, dass du weißt, wie sehr ich dich schätze und wie gerne ich es sehe, dass du im Unterricht so gut mitmachst. Ich hoffe, wir haben es beide weiterhin so gut miteinander. Dein Singlehrer
Was soll ich sagen? Natürlich hatten wir es gut miteinander! Er hatte zwei Jahre lang einen sehr schlechten, aber begeisterten Schüler in seinem Unterricht. Wir hatten auch Lehrer/innen, die bei falschen Antworten wütend wurden und herablassende Kommentare von sich gaben. Von meinem Singlehrer habe ich gelernt, dass sich auch «schwache» Schüler im Unterricht wohl fühlen können, wenn ihnen klar vermittelt wird, dass die Lehrperson zwischen der Leistungs- und der Beziehungsebene trennen kann.

Wer bist du und was interessiert dich?
Mein Biologielehrer am Gynmasium hatte eine sehr zeitsparende Methode, den Unterricht vorzubereiten: Er verzichtete praktisch darauf. Wir lasen gemeinsam Abschnitt für Abschnitt im Biologiebuch und sprachen darüber. Ab und zu erläuterte er etwas ein wenig genauer an der Tafel. Etwas Besonderes waren seine Prüfungen, bei denen man sehr viel nachdenken musste und die auf den Zehntel genau bewertet wurden. Eine 6 (die beste Note in der Schweiz) zu schaffen, war fast unmöglich, eine 5.8 oder 5.9 ein wahrer Triumph. Zu seinen Prüfungen meinte er mit breitem Grinsen: «Ich finde immer etwas, das ihr nicht gelernt habt!». Wir mochten den Kerl. Das lag unter anderem daran, dass er immer so aussah, als hätte er es nicht eilig. War die letzte Schulstunde Biologie, blieb meist ein kleines Grüppchen noch mit ihm im Zimmer. Wir setzten uns auf die Tischplatte eines Pults der ersten Reihe, er blieb auf seinem Drehstuhl vorne sitzen und wir plauderten über Gott und die Welt. Von ihm habe ich gelernt, wie wichtig es ist, dass man als Schüler gesehen und als ganzer Mensch wahrgenommen wird – und dass es manchmal wertvoller sein kann, etwas Zeit für die Beziehungspflege als für eine allzu akribische Vorbereitung aufzuwenden.

Guten Morgen, Fabian
Wie begrüßen Sie Ihre Schüler/innen am Morgen? Mehrere unserer Lehrer/innen hießen uns mit einem Händeschütteln willkommen. Sie standen an der Tür, schauten einem in die Augen, sagten etwas wie «Guten Morgen, Fabian» und manchmal bemerkten sie sogar, dass man die Haare geschnitten hatte, man nicht so gut drauf war oder besonders fröhlich wirkte. Studien aus dem Classroom-Management zeigen, dass Lehrpersonen, die ihre Schüler/innen persönlich begrüßen, weniger Disziplinprobleme haben also solche, die eher mit sich und ihren Materialien beschäftigt sind, bis der Unterricht beginnt oder sich als Allererstes darüber aufregen, dass zwei zu spät sind, bevor sie die begrüßen, die pünktlich waren.

Schade, dass du dich nicht für mein Fach interessierst – und ich mag dich
Mathematik interessierte mich am Gymnasium nicht im mindesten. Differential- und Integralgleichungen zu lösen, erschien mir so ziemlich das Sinnloseste zu sein, womit man sich beschäftigen kann. Im Unterricht hörte ich selten zu und bei der Matura (Abitur) achtete ich darauf, dass meine Vornoten und die Zensuren in den anderen Fächern so gut waren, dass ich in Mathematik ein leeres Blatt abgeben konnte. Ich schrieb lediglich einen kurzen Text für meinen Lehrer, weil ich ihn wissen lassen wollte, dass ich zwar nichts für die Prüfung gelernt hatte, ihn aber stets als fairen und guten

Lehrer empfand. Vor der Zeugnisvergabe meinte er zu mir: «Du, ich hatte bereits Angst, dass du wegen Mathematik nicht durchkommst – aber ich habe gesehen, dass du zum Glück in den anderen Fächern sehr gut warst.» Er freute sich für mich. Manchmal fühlen wir uns als Lehrperson zu verantwortlich für die Motivation und das Interesse der Schüler/innen – und reagieren entsprechend gekränkt, verunsichert oder verärgert, wenn sich ein Lernender nicht für unser Fach begeistern kann. Mein Mathematiklehrer gestaltete den Unterricht so motivierend wie möglich und wusste, dass er dennoch nicht alle erreichen kann – und er schaffte es, die Beziehungsebene und das Interesse an den Inhalten auseinanderzuhalten.

Wer sind Ihre Vorbilder?
Wir könnten noch viele Beispiele erwähnen. Vielleicht haben Sie sich in der einen oder anderen Lehrperson wiedergefunden? Oder etwas entdeckt, worauf Sie ebenfalls mehr Wert legen möchten? Vielleicht sind Ihnen eigene Vorbilder und Beispiele eingefallen, an denen Sie sich vermehrt orientieren möchten? Wenn Lehrpersonen uns fragen, wie sie mit «schwierigen» Schüler/innen umgehen können, bitten wir sie oft, sich als Teil der Unterrichtsvorbereitung jeweils kurz zu überlegen, wie Sie dieser Schülerin/ diesem Schüler heute ein positives Beziehungssignal senden könnten. Das kann ein Lächeln oder Nicken im richtigen Moment sein, eine positive Rückmeldung zu einem Aspekt, der ihm oder ihr schwerfällt, eine persönliche Bemerkung oder ein kurzes Gespräch über etwas Außerschulisches. Oft berichten uns die Lehrpersonen, dass sie schon bald eine Veränderung bei diesem Kind feststellen konnten und sich die Situation entspannt hat.

62 ©copyrights, Haftungsausschluss

©Copyrights by ©F.Klingler KN, ©M.Signitzer SM, ©R.Salvador SV, ©G.Schlemmer XM, ©C.Schönherr XH, ©A.Stumpfel YU, ©G.Steinwender YW

Die Angaben in diesem Dokument sind ohne Rücksicht auf Patentschutz oder Urheberrechte angeführt, und die Verwendung ist ausschliesslich auf den Schulunterricht an der HTL Innsbruck Anichstraße zur Verwendung als Diskussionsgrundlage in den unterrichteten Klassen eingeschränkt; eine andere Nutzung, gewerbliche Nutzung, Verbreitung, Veröffentlichung, Wiederveröffentlichung udergl. ist untersagt und widerrechtlich. Warennamen, Produktbezeichnungen, Logos, Abbildungen, Zitate udergl. sind ohne Gewährleistung freier Verwendbarkeit angeführt. Es wird mit äusserster Sorgfalt vorgegangen, was jedoch Fehler und Irrtümer leider nicht ausschließt, für deren Folgen keine wie immer geartete juristische Verantwortung oder Haftung übernommen werden kann. Ich erkläre den Inhalt dieses Dokuments jedem Zeitpunkt nach der Erstellung als widerrufen und gegenstandslos. Die Gestaltungen und Ausführungen sind in jeder rechtlichen Hinsicht bzgl. deren Ernsthaftigkeit insgesamt als frei und wirr zusammenphantasierte, unwahre, unreflektierte, inhaltlich unsinnige sprachliche Kunstwerke zu betrachten. Alle Rechte incl. fotomechanischer oder elektronischer Wiedergabe, Speicherung oder Übertragung vorbehalten.

Angaben nach TDG (Teledienstgesetz)

Verantwortlich für den Inhalt: XH

Schutzrechte: Die Verwendung von grafischen Elementen dieser und aller meiner Werke und Auftritte in elektronischen und nichtelektronischen Medien ist nur mit meiner ausdrücklichen schriftlichen Zustimmung erlaubt. Sämtliche Urheber-, Schutz- und sonstige Nutzungsrechte liegen bei mir. Dies gilt nicht für Texte, Grafiken und Bilder, die mir freundlicherweise zur Verfügung gestellt wurden. Hier gilt das Urheberrecht des jeweiligen Verfassers uneingeschränkt. Alle Rechte vorbehalten.

Rechtliche Hinweise: 1. Urheberrecht: Meine Werke und Auftritte genießen urheberrechtlichen Schutz. Insbesondere Vervielfältigungen, Übersetzungen und die Einspeicherung und Verarbeitung in anderen elektronischen Medien sind urheberrechtlich geschützt. Nachahmung und Verwertung - auch auszugsweise - sind nur mit meiner schriftlichen Genehmigung statthaft. Inhalte und Strukturen sind urheberrechtlich geschützt. Die Vervielfältigung von Informationen oder Daten, insbesondere die Verwendung von Texten, Textteilen oder Bildmaterial, bedarf der vorherigen schriftlichen Zustimmung. Die Abbildungen genießen den Schutz des § 72 UrhG. Die Veröffentlichungs- und Vervielfältigungsrechte liegen bei mir. Die Rechte bleiben auch in vollem Umfang bestehen, wenn Bilder elektronisch oder händisch in ein Archiv übernommen werden. 2. Haftungsausschluß für Seiten/Darbietungen Dritter: a. Die Werke und Auftritte enthalten auch Verknüpfungen (sog. "Hyperlinks") zu Werken und Websites im Internet, die von Dritten gepflegt werden und deren Inhalte mir nicht bekannt sind. Ich vermittle lediglich den Zugang zu diesen und übernehme keinerlei Verantwortung für deren Inhalte. Meine Links auf fremde Werke und Internetseiten dienen lediglich zur Erleichterung Ihrer Navigation. Ich mache mir die auf verwiesenen/verlinkten Werke und Auftritte dargestellten Aussagen nicht zu eigen. Insbesondere hafte ich nicht für dort begangene Verstöße gegen gesetzliche Bestimmungen und Rechte Dritter.

b. Die Inhaber der Werke und Internetseiten, zu denen über die von mir erstellten Werke und betriebenen Internetauftritte Hyperlinks bestehen, sind sowohl für deren Inhalt als auch für den Verkauf der dort angebotenen Produkte und die Abwicklung der Bestellung allein verantwortlich. c. Ich hafte nicht für die Verletzung von Urheberrechten, Marken und Persönlichkeitsrechten, die auf einer mit einem Hyperlink versehenen Seite begangen werden. d. Im Falle einer Bestellung kommt lediglich ein Vertrag zwischen dem Nutzer und dem jeweiligen Inhaber der Internetseite bzw. dem dort präsenten Anbieter, in keinem Falle jedoch aber ein Vertrag zwischen mir und dem Nutzer zustande. Bitte beachten Sie die allgemeinen Geschäftsbedingungen des jeweiligen Anbieters der verlinkten Internetseite.

3. KEINE ABMAHNUNG OHNE KONTAKTAUFNAHME: Sollte irgendwelcher Inhalt oder die designtechnische Gestaltung einzelner Angebotsseiten oder Teile dieses Angebots/Werks fremde Rechte Dritter oder gesetzliche Bestimmungen verletzen oder anderweitig in irgendeiner Form wettbewerbsrechtliche Probleme hervorbringen, so bitte ich unter Berufung auf Phar. 8 Abs. 4 UWG, um eine angemessene, ausreichend erlaeuternde und schnelle Nachricht ohne Kostennote. Ich garantiere, dass die zu Recht beanstandeten Passagen oder Teile dieser Angebots(web)seiten in angemessener Frist entfernt bzw. den rechtlichen Vorgaben umfänglich angepasst werden, ohne dass von Ihrer Seite die Einschaltung eines Rechtsbeistandes erforderlich ist. Die Einschaltung eines Anwaltes, zur für den Dienstanbieter kostenpflichtigen Abmahnung, entspricht nicht dessen wirklichen oder mutmasslichen Willen und würde damit einen Verstoß gegen Phar. 13 Abs. 5 UWG, wegen der Verfolgung sachfremder Ziele als beherrschendes Motiv der Verfahrenseinleitung, insbesondere einer Kostenerzielungsabsicht als eigentliche Triebfeder, sowie einen Verstoß gegen die Schadensminderungspflicht darstellen.

Haftungsauschluss

* Der Inhalt dieser Präsentation wurde unter angemessener Sorgfalt erstellt
* Allerdings erfolgt keine Gewähr, dass die Inhalte korrekt, vollständig oder aktuell sind

Jegliche Nutzung der Inhalte erfolgt auf eigene Gefahr, unter Ausschluß eines Anspruches auf Schadenersatz, weder für materielle noch immaterielle Schäden, so wie körperliche Schäden

* Die Überlassung der Präsentation erfolgt nur für den internen Gebrauch des Empfängers ohne Veröffentlichung auf WEB-Seiten oder nach Anfrage
* Die Präsentation stellt keine Beratung dar

Abmahnungsbestimmungen:

* Sollte irgendwelcher Inhalt oder die designtechnische Gestaltung einzelner Seiten oder Teile dieser fremde Rechte Dritter oder gesetzliche Bestimmungen verletzen oder anderweitig in irgendeiner Form wettbewerbsrechtliche Probleme hervorbringen, so bitten wir unter Berufung auf †8 Abs. 4 UWG, um eine angemessene, ausreichend erläuternde und schnelle Nachricht ohne Kostennote
* Dennoch von Ihnen ohne vorherige Kontaktaufnahme ausgelöste Kosten werden wir gänzlich zurückweisen und gegebenenfalls Gegenklage wegen Verletzung vorgenannter Bestimmungen einreichen.

Sonstiges:

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MER-

CHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Die Benutzungsbedingungen unterliegen österreichischem Recht. Gerichtsstand für Streitigkeiten, die meine Auftritte betreffen, ist Innsbruck. XH

63 Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Todo list