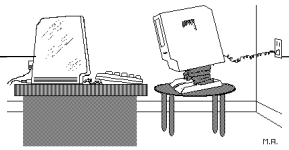
Steuern, Regeln & Computer ?



Lernen in 15 Stationen

1. **Benutzer-Schnittstelle**: Oberfläche eines Computers
   PowerBook
   Software: „MacIntro”

2. **Logische Schaltungen**: NOT, AND, OR, NAND, NOR
   Gerät für logische Schaltungen
   Physik-Buch S.74 ff, Technik-Buch S. 85 ff

3. **bit und byte**: dezimale und binäre Schreibweise von Zahlen
   Mac + SIOS-Interface+ Kartenleser
   Software: „Kartenleser”

4. **Messen - Steuern - Regeln**: analog und digital
   Mac + SIOS-Interface
   Software: „SIOS - Grundlagen”

5. **Daten und Adressen**: Befehle in Maschinencode
   Kosmos CP1, Befehlsliste

6. **Per Bus zum Prozessor und zurück**: RAM und ROM, Akku und Register
   PowerBook / Mac
   Software: „SimuComp”, Technik-Buch S. 88 f

7. **Interface**: Schnittstelle für Peripheriegeräte
   Kosmos CP1
   Technik-Buch S.93, Abb. 3-5

8. **Steuern**: Digital-Ausgänge, seriell und parallel
   Mac + SIOS-Interface
   Software: „SIOS Steuern”, Modelle: Ampelkreuzung, 7-Segment-Anzeige

9. **Messen**: digitaler und analoger Eingang, ASCII-Code
   Mac + SIOS-Interface + Poti + NTC + LDR + Meßfühler
   Software: „SIOS Messen”

10. **Programmieren in BASIC**: Ampelkreuzung, zeitgesteuert
    Schneider CPC mit BASIC + Interface + Ampelkreuzung-Modell
    Technik-Buch S. 94 ff

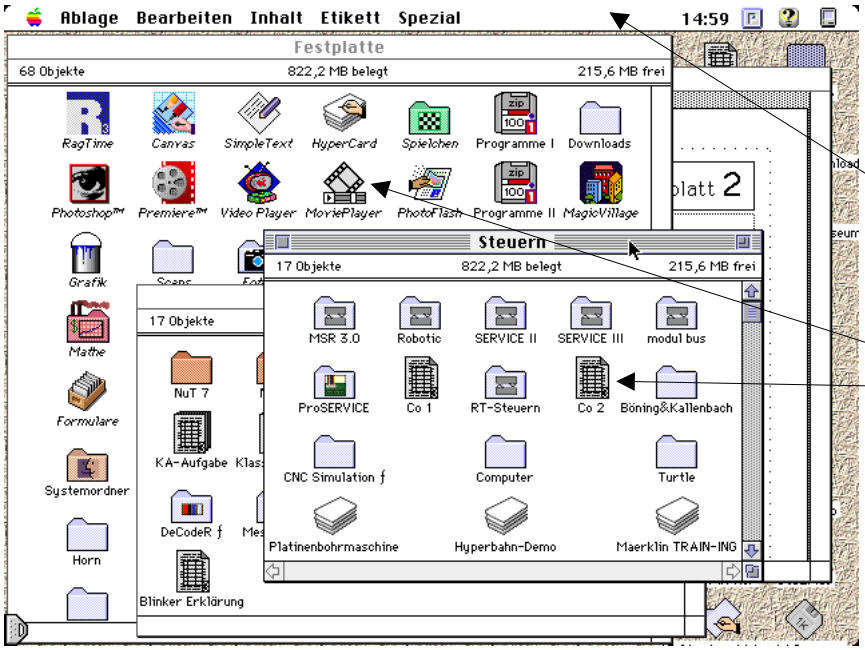
11. **Steuern mit Rückmeldung**: Taster und Impulse
    Mac + SIOS-Interface+ Motoren + Taster
    Software: „RoboticPerformer”

12. **Steuern mit Schrittmotor**: präzises Steuern ohne Rückmeldung
    Mac + SIOS-Interface + Aufzug-Modell
    Software: „Aufzug”, Technik-Buch S. 100 f

13. **CNC**: Steuern von Werkzeugmaschinen
    Powerbook / Mac
    Software: „CNCSim” und „CNCSim Intro”, Technik-Buch S. 104 f

14. **Regeln**: Abschlußarbeit
    Mac + SIOS-Interface + Gebläse-Modell
    Software: „mac it”

15. **Präsentieren**: Buttons, Links, Sound, ...
    PowerBook / Mac
    Software: „HyperCardTour”



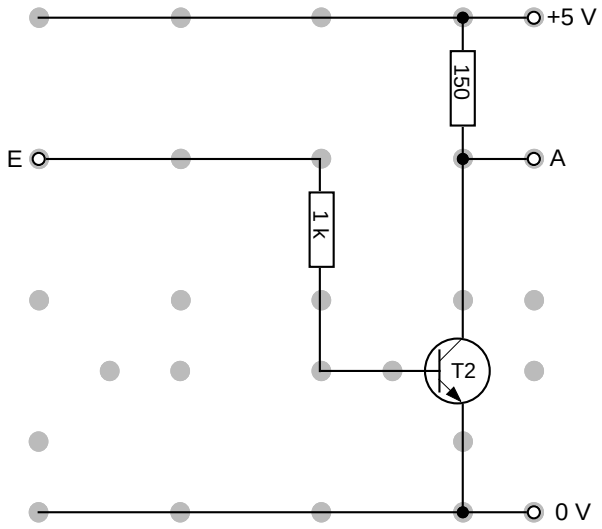
Setze im Text die passenden Worte ein und ordne weitere Begriffe durch Pfeile dem Bildschirmfoto zu!

Menüleiste	Maus
Menüs	DOS
Befehle	Windows
Icons	Server
Ordnern	Mac
Programme	MacOS
Dokumente	Unix
Fenster	drag & drop
Rollbalken	Mausklick
Rollpfeil	öffnen
Rollpfeil	Schließen
Rollbox	copy & paste
Schließfeld	Schnittstelle
aktiven	

Heute wird kaum noch ein Computer ohne _____ verkauft, was für _____-User lange Jahre nicht selbstverständlich war. Heute bietet auch das am weitesten verbreitete _____ eine grafische Benutzer-_____ wie sie der _____ schon immer hat. _____ ist das einzige aktuelle Computer-Betriebssystem ohne grafische Benutzer-Schnittstelle. Die meisten _____ arbeiten mit diesem sehr schnellen System. Auf Web-Servern wird am zweithäufigsten das _____ eingesetzt, das aus Kostengründen auch von den meisten Publishern (für Grafiken, Fotos, Zeitungen, Movies, CD-ROMs, Webseiten u.a.) bevorzugt wird.

_____ und _____ werden in einer grafischen Benutzer-Schnittstelle durch Symbole bzw. _____ dargestellt bzw. repräsentiert. Sie werden meist in _____ verwaltet. Per _____ & _____ lassen sich Programme, Dokumente und Ordner verschieben. In der _____ kann man per _____ verschiedene _____ öffnen, die mehrere _____ zur Verfügung stellen. Ein Dokument oder eine Programm oder einen Ordner kann man z.B. _____, indem man den Befehl „Öffnen” aus dem Menü „Ablage” wählt. Es wird dann in einem _____ dargestellt. Den Inhalt des _____ Fensters im Vordergrund kann man verschieben, indem man auf einen _____ oder einen _____ klickt oder indem man die _____ mit der Maus verschiebt. _____ kann man ein aktiviertes Fenster, indem man den Befehl „Schließen” aus dem Menü „Ablage” wählt oder indem man in das _____ klickt. Mit _____ & _____ kann man Texte, Bilder, Töne, Videos, Dokumente, Ordner und Programme vervielfältigen.

Gerät mit logischen Schaltungen, Physik-Buch S.74 ff, Technik-Buch S. 85 ff **Logische Schaltungen**



0 steht in der Computer-Technik für < 0.7 V (nein).
1 steht in der Computer-Technik für 2 - 5 V (ja).

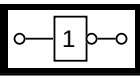
Wähle an dem Gerät für logische Schaltungen z.B. die ODER-Schaltung (OR) und finde heraus, ob am Ausgang Spannung (1) anliegt oder nicht (0), wenn die Eingänge .... Trage dein Ergebnis in der zugehörigen Tabelle ein. Fülle so alle Tabellen aus.

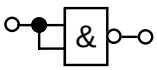
Prüfe, welche Schaltungen links dargestellt sind.

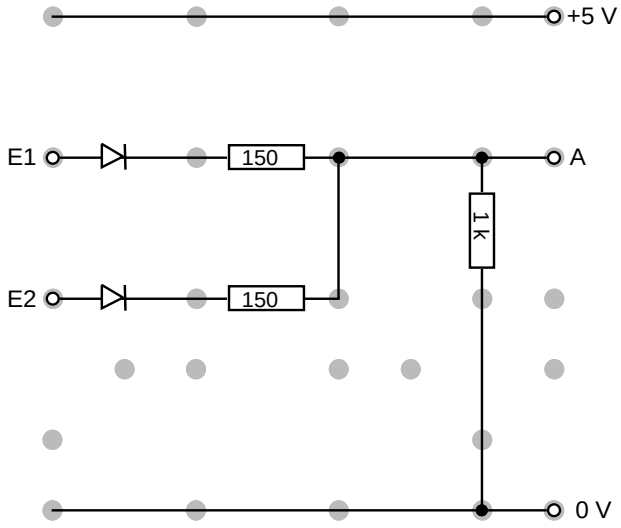
Mit einem IC, der vier NAND-Gatter enthält, kann man jede der hier gezeigten Schaltungen aufbauen.

NOT

E	A
0	➡
1	➡

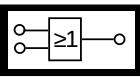


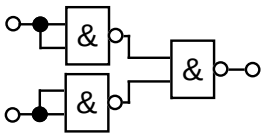




OR

E1	E2	A
0	0	➡
0	1	➡
1	0	➡
1	1	➡

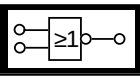


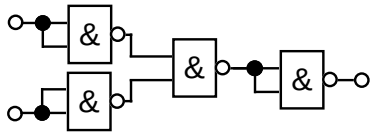


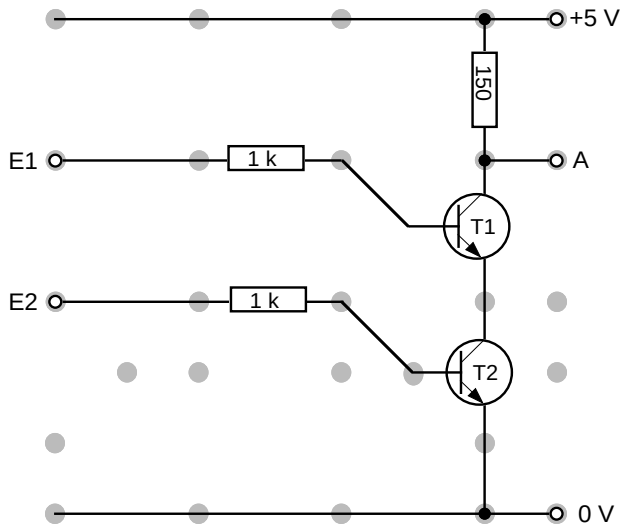
Schaltet man hinter das OR-Gatter ein NOT-Gatter... :

NOR

E1	E2	A
0	0	➡
0	1	➡
1	0	➡
1	1	➡

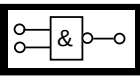


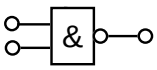




NAND

E1	E2	A
0	0	➡
0	1	➡
1	0	➡
1	1	➡

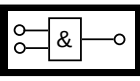


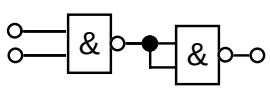


Schaltet man hinter das NAND-Gatter ein NOT-Gatter... :

AND

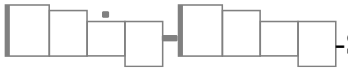
E1	E2	A
0	0	➡
0	1	➡
1	0	➡
1	1	➡





Mac + SIOS-Interface + Kartenleser mit Lochkarten + Software: „Kartenleser"

bit und byte

1 bit ist die kleinste denkbare Informationseinheit. Sie kann zwischen ___ Zuständen unterscheiden. Zum Speichern von 1 bit kann man z.B. ein Loch verwenden. Es kann voll oder leer sein, und so eine Information beinhalten. Auch eine  -Schaltung kann 1 bit speichern.

Die beiden möglichen Zustände eines Flip-Flop nennt man Flip und Flop, häufiger jedoch bezeichnet man sie mit **O** und **I**, wie in der nebenstehenden Tabelle. Betrachte in dieser Tabelle das mittlere der drei dick gerahmten Rechtecke und mache dir klar, daß man mit 2 bit 4 verschiedene Zustände speichern kann:

O O oder **O I** oder **I O** oder **I I**.

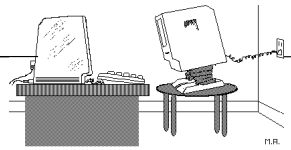
Wieviele verschiedene Zustände lassen sich mit 3, 4, 5, 6, 7 und 8 bit speichern?

bit Nr.	7	6	5	4	3	2	1	0
Wert	128	64	32	16	8	4	2	1
0	O	O	O	O	O	O	O	O
1	O	O	O	O	O	O	O	I
2	O	O	O	O	O	O	I	O
3	O	O	O	O	O	O	I	I
4	O	O	O	O	O	I	O	O
5	O	O	O	O	O	I	O	I
6	O	O	O	O	O	I	I	O
	O	O	O	O	O	I	I	I
	O	O	O	O	I	O	O	O
253								
254								
255								

Mit 1 bit gibt es 2 Möglichkeiten (2^1).
Mit 2 bit gibt es 4 Möglichkeiten (2^2).
Mit 3 bit gibt es _____ Möglichkeiten (2).
Mit 4 bit gibt es _____ Möglichkeiten (2).
Mit 5 bit gibt es _____ Möglichkeiten (2).
Mit 6 bit gibt es _____ Möglichkeiten (2).
Mit 7 bit gibt es _____ Möglichkeiten (2).
Mit 8 bit gibt es _____ Möglichkeiten (2).

1 byte besteht aus ____ bit und kann _____ verschiedene Zustände annehmen. Somit können durch 1 byte z.B. die Zahlen von 0 bis _____ repräsentiert werden.

Die Tabelle rechts zeigt einige dieser Zahlen in **Dezimal**- und in **Binär**schreibweise. Ergänze sie durch weitere Beispiele. (Beachte dabei, daß das bit 0 den Wert 1, das bit 1 den Wert 2, das bit 3 den Wert 4, ... hat.)



Notebookes werden komplett mit **Bildschirm**, **Tastatur** und **Trackball** oder **Trackpad** verkauft, sodaß man Daten in den Computer eingeben und aus ihm auslesen kann. Bei Schreibtisch-Computern sind diese Eingabe- und Ausgabegeräte getrennt vom eigentlichen Computer. Darüber hinaus gibt es noch viele andere **Peripheriegeräte**, die man an einen Computer anschließen kann:

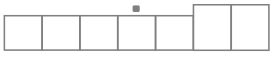
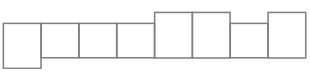
Geräte zur Eingabe von Informationen:	Geräte zur Ausgabe von Informationen:
Video-Kamera,	*Lautsprecher,*

Manche Geräte schließt man an einen Computer an, um etwas zu messen oder Vorgänge zu steuern oder zu regeln. Bei allen drei Aufgaben werden Informationen zwischen Computer und Peripherie-Gerät transportiert, nur in unterschiedlichen Richtungen:

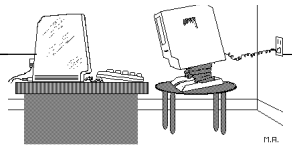
Computer	→	Gerät	
Computer	←	Gerät	
Computer	⇄	Gerät	

Will man Geräte an den Computer anschließen, ist es wichtig, zu wissen, ob es sich um **analoge** oder um **digitale** Geräte handelt. Analoge Geräte arbeiten stufenlos, sie können zumindest theoretisch beliebig viele Zustände annehmen. Digitale Geräte haben eine genau definierte Anzahl von Möglichkeiten, arbeiten also gestuft. (Bei 1 byte gibt es z.B. 256 Stufungen.)

Analoge Geräte sind z.B.:	**Digitale** Geräte sind z.B.:
Mikrofon, Lautsprecher,	*Digital-Fotoapparat, Tintenstrahl-Drucker,*

Zwischen Computer und Peripherie-Geräten werden Daten entweder  (bit für bit hintereinander in einer Leitung) oder  (1 byte gleichzeitig über 8 parallele Leitungen) übertragen.

Parallele Schnittstellen sind z.B. **Centronics** und **SCSI**, serielle sind z.B. **V24** und **RS 232**.



Den Datenspeicher eines Computers kann man sich als Schrank mit vielen Schubladen vorstellen. Beim Kosmos-Computer kann man diese Schubladen öffnen und etwas hineintun. (Auch bei anderen Computern ist das möglich, aber nicht ganz ungefährlich!) Beim Einschalten des Computers sind alle Schubladen leer. Wenn du in einer Schublade nachsehen willst, mußt du dem Computer sagen in welcher. Dazu hat jede Schublade eine **Adresse**. Tippe z.B. ein: **105 OUT.** Der Computer zeigt dir, was die Schublade enthält, nämlich nichts bzw. **00.000**. Lege jetzt eine anderes **Datum** (Einzahl von Daten) in die Schublade, indem du z.B. eintippst: **00022 INPUT**.

Versuche jetzt ein paar Daten unter verschiedenen Adressen abzulegen. Beachte dabei daß die Adressen mit drei Ziffern und die Daten mit fünf Ziffern eingegeben werden müssen. Also z.B. **120 OUT - 00230 INPUT**.

Vielleicht bemerkst du dabei, daß die Möglichkeiten dieses Computers recht beschränkt sind. Es stehen nur **128** Adressen zur Verfügung nämlich von **000** bis **127**. In jeder Adresse kann man nur Zahlen von **00000** bis **00255** eingeben, das sind **256** verschiedene Zahlen. Jede Schublade hat offensichtlich **8** Fächer (Flip-Flops), von denen jedes 1 **bit** speichern kann. Deshalb können unter jeder Adresse **1 byte** Daten gespeichert werden. Bei 128 Adressen sind das also insgesamt 128 byte.

Aktuelle Computer können z.B. in Ihrem **RAM** (Schreib-Lese-Speicher) je nach Ausstattung 8 Megabyte bis 256 Megabyte Daten speichern. 1 Megabyte sind ca. 1000 Kilobyte, 1 Kilobyte sind ca 1000 byte. (Genau: 1 kbyte = 1024 byte = 2^8 byte.)

Der **Akku** ist eine besondere "Schublade" des Computers, in die man nicht nur etwas hineintun kann (z.B. mit **AKO, LDA** oder **LIA**) - man kann zum Inhalt des Akku auch etwas addieren (mit **ADD**) oder von diesem etwas subtrahieren (**SUB**). Wegen dieser Eigenschaften wird er häufig mit neuen Daten geladen, daher der Name!

Daß ein Computer nicht nur Daten speichern sondern auch verarbeiten kann, läßt sich mit einem kleinen **Programm** zeigen, daß du unten auf dieser Seite findest.

Die Punkte dienen der besseren Lesbarkeit, sind aber nicht einzutippen. Zahlen vor dem Punkt sind **Befehle**. Wie du aus der Befehlsliste ersehen kannst, versteht der **Prozessor** dieses Computers 21 solcher Befehle.

Mit diesem **Maschinencode** werden in anderen Computern die Programmteile programmiert, bei denen es besonders auf schnelle Verarbeitung ankommt (z.B. im Betriebssystem). Weil andere Prozessoren größere Befehlssätze mit unterschiedlichen Codes haben, werden sie meist mit **Mnemonics** (Merkhilfen) programmiert, die dann ein **Assembler**-Programm in den Maschinencode übersetzt.

Das Programm kannst du starten mit: **001 - PC - RUN**.

Versuche anhand der Befehlsliste den Ablauf des Programmes zu verstehen und erläutere die einzelnen Befehle.

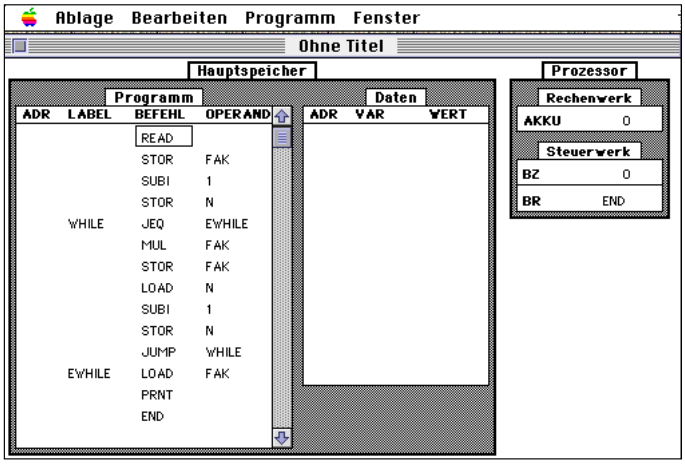
Adresse	Mnemonic	Datum
000		**00.128**
001	**AKO 001**	**04.001**
002	**P2A 000**	_____
003	**VZG 200**	**03.200**
004	**ABS 009**	_____
005	**____ 000**	**10.000**
006	**SPB 001**	_____
007	**ADD 009**	_____
008	**SPU 002**	_____
009		**00.001**

Erläuterungen:

Mac + Software: „SimuComp" + Technik-Buch S. 88 f.

Per Bus zum Prozessor und zurück

Starte auf einem Mac das Programm „SimuComp". Gib nebenstehendes Programm ein. Benutze Maus, Eingabe- und Cursortasten, um die entsprechenden Felder zu erreichen. Wähle dann aus dem Menü „Programm" den Befehl „Assemblieren". Wähle anschließend aus demselben Menü die Befehle „Bus-Animation" und „Befehlssatz 1" falls sie nicht bereits mit einem Haken versehen sind. Jetzt kannst du aus dem Menü „Programm" das „Programm starten". Du siehst, wie der Computer die „Fakultät" einer eingegebenen Zahl (max. 5) berechnet. Die Fakultät von 5 ist z.B.: 5·4·3·2·1. Beobachte vor allem, was im Prozessor geschieht.



Lies anschließend im Technik-Buch die Seiten 88 und 89. Kannst du jetzt den folgenden Text

Im [redacted] (**ReadOnlyMemory**) oder [redacted] des Computers sind z.B. Teile seines Betriebssystems so gespeichert, daß sie nicht verändert werden können. In das [redacted] (**RandomAccessMemory**) bzw. den [redacted] hingegen kann man Daten und Programme auch hinein schreiben.

Der [redacted] eines Computers besteht aus [redacted] und [redacted] und führt das Programm aus. Die [redacted] kommen in den **Befehlszeiger** (BZ), die [redacted] in das **Befehlsregister** (BR) und die Zahlen in den [redacted].

Transportiert werden die unterschiedlichen Informationen innerhalb des Computers in drei verschiedenen Leitungen: dem [redacted], dem [redacted] und dem [redacted]. Weil es auf Geschwindigkeit ankommt und nur kurze Wege zurückzulegen sind, werden immer mindestens 8 bit parallel transportiert. Es gibt aber auch 16 bit oder 32 bit breite Busse. Alle drei Busse führen auch zu [redacted] des Computers, damit die entsprechenden Informationen auch mit Peripheriegeräten ausgetauscht werden können.

Wichtig ist, daß alle Bauteile die Informationen im gleichen Takt senden, empfangen oder verarbeiten. Dieser Takt wird durch einen [redacted] bestimmt. Sein Takt wird in Megahertz ([redacted]) angegeben. Aktuelle Prozessoren arbeiten mit einer Frequenz von bis zu 500 MHz, können also mindestens 500 Millionen byte pro Sekunde verarbeiten.

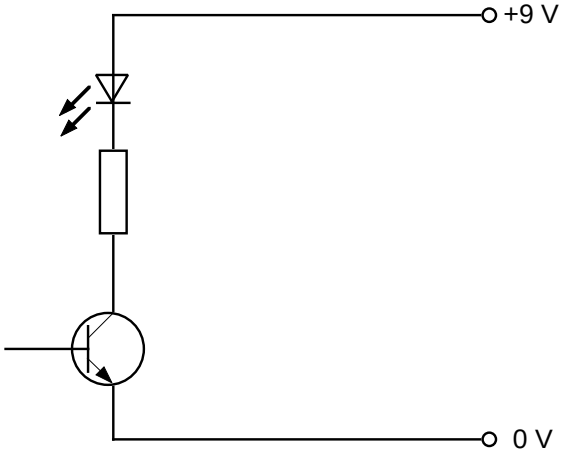
Kosmos-Computer CP1 mit Interface + Technik-Buch S. 93, Abb. 3-5

Interface

Für die Kommunikation mit Peripheriegeräten besitzen Computer sogenannte [redacted] oder [redacted]. Über diese können Daten z.B. an einen Drucker ausgegeben werden oder von einem Scanner eingelesen werden. Die Kontakte dieser Anschlüsse kann man per Software schalten, entweder auf [redacted] (high) oder auf [redacted] (low).

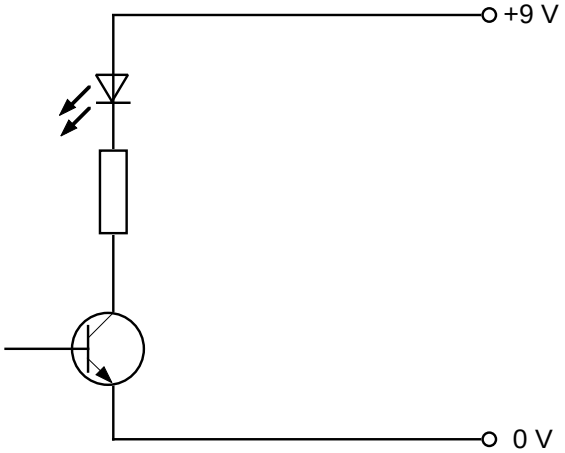
Skizze 1:

(Übernimm Abbildung 3 aus dem Buch!)



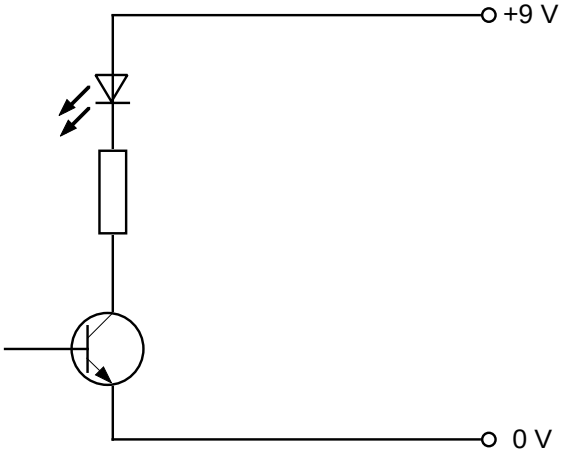
Skizze 2:

(Übernimm Abbildung 4 aus dem Buch!)



Skizze 3:

(Übernimm Abbildung 5 aus dem Buch!)

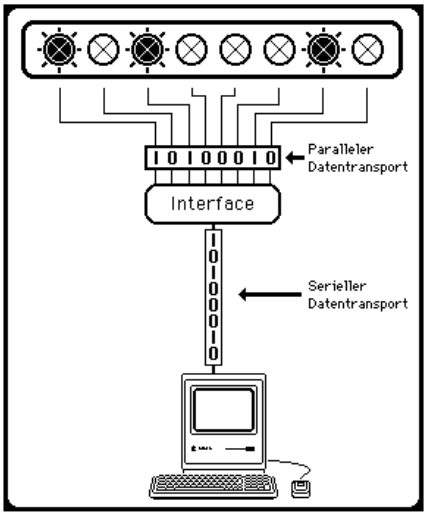


Weil das Innere eines Computers nur für sehr geringe Ströme ausgelegt ist, darf man an ihn keine Lampen oder Motoren anschließen, die zu starke Ströme entnehmen würden. Zum Anschluß solcher Geräte verwendet man ein **Interface**. Seine Transistoren erhalten Ihre Kollektor-Emitter-Spannung von einem seperaten Netzgerät, der Computer versorgt nur die Basis über einen Schutz-Widerstand mit Strom.
(Skizze 1)

Der Minus-Pol des Netzgerätes und Computer-Ground (**GND**) sind im Interface verbunden. Für die Anzeige des Schaltzustandes wird eine Leuchtdiode (**LED**) verwendet, zum Schalten von Verbrauchern ein **Relais** mit parallel geschalteter **Freilauf-Diode**. Da Computer und Verbraucher leitend miteinander verbunden sind, kann ein Kurzschluß oder Überlastung im Arbeitskreis Bauteile des Computers zerstören.
(Skizze 2)

Ein **Optokoppler** kann ein solches Interface sicherer machen. Das Licht einer LED, die vom Computer mit Strom versorgt wird, schaltet einen Fototransistor, der seinen Strom vom separaten Netzteil bezieht. Dadurch entfällt die leitende Verbindung zwischen Computer und Verbraucher.
(Skizze 3)

Der Kosmos-Computer CP1 hat ein einfaches Interface. Betrachte es von unten und vergleiche mit den Skizzen!



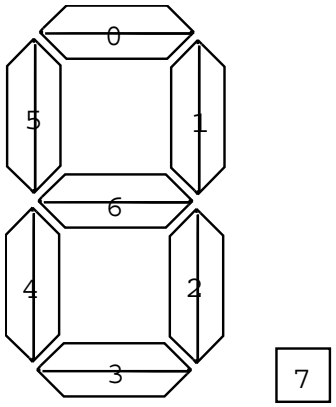
Um Geräte mit einem Computer steuern zu können, braucht man ein **Interface**, das den Computer vor zu großen Strömen schützt. Vom Computer werden die Daten **seriell** zum Interface übertragen. Dies dauert zwar etwas länger als eine **parallele** Datenübertragung, ist dafür aber sicherer und ermöglicht (bei höheren Spannungen) längere Kabel. Das Interface stellt die **8** Informationen eines bytes an seinen Ausgängen parallel zur Verfügung, sodaß sie von hier mit einem **Flachbandkabel** parallel zu den zu steuernden Geräten transportiert werden können.

An die 8 **digitalen** Ausgänge des Interface lassen sich maximal ___ Geräte(gruppen) anschließen. Man kann damit _____ verschiedene Schaltzustände erreichen. Ein Motor, der vorwärts und rückwärts laufen soll, benötigt allerdings 2 digitale Ausgänge.

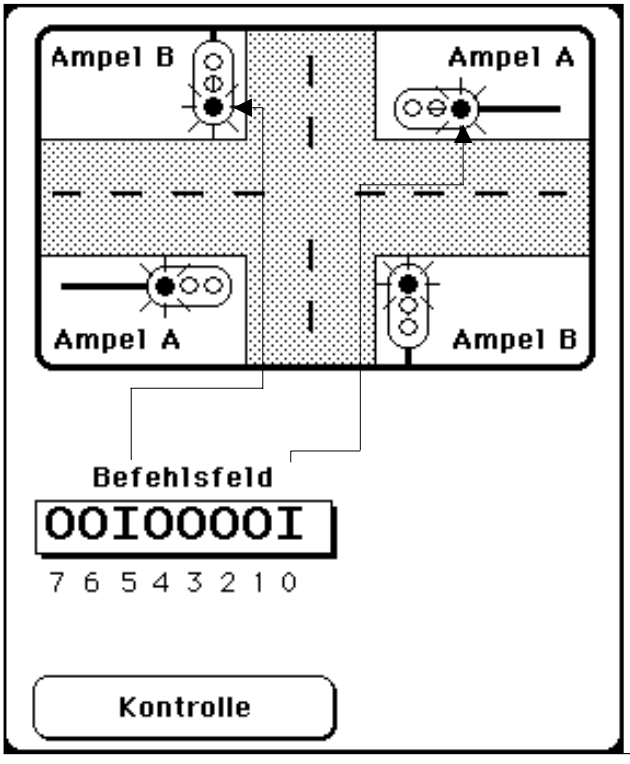
Mit den 7 Leuchtdioden einer 7-Segment-Anzeige kann man gut die Ziffern von 0 bis 9 darstellen. Eine 8. LED kann zusätzlich einen Punkt zeigen.

Eine solche Anzeige kann man an die 8 **Digitalausgänge** eines Interface anschließen. Welche bit müssen geschaltet werden, um die verschiedenen Ziffern darzustellen?

I = **EIN**, **O** = **AUS**



bit Nr.	7	6	5	4	3	2	1	0
Ziffer **1**	O	O	O	O	O	I	I	O
Ziffer **2**								
Ziffer **3**								
Ziffer **4**								
Ziffer **5**								
Ziffer **6**								
Ziffer **7**								
Ziffer **8**								
Ziffer **9**								
Ziffer **0**								



Ein einfaches Steuer-**Programm** besteht aus einer zeitlich festgelegten Abfolge von Steuer-Befehlen, denen jeweils eine Zeitdauer zugeordnet ist.

Schreibe ein Programm für eine Ampel-Kreuzung. Zeichne dazu zunächst in der Skizze links ein, welches bit welche Lampe(n) steuert. Ergänze dann die untenstehende Tabelle.

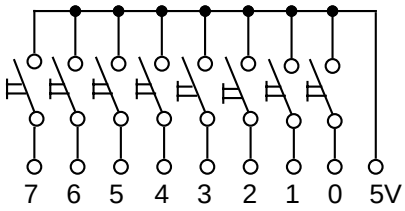
	-	-	Straße A			Straße B			
	-	-	grün	gelb	rot	grün	gelb	rot	
bit	7	6	5	4	3	2	1	0	Dauer
Wert	128	64	32	16	8	4	2	1	
Phase 1	O	O	I	O	O	O	O	I	
Phase 2									
Phase 3									
Phase 4									
Phase 5									
Phase 6									

Im **ASCII**-Code (American Standard Code for Information Interchange) werden die Buchstaben des (amerikanischen) Alphabetes sowie Ziffern und Zeichen bestimmten Zahlen zugeordnet. Statt Dezimalzahlen verwendet man dazu häufig Hexadezimalzahlen (Sechszehner-System).

	dez.	hex.		dez.	hex.
@		40	´		60
A		41	**a**		61
B		42	**b**		62
C		43	**c**		63
D		44	**d**		64
E		45	**e**		65
F		46	**f**		66
G		47	**g**		67
H		48	**h**		68
I		49	**i**		69
J		4A	**j**		6A
K		4B	**k**		
L		4C	**l**		6C
M		4D	**m**		6D
N		4E	**n**		
O		4F	**o**		
P		50	**p**		70
Q		51	**q**		71
R		52	**r**		72
S		53	**s**		73
T		54	**t**		74
U		55	**u**		75
V		56	**v**		76
W		57	**w**		77
X		58	**x**		78
Y		59	**y**		79
Z		5A	**z**		7A
		5B			7B
		5C			7C
		5D			7D
		5E			7E
		5F			7F

Digital-Eingang

Das SIOS-Interface hat unter der Beschriftung „DIGITAL EINGANG” acht Buchsen. Jede dieser Buchsen kann über einen Taster mit Spannung (5V) verbunden werden. Jeder Eingang kann **low** oder **high** sein (Taster offen / geschlossen). Der Computer kann an jeder Buchse erkennen, in welchem dieser zwei Zustände sich der Taster befindet, nicht mehr. Deshalb heißen diese Eingänge



digital.

Mit acht digitalen Eingängen kann ein Computer bis zu _____ verschiedene Zustände erkennen. Mit einer **Dioden-Matrix** könnte man sogar erreichen, daß 256 Taster oder auch z.B. Lichtschranken oder Transistoren abgefragt werden könnten. Damit ließen sich z.B. 256 unterschiedliche Positionen eines Roboters erkennen.

Analog-Eingang

Der „ANALOG EINGANG” des SIOS-Interface kann im Grunde nicht mehr, und trotzdem kann man mit ihm viel leichter messen. An ihn kann man z.B. ein Poti (Potentiometer) anschließen. Weil man ein Poti stufenlos auf jede beliebige Zwischenstufe einstellen kann, ist es ein **analoges** Gerät. Über den analogen Eingang kann der Computer die Stellung des Potis erkennen. Da im Interface allerdings die analogen Signale in einem **Analog-Digital-Wandler** in 8 digitale Informationen umgewandelt werden (8 bit = 1 byte), ist es nur möglich, zwischen maximal 256 Winkeln zu unterscheiden.

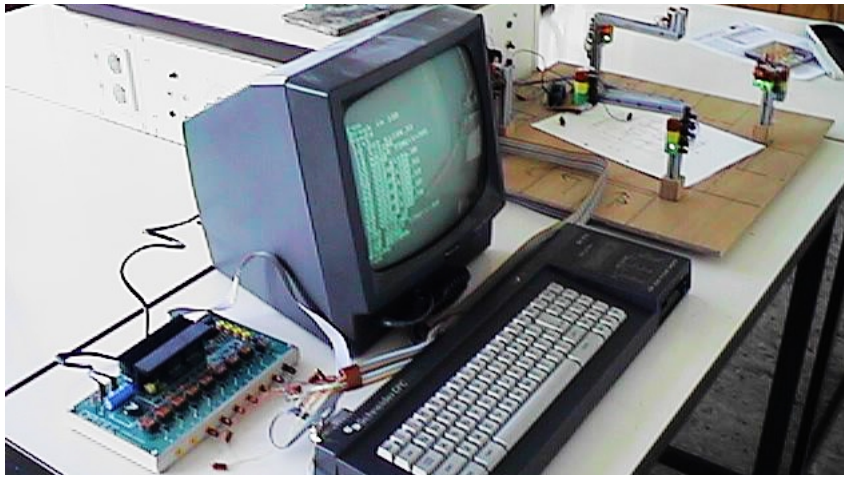
Statt eines Potis kann man auch andere analoge Bauteile bzw. Geräte an den analogen Eingang anschließen. So kann man z.B. mit einem LDR die Helligkeit oder mit einem NTC Temperaturen messen. Allerdings immer nur max. ___ Volt mit einer **Auflösung** von ___ bit, was bedeutet, daß immer nur maximal 256 unterschiedliche Meßwerte erfasst werden können. **LDR** oder **NTC** müssen dabei Teil einer **Spannungsteiler**-Schaltung sein, damit sich mit ihrem Widerstand auch die anliegende Spannung ändert. Nur diese wird vom SIOS-Interface gemessen.

Anschlußplan

Mit der hier dargestellten Schaltung kann man die Helligkeit messen. Wenn es heller wird, nimmt der Widerstand des LDR *zu / ab*. Dadurch liegt an ihm *mehr / weniger* Spannung an. Die vom Interface ge-

Die Ampeln einer Straßenkreuzung werden an 6 Ausgänge eines Computer-Interfaces angeschlossen wie aus der nebenstehenden Tabelle (und dem Foto) zu ersehen ist. In Phase 1 soll Straße A rot haben (bit 5 = I) und Straße B grün (bit 0 = I). bit 5 hat den Wert 32, bit 0 hat den Wert 1. Das macht zusammen den Dezimalwert 33. Ergänze die Tabelle!

	-	-	Straße A			Straße B				
	-	-	rot	gelb	grün	rot	gelb	grün		
bit	7	6	5	4	3	2	1	0		
Wert	128	64	32	16	8	4	2	1	dezimal	hexadezimal
Phase 1	O	O	I	O	O	O	O	I	**33**	**&21**
Phase 2	O	O	I	I	O	O	I	O	**50**	**&32**
Phase 3	O	O	I	I	O	I	O	O		
Phase 4										**&0C**
Phase 5										
Phase 6										



BASIC ist eine Programmiersprache, die für Anfänger entwickelt wurde und deshalb in „Anfänger-Computer" wie den Schneider CPC oder den Commodore C64 fest eingebaut wurde. Deshalb kann man nach dem Einschalten des Computers sofort mit dem Programmieren beginnen. Schreibe folgendes Ampel-Programm:

10 OUT 61184,33

20 OUT 61184,50

30 OUT 61184, (ergänzen!)

40 OUT 61184, (ergänzen!)

50 OUT 61184, (ergänzen!)

60 OUT 61184, (ergänzen!)

Weil in diesem (alten) BASIC-Dialekt jede Programmzeile nummeriert sein muß, beginnt jede Zeile mit einer Zahl. Damit man später noch Zeilen einfügen kann, nummeriert man meist nicht: 1,2,3,... sondern eher wie oben in Zehnerschritten: 10, 20, 30, ...

Die erste Zahl hinter **OUT** ist die Adresse des Interface. Die zweite Zahl ist der Befehl. Die Befehle kannst du mit der Tabelle rechts oben berechnen.

Damit sich das Ganze auch wiederholt, schreibe noch:

70 GOTO 10

Um das Programm zu starten, tippe ein: **RUN**

Vermutlich wird dir das Programm viel zu schnell ablaufen. Durch zweimaliges Drücken der ESC-Taste kannst du das Programm unterbrechen. Füge dann nach jedem Befehl eine Warteschleife ein. Tippe z.B. folgendes ein:

12 t = time

14 while time<t+3000

16 wend

Diese Zeile werden an der richtigen Stelle eingefügt, wenn du den Befehl **LIST** eingibst.

Diese Befehlsfolge sorgt dafür, daß der Computer 10 Sekunden wartet, bevor er den nächsten Befehl (in Zeile 20) ausführt.

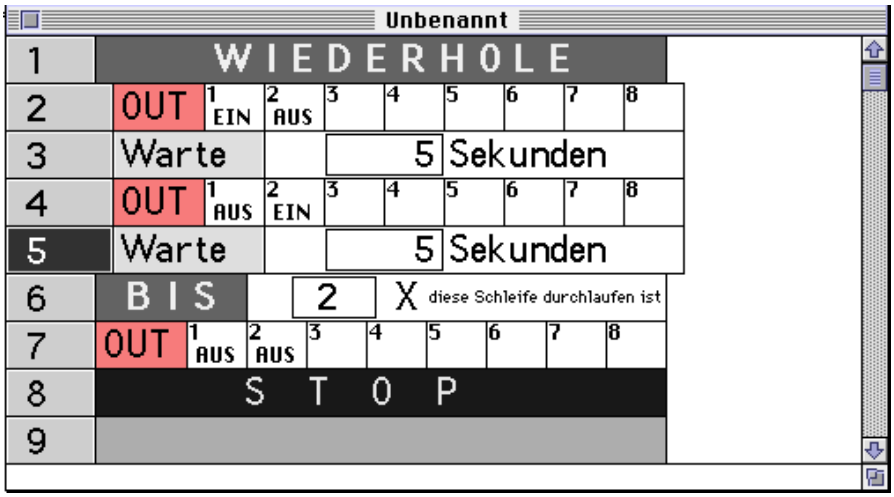
Der Schneider CPC mißt die Zeit in 300stel-Sekunden-Abständen. Versuche durch Einfügen weiterer Warteschleifen ein sinnvolles Programm zur Steuerung einer Ampelkreuzung zu schreiben. Überlege welche Phasen länger oder kürzer dauern sollten.

Neben acht Ausgängen stellt das Interface vier Eingänge des Joystick-Anschlusses bereit. Füge z.B. folgende Zeile ein:

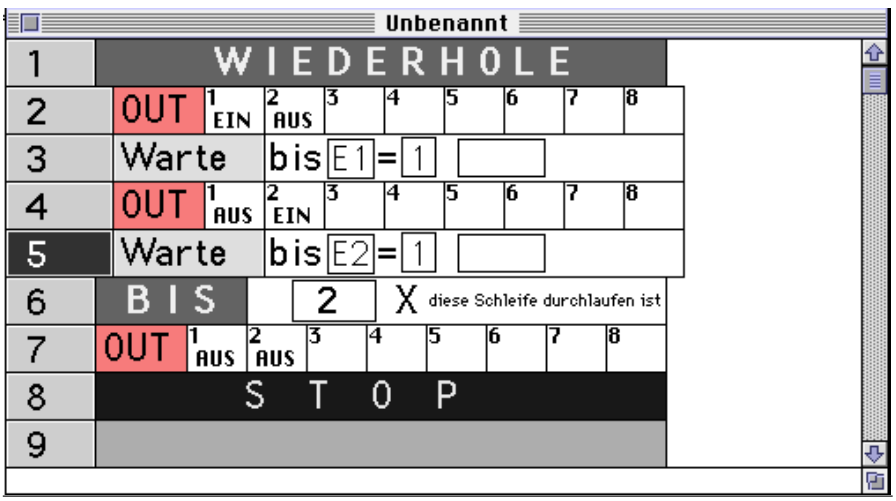
69 IF JOY(0)=1 THEN END

Kannst du ein Programm für eine Fußgängerampel schreiben?

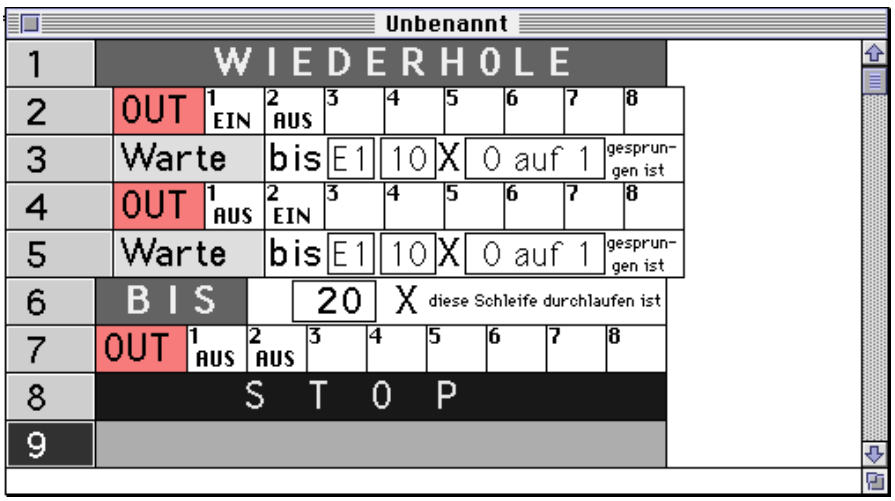
Das Programm „RoboticPerformer”, das du hier kennenlernen wirst, hat ein Schüler speziell zum Steuern von Geräten geschrieben. Es ist ausschliesslich mit der Maus (kinderleicht?) zu bedienen. Nachdem du das Programm gestartet hast („Modulbus” anklicken!), teste zunächst einmal die Möglichkeiten des Programmes. Danach sollst du etwas über die Steuerung von Motoren lernen.



Schliesse für das erste Experiment den Motor der „Laufkatze” an Klemme 0 und 1 des Digital-Ausganges des SIOS-Interfaces an. (Im Programm sind diese Ausgänge mit 1 und 2 bezeichnet.) Der Motor kann jetzt vorwärts und rückwärts laufen. Lass ihn mit einem Programm wie dem nebenstehenden einige Male dieselbe Zeit vor und zurück laufen. Kommt die Laufkatze wieder an derselben Stelle an, wo sie gestartet ist?



Schließe jetzt den einen Taster an der Laufschiene an die Klemme 0 des Digital-Einganges des Interfaces und an 5V an, den anderen an 1 und 5V. Mit nebenstehendem Programm wird die Laufkatze, obwohl sie nicht immer gleich schnell läuft, immer gleich weit fahren. Eine solche Rückmeldung durch Taster oder auch Lichtschranken ist beim Steuern von Motoren oft recht nützlich.



Wie sich mit nur einem Taster (oder einer Lichtschranke) Bewegungen sehr genau kontrollieren lassen, kannst du mit nebenstehendem Programm sehen, wenn du statt der Laufkatze den Motor mit der Drehscheibe und dem zugehörigen Taster richtig an das Interface anschliesst. So lässt sich z.B. ein Kran an verschiedene Plätze führen, ohne daß jeder Platz durch Taster kontrolliert wird.

Während die Bewegungen im ersten Experiment zeit**gesteuert** abliefen, werden sie in den anderen Fällen durch Rückmeldungen von z.B. Tastern oder Lichtschranken **geregelt**.

Mac + SIOS-Interface + Aufzug-Modell + Software: „Aufzug" + Technik-Buch S. 100 f.

Schritt-Motor

In Schrittmotoren dreht sich ein Dauermagnet (**Rotor**) im Feld zweier Elektromagneten. Wird einer der E-Magneten umgepolt dreht sich der Rotor um genau 90°. Je nachdem welcher der beiden E-Magneten umgepolt wird, dreht sich der Motor nach links oder rechts.

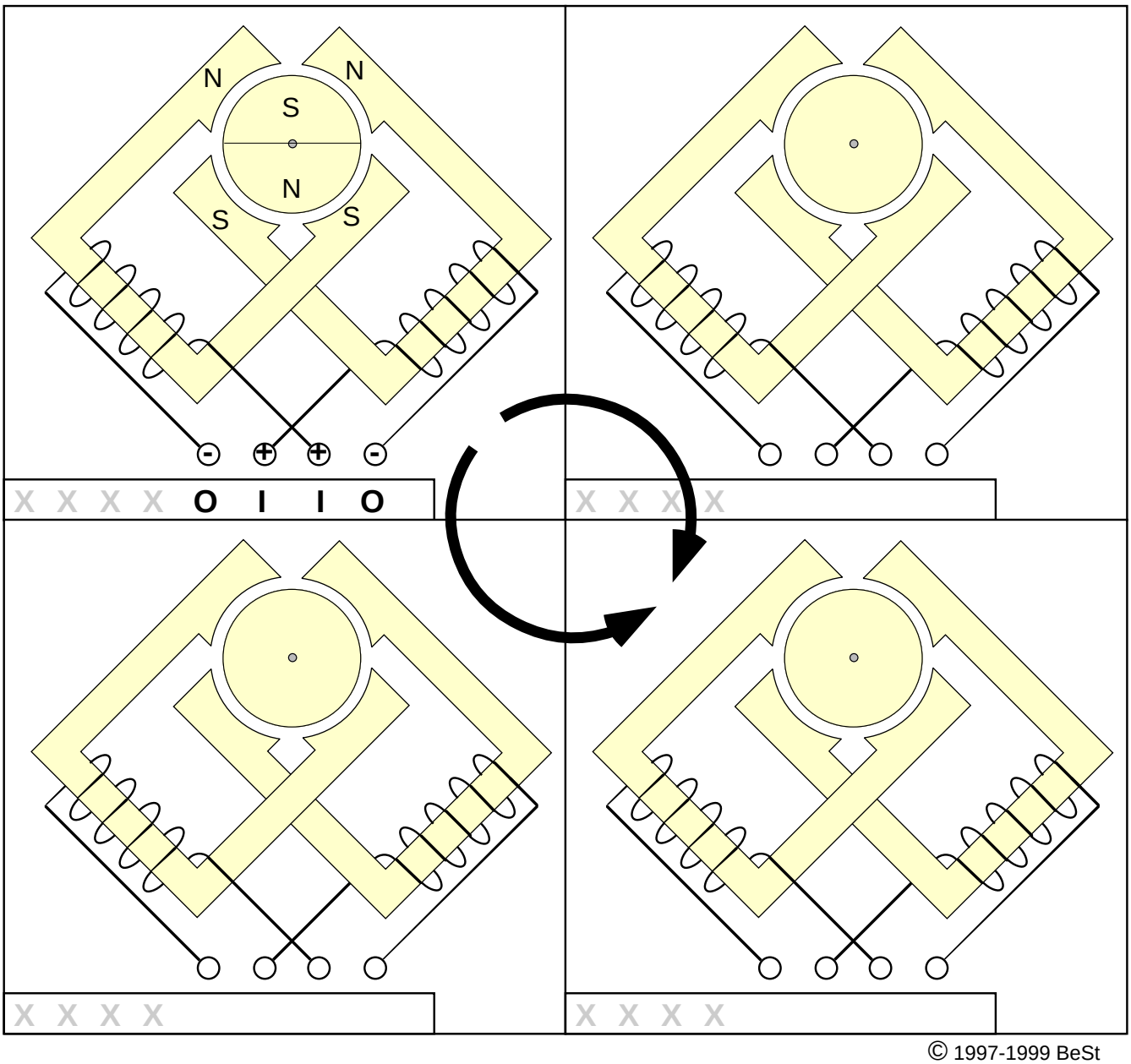
Die vier Skizzen zeigen einen **bipolaren** Schrittmotor. Die erste Skizze ist vollständig. Färbe hier zunächst alle Pole:

In der Skizze rechts daneben soll sich der Rotor gegenüber der linken Skizze um 90° nach rechts gedreht haben.

Beschrifte und färbe den Rotor entsprechend. Überlege dann, welcher der beiden E-Magnete umgepolt werden muß, damit sich der Rotor tatsächlich in diese Stellung bewegt. Färbe und beschrifte die E-Magneten und die Spannungsquellen entsprechend.

Ergänze ebenso die beiden anderen Skizzen.

Weil sich Schrittmotoren sehr präzise steuern lassen, werden sie z.B. in Floppy-Laufwerken, Druckern und Plottern eingesetzt. Für ihre Ansteuerung müssen immer vier bits bzw. Ausgänge geschaltet werden. Wenn du weißt, daß hier „I" für Plus steht und „O" für Minus, kannst du sicher auch die drei fehlenden Befehle ergänzen.



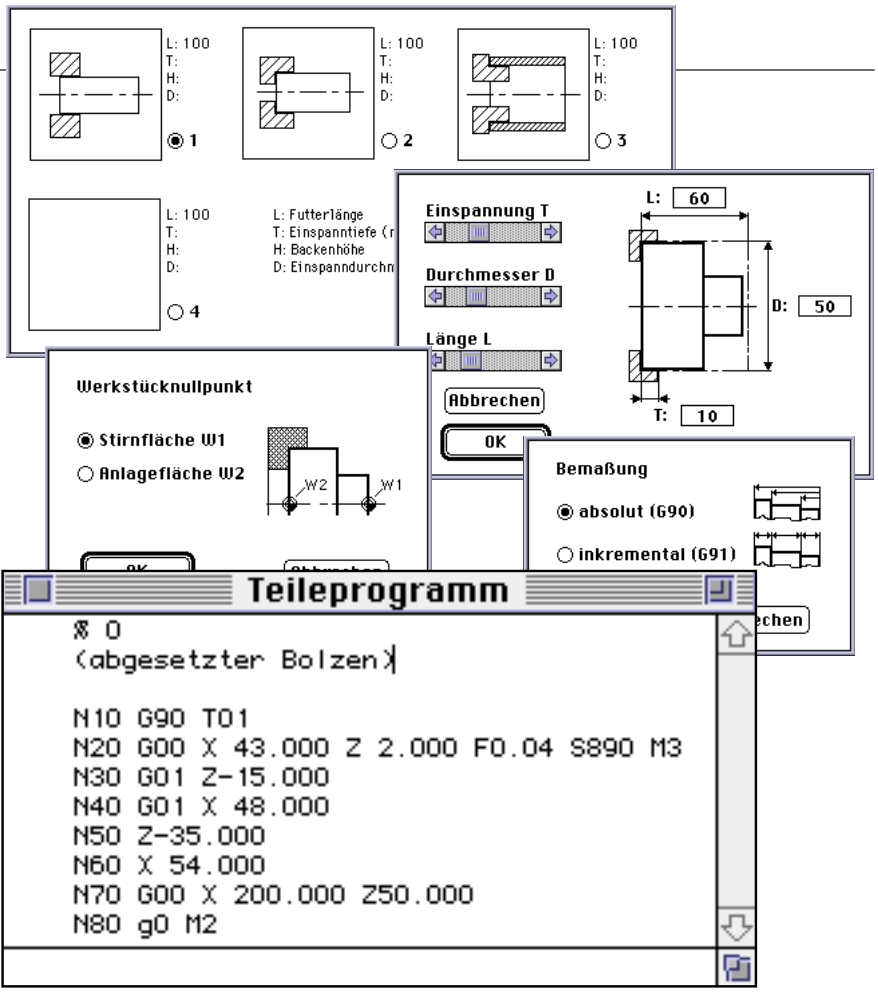
Mac + Software: „CNCSim" und „CNCSim Intro" + Technik-Buch S. 104 f. **CNC**

Moderne Werkzeugmaschinen wie z.B. [redacted]-, [redacted]- und [redacted] arbeiten häufig mit einer [redacted] (**C**omputer**N**umerical**C**ontrol). [redacted] bewegen Werkzeug und Werkstück nach einem eingegebenen Programm auf 3 [redacted], wobei ein [redacted] ständig die Bewegung dieser Schlitten kontrolliert und gegebenenfalls [redacted].

Mit dem Programm CNCSIM kannst du die Programmierung einer Drehmaschine simulieren. Öffne dazu das Programm und rufe danach zuerst aus dem Menü „Rüsten" nacheinander die Befehle „Spannmittel ...", „Rohteil ...", „Werkstücknullpunkt ..." und „Bemaßung ..." auf. Vergleiche die erscheinenden Dialogfelder mit den nebenstehenden Abbildungen und passe sie gegebenenfalls an die Abbildungen an.

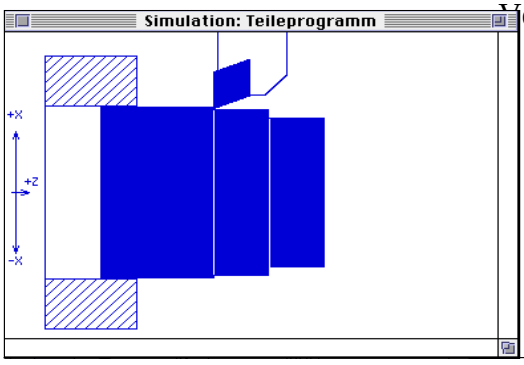
Wähle dann aus dem Menü „Ablage" den Befehl „Neu" und gib im Fenster „Teileprogramm" das nebenstehende Programm ein.

Danach kannst du mit dem Befehl „Start" aus dem Menü „Simulation" das Programm in einer Simulation ausführen lassen. Wenn du keinen Fehler gemacht hast, wirst du etwa folgendes sehen:



Ein CNC-Programm besteht im Wesentlichen aus **Zeilennummern** und **Sätzen**. Ein Satz enthält meist mehrere **Befehle**. Jeder Befehl besteht aus einem Buchstaben gefolgt von einer Zahl.

Versuche herauszufinden, was z.B. folgendes bedeutet:



N10: _____

T01: _____

X43: _____

G01: _____

Satz N60: _____

Satz N70: _____

Abschlußarbeit zum Thema „Regeln”:

Statt für die Abschlußarbeit eine elektronische Schaltung anzufertigen, kannst du ein Computer-Programm schreiben, das ein Gerät bzw. einen Vorgang regelt. Dazu müssen Aus- und Eingänge des Computers genutzt werden.

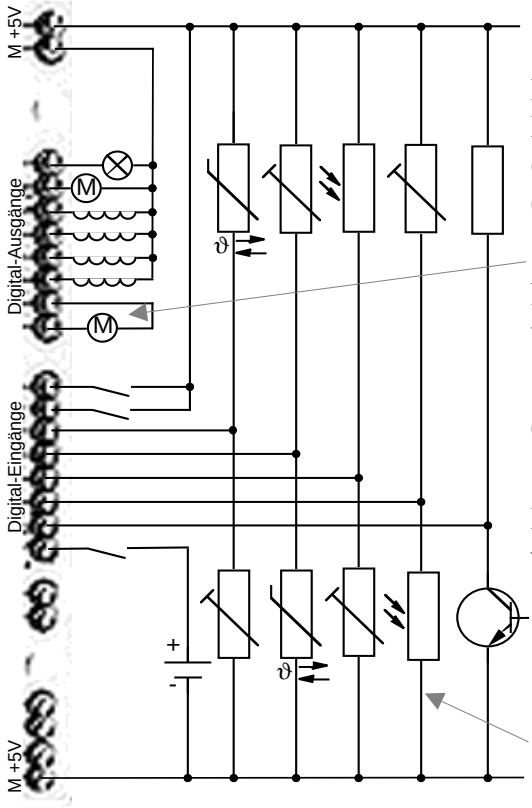
Fertige dazu ein Modell, das die Anwendung des Programmes veranschaulicht. Noch besser ist, wenn dieses Modell eine selbst gefertigte elektronische Schaltung beinhaltet.

Wichtig ist, daß die Arbeit gut dokumentiert und präsentiert wird. (Text, Zeichnungen, Fotos, Videos, Animationen sind auch als Teil des Computer-Programmes möglich!) Programm (und Schaltung) müssen verstanden und erklärbar sein.

Überlege auch, ob der Einsatz eines Computers für diesen Zweck sinnvoll ist. Könnte man das Problem statt dessen mit einer elektronischen Schaltung lösen?

Je nachdem ob du den Schwerpunkt mehr auf das Programmieren, das Modell oder die Präsentation legen willst, kannst du dein Programm mit Hilfe von Maschinencode, BASIC, RoboticPerformer, DoIt, MacIt und HyperScript oder eventuell auch mit RagTime oder CNCSim erstellen.

Die in der rechten Spalte aufgeführten Themen sollen Anregungen für ein eigenes Thema geben. Besprich deine Idee(n) wiederholt mit dem Lehrer, um zu klären, was machbar ist, welche Schwierigkeiten auf dich zu kommen, ob die Lösung des Problems für die ge-



- Mit **einem Ausgang** des SIOS-Interface (und der Masse) kann man z.B. eine Lampe oder einen anderen elektrischen Verbraucher ein- und ausschalten.
- Mit **zwei Ausgängen** kann man einen Motor links und rechts drehen lassen.
- Mit **vier Ausgängen** (und der Masse) kann man einen Schrittmotor betreiben.

Ein **Eingang** kann HIGH werden z.B. durch:
- Anschluß an +5V (>+2V)
- Hitze
- Kälte
- Licht
- Dunkelheit
- Sperren eines Transistors

Gewächshaus:
- Lüftung
- Heizung
- Bewässerung
- Beschattung / Beleuchtung

Haus:
- Einbruch-Sicherung
- autom. Sicherheitstüre
- Tressor mit Code-Schloß

Aufzug:
- mehrere Stockwerke
- Stockwerksanzeige
- Speichern der Wünsche

Treppenhaus-Licht:
- Verlängerung bei Wiederw.
- Spar-Schaltung

Wellenreiten:
- Punkte-Wertung
- Verlängerung der Spielzeit
- Schwierigkeit veränderbar

Radarfalle
Wind- / Drehzahlmesser
Strichcode-Leser
Fernbedienung

Motorboot in Richtung halten
Segelboot im Wind steuern
Fahrzeug automatisch fahren
Antiblockier-System
Auto-Abstandhalter

Verkehrsampel:
- Fußgänger-Ruftaste
- verkehrsabhängige Regelung
- Tag- und Nacht-Schaltung
- Grünschaltung für Busse

Parkhaus-Schranke
Parkhaus-Belegung

Blockstrecken-Sicherung
Bahnübergang
Ablaufberg

Zugbrücke
Mausefalle
Fischfütterung
Waschmaschine
Abfüllanlage
Auto-Waschanlage
Roboter
Plotter
Hochregal
Flipper

Ein HyperCard-Stapel besteht aus (einer oder) mehreren **Karten** derselben Größe, von denen immer eine zu sehen ist. Für die aktive Arbeit mit HyperCard können auch mehrere **Stapel** gleichzeitig geöffnet sein.

Jede Karte hat einen **Hintergrund** und einen durchsichtigen **Vordergrund**. Mehrere (oder alle) Karten eines Stapels können denselben Hintergrund benutzen.

Hintergrund und Vordergrund können z.B. Texte, Zeichnungen, Fotos, Movies, Formulare, Kalendarien, Adressfelder und vieles andere enthalten.

Eine Spezialität von HyperCard sind die **Buttons**, wie sie heute auch im Internet üblich geworden sind. Mit ihnen kann man per Mausklick fast beliebige Aktionen auslösen. Häufig beinhalten sie **Links** (Verbindungen), die einen zu anderen Karten führen. Genauso gut kann man mit ihnen aber auch optische und akustische Effekte auslösen.

Solche Buttons kann man entweder aus vorhandenen Karten kopieren (und evtl. verändern) oder man kann sie neu erzeugen und per Mausauswahl oder per Script programmieren.

Grundsätzlich lassen sich alle Medien wie Schrift, Sprache, Geräusche, Musik, Grafiken, Fotos, Filme und Animationen in HyperCard verwenden.

Eine Besonderheit von HyperCard ist, daß es jede Änderung automatisch sichert. Was solange von Vorteil ist, solange man etwas verbessert. Wenn man allerdings Fehler einbaut, ist die alte bessere Version verloren. Deshalb empfiehlt es sich, die Originalversion eines wichtigen Stapels auf zwei Speichermedien zu sichern und bei größeren Arbeiten gelegentlich eine Kopie zu sichern.

Mit der **Script**-Sprache von HyperCard lassen sich eigene Programme schreiben. Sofort nach jeder Änderung kann das veränderte Programm gestartet bzw. getestet werden.

Da solche Programme auf einem anderen Programm (HyperCard) aufsitzen, ist ihre Schnelligkeit nicht berauschend. Was sich bei langsamen Rechnern bemerkbar macht. Schneller laufen Programme die z.B. mit **BASIC**, **Pascal**, **C** oder ähnlichen Programmiersprachen geschrieben sind. Programmteile (z.B. auch des Betriebssystemes), bei denen es besonders auf Schnelligkeit ankommt, werden in **Maschinencode** geschrieben.