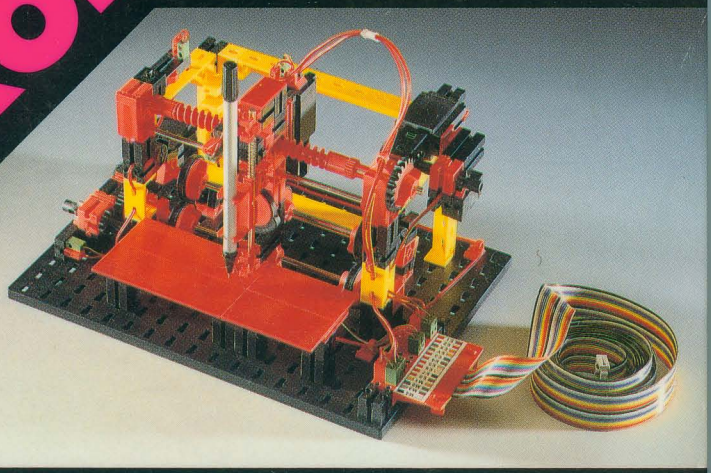
CHIP INSIDE

34,– DM

fischertechnik PROFI COMPUTING

Modellsteuerung mit dem Computer

PROFI COMPUTING



- Messen, Steuern, Regeln mit dem Computer
- Realitätstechnik erleben mit fischertechnik
- Programmieren ohne zu programmieren mit fischertechnik Software LUCKY LOGIC
- fischertechnik PROFI SENSORIC – Einsteigen in die Welt der Sensoren
- Experimentieren mit Fuzzy Logic

Im Heft
auf MS-DOS-Diskette:
Programmbeispiele

Eine Publikation von CHIP SPECIAL
in Zusammenarbeit mit fischerwerke

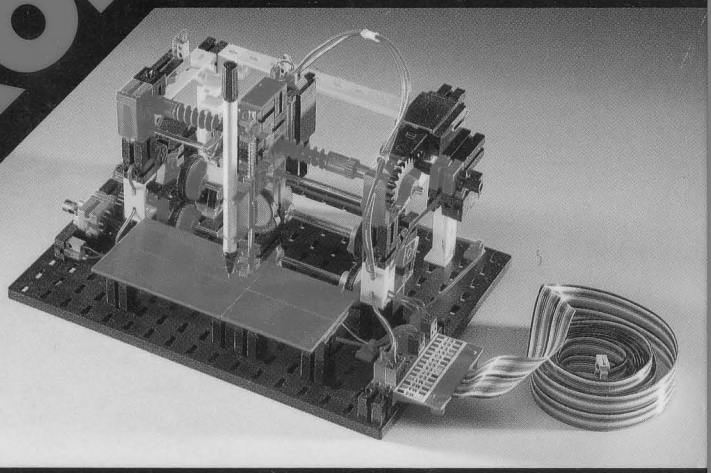
CHIP INSIDE

34,– DM

PROFI COMPUTING

fischertechnik PROFI COMPUTING

Modellsteuerung mit dem Computer



- Messen, Steuern, Regeln mit dem Computer
- Realitätstechnik erleben mit fischertechnik
- Programmieren ohne zu programmieren mit fischertechnik Software LUCKY LOGIC
- fischertechnik PROFI SENSORIC – Einsteigen in die Welt der Sensoren
- Experimentieren mit Fuzzy Logic

Im Heft auf MS-DOS-Diskette: Programmbeispiele

Eine Publikation von CHIP SPECIAL in Zusammenarbeit mit fischerwerke

Automatisierung ist Teil unserer Welt

Ob wir es nun wahrhaben wollen oder nicht: Die Automation bestimmt zunehmend in allen Bereichen unser Leben. Ob wir eine Fahrkarte kaufen, in eine Parkgarage einfahren, eine CD in unseren Player einlegen oder den letzten Kontostand abfragen, immer und überall treffen wir auf automatisierte Vorgänge. Ebenso geht es uns am Arbeitsplatz. Auch dort ist der Automat unser ständiger Begleiter. Diese Entwicklung kann man als gegeben hinnehmen und sich mit den Ergebnissen der Automation zufrieden geben. Man kann sich aber auch intensiv mit diesem Thema auseinandersetzen. Dann sollte man wissen, wie alles funktioniert. Wie wird gemessen? Was ist der Unterschied zwischen Steuern und Regeln? Wie funktioniert das alles?

Es ist gerade fünf Jahre her, daß unser CHIP-SPECIAL "Fischer-Technik und Computer" erschienen ist. Der Vergleich der Themen beider Hefte zeigt, wie sich die Schwerpunkte verschoben haben. Einen großen Teil des ersten Heftes nahmen Programmlistings und speziell gefertigte Platinen bzw. deren Bestückung ein. Heute wird auch noch programmiert. Aber Platinen fertigen wohl nur noch Spezialisten an. Und programmieren kann man die fischertechnik-Projekte inzwischen mit Lucky-Logic. Alles ist einfacher geworden. Aber gleichzeitig ist das vorauszusetzende Wissen immer umfangreicher geworden. Diesen gestiegenen Anforderungen trägt unser neues Heft Rechnung. Das Heft vermittelt die für das Verständnis der Automation und Robotik erforderlichen Grundlagen.

Der Themenkreis ist sehr weit gespannt. Von der geschichtlichen Entwicklung über die Theorie des "Messen, Steuern, Regeln"-Komplexes bis hin zu dem immer aktueller werdenden Thema "Fuzzy-Logic" werden alle Bereiche dargestellt, die man kennen muß, um die Thematik zu verstehen und wenn man mitreden will.

Mit Hilfe der fischertechnik-Kontruktionsbaukästen Profi-Sensoric und Profi-Computing wird dann dieses Wissen an wirklichkeitsnahen Modellen angewendet.

Zum Schluß dieses Editorials wollen wir eine Bitte an unsere Leser richten: Es war nicht Ziel dieses Heftes, die Thematik der Automatisierung und Robotik von ihrer gesellschaftskritischen Seite her zu untersuchen.

Aber diese Seite ist wichtig. Uns würde deshalb interessieren, wie Sie zu diesem Thema stehen. Was halten Sie von der Entwicklung? Wie stehen Sie zum Fortschritt von Wissenschaft und Technik? Bitte schreiben Sie uns. Verwenden Sie dazu das Stichwort "Quo vadis, Roboter?", denn das ist für uns alle sicher eine wichtige Frage: Wohin führt Dich Dein Weg, Roboter?

Viel Spaß mit diesem CHIP INSIDE wünscht Ihnen

Ihre Redaktion CHIP SPECIAL

40

A. Schwarz

Armin Schwarz
(Chefredakteur)

H. Haupt

Horst F. Haupt
(Autor)

Impressum

fischertechnik
PROFI COMPUTING
1. Auflage 1992
Best.-Nr. 1258
ISBN 3-8023-1258-9

Die Deutsche Bibliothek – CIP-Einheitsaufnahme
Haupt, Horst F.:
Fischertechnik Profi computing: Modellsteuerung mit dem Computer / [Autor: Horst F. Haupt]. –
1. Aufl. – Würzburg: Vogel, 1992
(Chip inside)
ISBN 3-8023-1258-9
NE: HST

CHIP INSIDE

Anschrift der Redaktion:
Schillerstr. 23a, 8000 München 2, Tel. (089) 51493-0,
Teletex 897190, Telex 17-897190,
Telefax (089) 535000

Chefredakteur: Armin Schwarz
(verantwortlich für den Inhalt)

Stellvertretender Chefredakteur: Ulrich Kern

Redaktion: Ulrich Kern, Jörg P. Jordan,
Armin Schwarz

Redaktionssekretariat: Petra Eibicht

Leserservice: Petra Eibicht

Autor dieser Ausgabe: Horst F. Haupt

Titelgestaltung: Hans Kuh

DTP-Layout: EDV-Beratung Brand

Verlag: Vogel-Verlag und Druck KG, Postfach 6740,
D-8700 Würzburg 1, Tel. (0931) 418-0,
Telex 68883, Telefax (0931) 44053.
Telegramme: CHIP-Würzburg

Verlagsdirektor: Dr. Andreas Kaiser

Anzeigenleiter: Friedrich Mangold, Würzburg
(verantwortlich für Anzeigen)

Anzeigenverkauf: Elke Saller, STYX
Computer-Vertrieb, Friedenstr. 9, 8011 Aschheim,
Tel.: (089) 9030640

Anzeigenservice: CHIP SPECIAL, Postfach 6740,
8700 Würzburg 1, Tel. (0931) 418-0,
Telex 68883, Franz Fenn,
Durchwahl 418-2350.

Vertrieb: Axel Herbschleb (Leitung), Alheidis Moers
(Stellvertretung, Auslieferung), Durchwahl der CHIP
SPECIAL-Auslieferung:
Würzburg (0931) 4182074

Vertrieb Handelsauflage: Vereinigte Motor-
Verlage GmbH & Co. KG, Leuschnerstr. 1,
D-7000 Stuttgart 1, Tel. (0711) 182-1548

Bezugsmöglichkeiten: Bestellungen nehmen der Verlag und alle Buchhandlungen im In- und Ausland entgegen. Sollte die Zeitschrift aus Gründen, die nicht vom Verlag zu vertreten sind, nicht geliefert werden können, besteht kein Anspruch auf Nachlieferung oder Erstattung vorausbezahlter Bezugsgelder.

Bankverbindungen Vogel-Verlag:
Dresdner Bank AG, Würzburg
(BLZ 79080052) 3148890000,
Bay. Vereinsbank AG, Würzburg
(BLZ 79020076) 2506173,
Kreissparkasse Würzburg
(BLZ 79050130) 17400,
Postgirokonto Nürnberg
(BLZ 76010085) 9991-853
Ausland: Postscheckkonto Zürich
8047064,
Niederlande 2662395
Crédit Industriel d' Alsace et de Lorraine, Strasbourg,
Kto.-Nr. 101/01 212576

Herstellung: Elfi-Teresa Wagner

Druck: Alois Erdl KG, Trostberg

VOGEL

Inhalt

Die Welt der Roboter

Es ist noch gar nicht so lange her, da wurden die ersten Arbeitsroboter als Weltsensation bestaunt und bewundert. Endlich hatte man die Möglichkeit gefunden, dem Menschen die schwere und die gefährliche Arbeit abzunehmen. Inzwischen gehören die "fleißigen Helfer" zu unserem Arbeitsalltag. In diesem Beitrag beschäftigen wir uns damit, was Roboter sind, wie sie zu dem wurden, was sie darstellen und spekulieren darüber, wohin der Weg wohl führt.

Für die Menschen aller Zeiten war es immer ein Traum: eherne, aus Eisen oder gar Gold geformte Geschöpfe in menschlicher Gestalt, die, ohne Schlaf oder Nahrung zu brauchen, ständig die ihnen übertragenen Aufgaben erfüllen.

Bei den alten Griechen war es der Gott Hephaistos, der Beherrscher des Feuers, der Mädchen aus Gold schmiedete, die sich bewegen, die sprechen, ja sogar denken konnten. Im Mittelalter schuf Rabbi Loew nach der Legende in Prag den Golem aus Lehm. Von Albertus von Lauingen, den man heute Albertus Magnus, Albert den Großen nennt, wird berichtet: "Vor das Schloß schleppten Diener ... ein hölzernes Pferd, das Albert in aller Heimlichkeit konstruiert hatte. ... Er saß auf und machte sich an einem Hebel zu schaffen. Sogleich begann das Pferd zu gehen und alsbald sich mit wundersamer Geschwindigkeit zu bewegen, wobei sein Reiter es wie ein lebendiges Roß aus Fleisch und Blut lenkte. ... schrie das Volk, dies sei Zauberei." (Dominique Webb in: Das Geheimniwissen des Albertus Magnus, Ariston Verlag Genf/München).

Leider sind uns, vielleicht deshalb, weil man diese ersten Roboter für Teufelswerk hielt, die meisten davon nicht erhalten. Noch heute bewundern kann man aber zum Beispiel den schreibenden Automaten des Pierre Jaquet-Droz (im Musee d'Art et d'Histoire in Neufchatel). Ein kleiner Junge sitzt vor einem Blatt Papier, tunkt immer wieder den Federkiel in die Tinte und schreibt mit exakter, feiner Schrift, ein Wunder der frühen Feinmechanik. Und heute? Nach dem Unglück von Tschernobyl setzte man, nachdem zunächst Menschen unter der Verkennung der Gefahr in die tödlichen Strahlen geschickt wurden, Roboter ein, den Schutt und Schrott zu entfernen. Hier hatten diese Maschinen einen Einsatzzweck gefunden, der dem Menschen wirklich half und weitere Menschen vor Schaden bewahrte.

Was ist ein Roboter?

Nach dem Duden Informatik ist das Wort Roboter - "(von tschech. robota = Frondienst)" - die "Umgangssprachliche Bezeichnung für flexible Handhabungsgeräte. Oft sind Roboter programmgesteuerte Maschinen, deren Arbeitsweise der eines Menschen nachgebildet ist und die komplexe manuelle Tätigkeiten eines Menschen ausführen können." Hier ist also die Nachbildung menschlicher Arbeitsweisen ein wesentliches Merkmal eines Roboters.

In Duden Deutsches Universal Wörterbuch steht unter dem Stichwort Roboter: "[nach dem engl. Titel 'Rossum's Universal Robots' des 1920 erschienenen sozialutopischen Dramas des tschechischen Schriftstellers K.Capek (1890-1938)]: (Der menschlichen Gestalt nachgebildeter) Automat mit beweglichen Gliedern, der bestimmte mechanische Tätigkeiten verrichtet; Maschinenmensch".

Hier kommt etwas Wichtiges zum Bild des Roboters dazu: Die Nachbildung des Menschen. Schon in den Legenden des Altertums und des Mittelalters gab es Automaten als Nachbildungen menschlicher Gestalten. Sie agierten und reagierten wie Menschen. Der Mensch als Schöpfer dieser Gestalten war plötzlich gottgleich, denn er schuf Gestalten "sich zum Bilde". Und waren nicht auch die Menschen von Gott geschaffene mechanisch ablaufende "Automaten"?

In Rembrandts Bild "Anatomie des Dr. Vulpius" wird die Sezierung eines menschlichen Körpers gezeigt. Und dort auf dem Tisch liegt kein Mensch mehr sondern ein in seine einzelnen Elemente zerlegbarer Mechanismus. Etwas Wesentliches, das was den Menschen vom Automaten unterscheidet, ist nicht mehr da: Die Seele, der Geist.

Und so sehen wir den Roboter auch in den Legenden unserer Zeit, in den Science-Fiction-Stories der Gegenwart. Die winzig kleine oder übergroße Gestalt, die wie ein Mensch aussieht, die reden, denken, handeln kann wie ein Mensch, die ihren Freunden hilft und die die Feinde ihrer Freunde bekämpft, das ist hier der Roboter. Er hat zwar oft einen höheren IQ als die Menschen, die ihn geschaffen haben, aber zum Glück ist ihm das erste Asimovsche Robotergesetz (siehe unten) eingepflanzt, das da lautet: "Töte nie einen Menschen". Auch wenn in den Stories manchmal die Roboter den Aufstand proben, hier liegt zum Glück für ihre "Mitmenschen" die absolute Grenze ihres Handelns. So viel zu den Legenden.

Die drei Gesetze der Robotik

Im Frühjahr 1942 wurden erstmals in einer Science-Fiction-Geschichte unter dem Titel "Runaround" Isaac Asimovs heute sogenannten drei Gesetze der Robotik beschrieben. Diese drei Gesetze lauten:

1. Ein Roboter darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen.

2. Ein Roboter muß den Befehlen eines Menschen gehorchen, es sei denn, solche Befehle stehen im Widerspruch zum ersten Gesetz.

3. Ein Roboter muß seine eigene Existenz schützen, solange dieser Schutz nicht dem ersten oder zweiten Gesetz widerspricht.

Diesen drei Gesetzen sollte jeder Roboter unterliegen. Es ist zu hoffen, daß alle künftigen Entwickler von "nützlichen Helfern" diese drei Gesetze berücksichtigen.

Zu den von uns schon aufgeführten Definitionen des Roboterbegriffs soll noch eine dritte Quelle kommen. Im Brockhaus Naturwissenschaften und Technik (1989) fanden wir folgende Beschreibung:

"Roboter, 1921 von K.Capek geschaffene Bezeichnung für einen künstlichen Menschen, eine Puppe, die Bewegungen scheinbar selbständig ausführt, zum Beispiel auf Grund drahtlos übermittelter Befehle. Im allgemeinen Sprachgebrauch wird unter Roboter eine menschenähnliche (antropomorphe) Maschine verstanden, die dem Aussehen des Menschen nachgebildet ist oder Funktionen, die der Mensch ausführt, zumindest teilweise übernehmen kann. Besonderes Kennzeichen gegenüber einem Automaten ist die Lernfähigkeit. In der Forschung gibt es viele Ansätze, Geräte mit künstlicher Intelligenz zu bauen. Sie können über Sensoren, so etwa Fersehkameras, mit ihrer Umgebung in Wechselwirkung treten und wiederholen einmal gemachte Fehler nicht. ... In der Entwicklung von Robotern stellen die seit Mitte der 60er Jahre einsatzfähigen Industrieroboter die erste Generation dar. ... Roboter der zweiten Generation besitzen Sensoren für Tast- und Sehfunktionen. Roboter im Sinn des eingangs gekennzeichneten Sprachgebrauchs könnten die dritte Generation werden."

So weit das Zitat. Auch hier wird also auf die Menschenähnlichkeit als ein wesentliches Merkmal der Roboter im weiteren Sinne Bezug genommen. Die Lernfähigkeit ist der wichtige Unterschied des Roboters zum Automaten. Während Automaten nach einem zwar veränderbaren Programm, aber gesteuert von diesem dann in immer gleicher Weise ablaufen, verändern Roboter aufgrund ihrer Lernfähigkeit den Ablauf ihrer Aktionen selbständig. Und es kommt noch ein neuer Begriff ins Spiel: die künstliche Intelligenz. Damit wären wir dann beim Computer. Aber lassen Sie uns zu diesem Thema später kommen.

Wir wollen hier versuchen, die Definition des Roboterbegriffs neu zu fassen: Ein Roboter ist eine Maschine, die von lernfähigen Programmen gesteuert menschliche Handlungsweisen nachbilden kann.

Die Gestalt des Roboters hat nach dieser Definition keinerlei Bedeutung. So können wir uns von dem Schreckensbild der Ungetüme in Menschengestalt lösen und dem Roboter ohne Ressentiment und vor allem ohne Unterlegenheitsgefühl nähern.

Der Automat oder der Ursprung des Roboters

Im folgenden Abschnitt werden wir uns mit einigen Stationen der geschichtlichen Entwicklung von Wissenschaft und Technik befassen, die Voraussetzung für die Entwicklung des Roboters waren, so wie wir diese Spezies heute kennen. Dabei geht es uns darum aufzuzeigen, daß Entwicklungen immer aufeinander aufbauen. Nur wenige menschliche Schöpfungen hatten keine Vorgänger. Der Abriß soll kein Geschichtsunterricht werden. Aus diesem Grund wollen wir auch die Jahreszahlen auf das erforderliche Minimum reduzieren. Aber der Blick in die Geschichte ist erforderlich, den Ursprung der Automation und Robotik zu verstehen.

Der Blick in die Geschichte der technischen und wissenschaftlichen Entwicklung der Menschheit zeigt, daß kein Volk dieser Welt einen Grund hat, auf andere überheblich herabzusehen. Viele Menschen zahlreicher Völker haben zur Gesamtentwicklung beigetragen. Wir wollen hier einmal davon absehen, die zwar interessanten, aber durch nichts beweisbaren Ideen des Besuchs der außerirdischen Götter mit in unsere Betrachtung einzubeziehen.

Nun kann man manche Entwicklung bedauern und darüber sinnieren, was gewesen wäre, wenn... Aber sehr hilfreich ist solch ein Ansatz sicher nicht. Besser erscheint uns, die Frage zu beantworten: "Was wird sein, wenn..?" Denn die Vergangenheit können wir nicht verändern. Aber wir sollten unsere Zukunft gestalten.

Die Maschinen

Die Entwicklung hilfreicher Maschinen war und ist ein ständiger Menschheitstraum. Wohl einer der ersten Automaten war bei den alten Griechen ein Rohr, aus dem nach Einwurf einer Münze ein Federkiel ausgegeben wurde. Weihwasserspender gaben auf Münzeinwurf geweihtes Wasser in Portionen aus. Dann entwickelte man Wasserräder, die beim Mahlen des Korns halfen. Schritt für Schritt ging die Entwicklung weiter. Die ersten Windmühlen entstanden in Persien (etwa 700 n.Chr.). Im Mittelalter (Frankreich 1180) lernte man auch in Europa die Kraft des Windes in Windmühlen zu nutzen. Das Spinnrad ersetzte (um 1298 aus Indien kommend) die Arbeit mit dem Spinnrokken. Mechanische Uhren zeigten ab dem 14. Jahrhundert zunächst die Zeit zwar nicht genauer an als Wasseruhren, waren aber besser zu handhaben und weniger störanfällig. Sie wurden jetzt mit einem Gewicht angetrieben.

Mit den mechanischen Uhren begann die Entwicklung mechanischer Maschinen überhaupt. In der Mitte des 15. Jahrhunderts schuf Gutenberg die Buchdruckerkunst. Waren vorher die Manuskripte noch mühsam in den Schreibstuben der Klöster von Hand geschrieben worden, so druckte jetzt Johann Gensfleisch zum Gutenberg, wie sein vollständiger Name war, als erstes Buch die Bibel in 300 gleichen Exemplaren.

Peter Henlein entwickelte in Nürnberg 1504 die erste Taschenuhr. Bei ihr war das Besondere, daß der Antrieb mit einer Zugfeder erfolgte. Eine neue Antriebsart war realisiert. Der weitere Fortschritt der Mechanik ging einher mit der Entwicklung der Uhren. Galileo Galilei entdeckte (1581 als Junge mit 17 Jahren) die Regelmäßigkeit von Pendelbewegungen. Er wurde mit seinen Versuchen zu fallenden Körpern zum Begründer der experimentellen Wissenschaft. Er war es auch, der das erste (zwar noch ungenaue) Thermometer entwickelte.

Mathematik und Rechenhilfen

Die erste Rechenhilfe war der Abakus. Er ist schon im Altertum nachgewiesen. Er hat sich in unterschiedlicher Form in Asien und Europa entwickelt und wurde für Additions- und Subtraktionsaufgaben benutzt. Noch heute kennt man in Rußland den Schtschoty, mit dem sogar die Kassiererinnen im Supermarkt schneller arbeiten als mit der Registrierkasse.

Ein für die europäische Mathematik besonders wichtiges Ereignis haben wir einem arabisch-persischen Mathematiker zu verdanken. Im Altertum hatte sich die römische Zahlenschreibweise durchgesetzt. Das römische Zahlensystem ist ein Additionssystem. Jedes Zeichen, jede Ziffer hat einen bestimmten Wert. Der Gesamtwert ergibt sich aus der Addition der einzelnen Ziffernwerte.

Im römischen Zahlensystem, wie wir es heute kennen, gibt es die dargestellten Ziffern. Dabei werden die Ziffern M, C, X und I als Grundzeichen, die anderen Ziffern D, L und V als Hilfszeichen bezeichnet. Die heutige Ziffernfolge basiert auf dem System der Römer, wird aber zum Teil erst seit dem Mittelalter in dieser Form verwendet.

Römische Ziffer	Dezimaler Wert
M	1000
D	500
C	100
L	50
X	10
V	5
I	1

Das römische System ist für Berechnungen sehr unhandlich. Größere Zahlenwerte sind schwer überschaubar. MCMXCIX ist zum Beispiel der Zahlenwert, der die Dezimalzahl 1999 repräsentiert. Es fehlt in dem Ziffernsystem auch noch die Ziffer für einen wichtigen Zahlenwert, nämlich die Ziffer Null.

Die Ziffer Null wurde erstmals im indischen Raum eingesetzt. Von dort wurde sie etwa um 800 von dem Mathematiker und Astronomen Mohammed b. Musa Al-Hwarizmi übernommen und in seinem grundlegenden Werk "Hisab al-gabr wa'l muqabala" (Deutsch: Die Arithmetik der Wiederherstellung und Gegenüberstellung) verwendet.

Al-Hwarizmi stammte aus Hwarizm (heute Chiwa in Usbekistan) und war vom Kalifen als Hofastronom an den Hof in Baghdad berufen worden. Auf seinen Namen wird der Begriff Algorithmus zurückgeführt.

Im Titel seines Hauptwerks findet man im Wort al-gabr außerdem die Grundlage für die heute gebräuchliche Bezeichnung Algebra für die Lehre von den Gleichungen. Die arabische Welt hat etwa zur gleichen Zeit von den Indern das Dezimalsystem übernommen. In Europa rechnete man immer noch mit dem römischen System. Das Dezimalsystem ist im Gegensatz zum römischen Additionssystem ein "Positions- oder Stellenwertsystem". Hier hat die Ziffer einen Wert, der mit dem Wert der Stelle multipliziert wird. Dabei wächst der Wert der Stellen von rechts nach links beim Dezimalsystem jeweils um eine Zehnerpotenz. Die Einer haben den Stellenwert 10^0 = 1, die Zehner den Stellenwert 10^1 = 10, die Hunderter entsprechend 10^2 = 100 ... Es ergibt sich daraus die allgemeine Formel

Zahlenwert = Ziffernwert * 10 ^Stelle,

wobei die Stellen von rechts nach links fortlaufend gezählt werden.

Für den interessierten Leser haben wir ein kleines BASIC-Programm geschrieben, das die Umwandlung von Zahlenwerten des Dezimalsystems in Ziffernfolgen des römischen Zahlensystems und umgekehrt erlaubt.

Wenn Sie noch keine Erfahrung mit dem Programmieren in BASIC haben, sollten Sie den Abschnitt "Programmieren des Computers" später in diesem Heft kurz durcharbeiten.

Für den Leser, der nur einmal kurz sehen will, wie das Programm funktioniert, haben wir eine selbständig lauffähige Version unter dem Namen "ROEM_DEZ.EXE" auf der Diskette. Dieses Programm können Sie unter DOS direkt durch Eingabe des Namens starten.

Nach der Entwicklung der Logarithmen durch den schottischen Mathematiker John Napier (1614 veröffentlicht) wurde 1622 der erste logarithmische Rechenschieber von dem englischen Mathematiker William Oughtred konstruiert. Mit Hilfe der Logarithmen und des Rechenschiebers wurde es möglich, Multiplikationen und Divisionen in einfache Additions- und Subtraktionsvorgänge, Potenzierungen in Multiplikationen und Wurzeloperationen zu Divisionen zu reduzieren. Rechenschieber werden noch heute als Rechenhilfe eingesetzt, wenn sie auch inzwischen vom Taschenrechner weitgehend ersetzt und in Geschwindigkeit, Leistungsfähigkeit und Genauigkeit übertroffen werden.

1642 erfand der französische Mathematiker Blaise Pascal eine mechanische Additionsmaschine, bei der gezahnte Rädchen die Stellen kennzeichneten wie im dezimalen Stellenwertsystem. Wenn eines der Rädchen zehn Einheiten gezählt hatte, wurde das Rädchen für die nächsthöhere Stelle um eine Einheit weitergedreht. 1692 wurde dann von Gottfried Wilhelm Leibniz eine Multiplikationsmaschine entwickelt, die durch Mehrfachwiederholung von Additions- oder Subtraktionsvorgängen multiplizieren und dividieren konnte. Leibniz war es auch, der um 1700 die theoretischen Grundlagen für das Binärsystem (siehe Abschnitt Grundlagen) schuf.

Im Jahre 1801 wurde von dem Franzosen Joseph-Marie Jaquard das nach ihm benannte und heute noch gültige Verfahren der Jaquard-Weberei entwickelt. Bei diesem Verfahren erfolgt das Anheben und Senken der Kettfäden, (bevor das Weberschiffchen hin- und hersaust) mit Hilfe einer

gelochten Pappkarte. Ist ein Loch vorhanden, wird der entsprechende Kettfaden angehoben, sonst bleibt er gesenkt.

Hier wurde erstmals das Prinzip der Ja/Nein-Mechanik angewandt, das später (1890) Hermann Hollerith für seine Zählkarten bei der amerikanischen Volkszählung verwenden sollte. Schon vorher aber war Charles Babbage um 1820 auf den Gedanken gekommen, eine über Lochkarten programmierbare mechanische Rechenmaschine zu bauen. Er konnte seine Idee nicht umsetzen, da er immer wieder an die Grenzen der damaligen technischen Möglichkeiten stieß.

1847 veröffentlichte der englische Mathematiker George Boole seine "Mathematische Analysis der Logik", in der er die heute noch von uns beim Programmieren von digitalen Computern verwendeten Prinzipien der formalen Logik entwickelte, die wir auch nach ihm Boolesche Algebra nennen. Wir werden uns mit diesem Thema noch ausführlicher im Abschnitt Grundlagen beschäftigen.

Im Jahr 1930 stellte der amerikanische Elektroingenieur Vannevar Bush den ersten Rechner vor, der neben mechanischen auch elektronische Bauteile (Radioröhren) benutzte, um Differentialgleichungen zu lösen.

1936, weitgehend von der Weltöffentlichkeit unbemerkt, begann Konrad Zuse mit der Entwicklung seines ersten programmgesteuerten elektronischen Rechners. 1946 wurde in den USA der ENIAC (electronic numerical integrator and computer), ein Riesenrechner in jeder Beziehung, der Öffentlichkeit vorgestellt. Dieser Rechner benötigte 140 Quadratmeter Fläche und wog fast 30 Tonnen. Er arbeitete mit Elektronenröhren.

Neun Jahre später war ENIAC in bezug auf Rechenleistung, Schnelligkeit, Größe und Betriebskosten von neuen Rechnertypen schon so überholt, daß er außer Betrieb genommen wurde.

Einige Meilensteine aus der Entwicklung des Computers seien hier noch erwähnt: 1952 begann die aus der Hollerith-Company entstandene IBM (International Business Machines) mit der Produktion und dem Verkauf von Elektronenrechnern begonnen. Dies waren zunächst Großrechner, die im Verhältnis zu dem, was heute in unseren Büros steht, immer noch Riesen waren.

Im Januar 1975 zeigte die Zeitschrift Popular Electronics den ersten Bausatz für den Selbstbau von Computern, den Altair. Tandy TRS-80 und Commodore PET 2001 kamen als die ersten wirklich ernsthaften Mikrocomputer auf den Markt. Kurz danach stellten Steven P. Jobs und Steve Wozniak den ersten Apple vor, den ersten Personal Computer, der sehr schnell ein Verkaufserfolg wurde.

1981 wurde der erste PC von IBM präsentiert. Dieser Rechner war der erste, bei dem ein "offenes System" jedermann die Ergänzung und Erweiterung des Rechners - und seinen Nachbau - ermöglichte. Idee und allgemeine Technik von IBM, Prozessoren von Intel, Betriebssysteme von Microsoft, das wurde "der" PC. Das Apple-System dagegen war ein geschlossenes System, dessen Struktur nur in Teilen öffentlich bekannt gemacht wurde. Dieser Unterschied führte mit dazu, daß an der Weiterentwicklung des IBM-PC viele Unternehmen mitwirkten und mitwirken und dieses System zum Industriestandard wurde.

Die Elektronenrechner brauchten keine "verantwortliche Position" zu suchen, wie es noch in einem Buchtitel 1967 hieß, sie hatten sie. Elektronenrechner boten den Ansatzpunkt für die Steuerungsanlage, das "Gehirn" der Roboter.

Energie und Antrieb

Ohne Antrieb gibt es keinen Roboter, und ohne Energie keinen Antrieb. Als erste Energien nutzte der Mensch die aus brennendem Holz strahlende Wärme und das Licht. Sie erst machten seine Höhle wohnlich. Und das Feuer war es auch, das zu den ersten technischen Entdeckungen führte.

Etwa 4000 v.Chr. entdeckten Menschen in Vorderasien, daß aus bestimmten Steinen, genauer gesagt, aus Erzen, Kupfer zu gewinnen war. Dieses Material konnte man verformen. Es war zwar für dauerhafte Werkzeuge zu weich, aber schon bald fand man die Bronze. Sie ist eine Kupfer-Zinn-Legierung, die gleichfalls aus natürlich vorhandenen Erzen gewonnen werden konnte. Damit fanden die Menschen eine für die Entwicklung der Werkzeuge - und der Waffen - entscheidende Möglichkeit.

Dieses Metall konnte man schärfen, und es behielt diese Schärfe bei. Beim Kampf um Troja schützten sich die Kämpfer durch bronzene Rüstungen, sie nutzten Bronzeschwerter, mit Bronze beschlagene Schilde und Pfeile und Speere mit Bronzespitzen. Die für die gesamte technische Entwicklung wohl entscheidenste Erfindung, das Rad, ist gleichfalls in Vorderasien um diese Zeit schon bekannt.

Inzwischen konnte man auch schon die Windenergie zur Fortbewegung von Schiffen einsetzen. In Ägypten entdeckte man um 2500 v.Chr., wie man Glas aus Sand herstellen konnte. Ab etwa 1000 v.Chr. ersetzte das mit Holzkohle aus Erzen gewonnene Eisen zunächst bei den Hethitern in Kleinasien die Bronze. Der Grieche Archimedes stellte 260 v.Chr. die ersten genauen Berechnungen der Hebelwirkung an. Er fand auch das "einfache" Prinzip der zerstörungsfreien Messung des Volumens von Körpern. Taucht man einen Körper (Gegenstand) völlig in ein randvoll mit Wasser gefülltes Gefäß, dann entspricht das Volumen des überfließenden Wassers dem des Gegenstandes.

Wasserräder und Windmühlen stellten die Kraft zur Verfügung, Mühlen zu anzutreiben. 1712 erfand der englische Ingenieur Thomas Newcomen die atmosphärische Dampfmaschine. In ihr wurde in einem Zylinder das Wasser zu Dampf erhitzt und sofort wieder abgekühlt. Aus ihr entwickelte 1764 der schottische Ingenieur James Watt die Dampfmaschine, bei der Zylinder und Kondensator getrennt waren. Als er 1781 dann eine Vorrichtung entwickelte, mit der die Auf-/Ab-Bewegungen des Kolbens in Drehbewegungen eines Rades umgesetzt werden konnten, war die Grundlage geschaffen für die weite Verbreitung seiner Maschinen. Dies war auch der Anfang der "industriellen Revolution". Jetzt beschleunigte sich die Entwicklung. 1787 baute der Amerikaner John Fitch den ersten funktionstüchtigen Dampfer, der allerdings wirtschaftlich für

ihn kein Erfolg wurde. Erst das 1807 von dem amerikanischen Erfinder John Fulton gebaute Dampfschiff Clermont rechnete sich. 1819 überquerte die Savannah als erstes Dampfschiff den Atlantik. Der Engländer George Stephenson baute 1825 die erste zuverlässig arbeitende Dampflokomotive, die schneller fuhr, "als ein Pferd laufen" konnte. 1853 baute der "Begründer der Aerodynamic", der englische Ingenieur George Cayley, das erste Gleitflugzeug, mit dem ein Mensch flog. Er entwickelte das Prinzip des Flugzeugs, wie wir es auch heute noch kennen.

1860 baute der Franzose Jean-Joseph-Etienne Lenoir die erste funktionstüchtige Verbrennungskraftmaschine und kurz danach eine mit dieser angetriebene "pferdelose Kutsche". 1868 erfand der Amerikaner George Westinghouse die Luftdruckbremse für die Eisenbahn. 1876 wurde von dem deutschen Ingenieur August Otto die erste Viertaktmaschine, der "Ottomotor", konstruiert. 1885 wurde von Carl Friedrich Benz das erste Automobil mit Viertaktmotor gebaut. 1888 ließ sich der britische Erfinder John Boyd Dunlop den Gummireifen patentieren.

1891 startete Otto Lilienthal zu seinem ersten Gleitflug. 1903 starteten die Amerikaner Orville und Wilbur Wright zum ersten Mal mit einem durch einen Verbrennungsmotor angetriebenen Flugzeug und blieben fast eine Minute in der Luft. Das Flugzeug war erfunden. 1903 war es auch, als der russische Physiker Konstantin Ziolkowski sich eingehend mit der Raumfahrt auseinandersetzte. In einer Artikelserie für ein Luftfahrtmagazin beschrieb er Raumanzüge, Satelliten, den Bau einer Raumstation und die Kolonisierung des Sonnensystems. Es sollte zwar noch über fünfzig Jahre dauern, aber als am 4. Oktober 1957 der erste Sputnik das Zeitalter der Raumfahrt ein"piepte", da wurde wahr, was viele Menschen vorher geträumt hatten, der Mensch war in der Lage, seinen Planeten zu verlassen.

Quo Vadis, Roboter?

Am Schluß eines Abschnittes, der sich mit der Entwicklung der Roboter befaßt, stellen wir die Frage: "Wohin führt uns der Weg?" Als die ersten Maschinen den Webern ihre Arbeit nahmen, reagierten sie heftig, sie zerstörten die Webstühle. Aber, wie die Geschichte zeigt, konnte der handgreifliche Protest die Entwicklung dieser und anderer Maschinen nicht aufhalten. Wie die Optimisten erwarteten, nahmen die Maschinen den Arbeitern die schwere Arbeit ab, das Leben wurde einfacher.

Wenn wir vor zwei bis drei Jahren in der damaligen Bundesrepublik eine Umfrage gestartet hätten, wie der Mann auf der Straße die allgemeine Entwicklung beurteilt, dann wäre trotz allem ein begrenzter Optimismus für die Antworten kennzeichnend gewesen. Seit vielen Jahren aber war zunehmende Technikverdrossenheit zu beobachten. Und wenn man heute eine Umfrage dieser Art machen würde, wäre das Ergebnis sicher noch weitaus pessimistischer. Verunsicherung macht sich überall breit. Das Auto, jahrzehntelang "der Deutschen liebstes Kind" wird verdammt. Plötzlich ist es an allem schuld. Und opportunistische Politiker nutzen die Zeit. Plötzlich stehen sie da und glauben mit einem Benzinpreis von fünf Mark pro Liter alle Probleme lösen zu können. Aber eine wirkliche Alternative zum Auto, zu der Möglichkeit, individuell jederzeit an (fast) jeden Ort kommen zu können, eine solche Alternative zeigt keiner auf. Der technische Fortschritt führt nicht zwangsläufig zum Fortschritt für die Menschheit.

Nicht wir kontrollieren weiter die Maschine, sondern die Maschine beginnt, uns zu kontrollieren, oder vielmehr, andere nutzen die Maschine, um uns zu kontrollieren. Es gibt auch heute noch die bitter ernst zu nehmenden Wissenschaftler, die es für das erstrebenswerte und realisierbare Ziel der Rechnerentwicklung halten, den Menschen völlig zu ersetzen.

Es gibt also auf der einen Seite den Glauben an die Vollkommenheit der Maschinen. Und damit gepaart ist der Glaube, ja die Gewißheit der Unvollkommenheit des Menschen. In der letzten Zeit hat man von einem gewissen Flugzeugtyp des öfteren von Abstürzen gehört. Die immer gleiche Erklärung der Unfallursachen war, daß die Unfälle auf menschliches Versagen zurückzuführen waren. Was ist aber wirklich passiert? Man hat eine Maschine vollgepfropft mit Technik.

Man hat diese Technik so vollkommen gemacht, daß man glaubte, den Bordingenieur einsparen zu können. Und doch ließ man dem Piloten nach dem Grundsatz "Man kann ja nie wissen" die Möglichkeit, in dieses so vollkommene System einzugreifen. Da liegt der Fehler. Entweder ist ein System so gut, daß man den Bordingenieur nicht mehr braucht, und dann sollte auch der Pilot nicht eingreifen können, oder man braucht eben doch den dritten Mann im Cockpit. Denn die viele Technik kann den Menschen eben doch nicht ersetzen. Was man jetzt verlangt, ist, der Pilot soll auch noch diese Arbeit machen. Und wenn es nicht klappt, war es eben mal wieder "menschliches Versagen".

Die Frage ist also auch in diesem Zusammenhang, wohin der Weg uns führt, uns und die Maschinen. Ein wichtiger Ansatzpunkt ist die Erkenntnis, daß wir unsere Instrumente der Entwicklung nicht angepaßt haben. Was geschieht mit einer Volkswirtschaft, bei der immer mehr Menschen immer weniger Arbeit haben, weil ja die Maschinen uns die "schwere Arbeit" abnehmen. Es hat keinen Sinn, die Maschinen zu zerstören, das haben wir hoffentlich alle begriffen. Aber wir müssen Lösungen finden für unsere Probleme. Diese Lösungen sind sicher nicht einfach und gerade die einfach erscheinenden sind die trügerischen. Nicht "Auto weg" sondern "Was anstelle des Autos?" muß es heißen. Da sind unsere kreativen Kräfte gefordert. Da können wir mit den Maschinen, die wir beherrschen, die Lösung finden. Sie als Leser dieses Heftes haben schon damit angefangen.

Wenn wir wissen, wie etwas funktioniert, dann können wir es auch im positiven Sinn für uns nutzen. Nicht "Mensch oder Maschine", sondern "Mensch und Maschine" muß die Devise der Zukunft lauten.

Am Ende, der ein neuer Anfang sein wird, kann es dann doch heißen: Dein Freund, der Roboter.

Schlüssel zur Computertechnik

Der folgende Abschnitt richtet sich an den Computerneuling. Er erhält hier die Grundinformationen für die Arbeit mit dem Rechner. Dabei wird auf die Hardwareseite nicht eingegangen. Es werden vielmehr die allgemeinen Informationen gegeben, die als Hintergrundwissen zur Arbeit mit dem Computer vorhanden sein sollten.
Computer sind Rechner. Sie verarbeiten in Zahlen codierte Informationen. Diese Verarbeitung folgt den Regeln der Mathematik. Aus diesem Grund ist es gut, einige Grundlagen aus diesem Bereich zu kennen, wenn man sich intensiver mit dem Computer und seinen Möglichkeiten beschäftigt.

Für die Arbeit mit den Digitalrechnern gibt es zwei Zahlensysteme, die man kennen sollte, das Dualsystem und das Hexadezimalsystem. Digitalrechner kennen zwei interne Schaltzustände, die man mit "Ein" und "Aus" bezeichnen kann.

Diese Zustände kann eine Binärstelle (Bit, die kleinste Informationseinheit im Rechner) einnehmen. Als Ziffern verwendet man dafür "1" und "0".

Es hat sich in der Computertechnik durchgesetzt, Gruppen von jeweils acht Bit zu einem Byte, zwei Bytes zu einem Word (auch Wort) zusammenzufassen.
Ein Byte kann 256 verschiedene Zahlenwerte (2^8 = 2*2*2*2*2*2*2*2) darstellen. Bezieht man die Null als ersten Zahlenwert mit ein, so kann ein Byte Zahlenwerte zwischen 0 und 255, ein Wort 255*255 = 65536 Werte insgesamt, also zum Beispiel die Zahlen zwischen -32767 und +32768, die sogenannten (Short-)Integerwerte darstellen. Als weiterer wichtiger Wert werden 2^{10} (= 1024) Bytes als Kilobyte bezeichnet. Die Dezimalwerte zu einem Binärwert berechnet man nach der Formel:

DezWert = ZiffernWert * $2^{\text{Stellenwert}}$

Die Zahlensysteme der Computer

In unserer kleinen Tabelle für Computermaße haben wir die wichtigsten Zahlenwerte zusammengefaßt.

			2^n
1 Bit			
8 Bit	1 Byte	1 Word	2^0
16 Bit	2 KByte		2^1
1024 Byte	1 KByte		2^10
65536 Byte	64 KByte		2^16
655360 Byte	640 KByte		
1048576 Byte	1024 KByte	1 MByte	2^20
1073741824 Byte	1048576 KByte	1024 MByte	1GByte 02^30

KByte = Kilobyte MByte = Megabyte GByte = Gigabyte

Das Rechnen mit Binärzahlen

Die Informationen, die vom Computer verarbeitet werden können, sind binär codiert. Auch die Ausgabesignale sind dies. Wir werden uns im nächsten Abschnitt auch mit den Anschlüssen des Rechners beschäftigen. Zum Verständnis des dort Gesagten ist erforderlich, daß man das binäre Zahlensystem kennt und damit umzugehen versteht.

Wie wir schon vorher sagten, ist die Basis des binären Systems die Zahl 2. Mit Potenzen dieser Zahl können alle positiven ganzen Zahlen dargestellt werden. Für die Umrechnung der Binär- in die Dezimalzahl können wir folgende Formel verwenden:

DezWert = Ziffernwert * $2^{\text{Stellenwert}}$

Was ist aber mit der umgekehrten Berechnung? Sehen wir uns dazu zunächst ein paar der in der nachstehenden Tabelle dargestellten Zahlenbeispiele an. Es ist

	8421 (Wertigkeit der Stelle)
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

Die einzelnen Stellen haben (von rechts nach links) die Wertigkeit 1, 2, 4 und 8. Wenn an einer Stelle eine 1 steht, wird der Wert dieser Stelle zum Gesamtwert addiert.

So ergibt zum Beispiel

3 + 2 + 1 = 7

Es ist logisch: Wollen wir aus einer Dezimalzahl eine Binärzahl machen, müssen wir den umgekehrten Weg gehen. Es gilt:

$ZiffernWert_{Stelle}$ = INT(DezWert / 2^Stellenwert)

Wir haben diese Berechnung der Binärziffernkombination in eine Funktion eingefügt. Diese finden Sie nachfolgend im Pseudocode. Unser Pseudocode verwendet deutsche Schlüsselwörter. Sie können alle Umgangscode-Prozeduren in jede beliebige Programmierhochsprache übersetzen.

Für unsere BASIC-Freunde haben wir auf der Diskette ein Programm für die Pseudocode-Übersetzung. BASICTRS.EXE übersetzt die Prozeduren in BASIC-Textcode, den Sie so in jedes prozedurale BASIC also QBASIC, QuickBASIC, Visual Basic für DOS und Visual Basic für Windows übernehmen können. Es werden ASCII-Textfiles erstellt, die man problemlos in jede Entwicklungsumgebung laden kann. Das Programm kann so erweitert werden, daß es auch Übersetzungsbibliotheken für andere Programmiersprachen laden kann.Weitere Informationen zum Pseudocode und dem Übersetzungsprogramm finden Sie im Abschnitt "Wie man Computer programmiert".

In der Funktion BIN$ werden alle Integer-(Short-)Zahlen in Binärcode übersetzt. Die Funktion BINVAL übernimmt die Rückübersetzung. Diese Funktionen sind in den genannten BASIC-Versionen nicht integriert, können aber auf diese Weise leicht ersetzt werden.

In der Funktion BIN$ werden die zwei Grenzwerte, zwischen denen sich (Short-)Integer-Zahlen bewegen dürfen, als Grenzen für die Umwandlung verwendet. Dieser Bereich ist 64 kByte groß, kann damit mit "normalen" Bytes (2^16 = 65536 (Zahlenbereich von -32767 bis +32768)) dargestellt werden. Es ergeben sich Darstellungen mit 16 Bit (= 2 Byte = 1 Word) Länge. Dabei wird das erste Bit (Minus-Bit) immer dann gleich 1 gesetzt, wenn die umzusetzende Zahl negativ ist und bei positiven Zahlen gleich 0. Unsere Funktion erstellt Zeichenketten der Bit mit 16 Zeichen. In der Mitte wird, der leichteren Lesbarkeit halber, eine Leerstelle eingefügt, und die Zeichenkette erhält zur Identifikation die Vorzeichen &B für Binär.

Addition und Subtraktion binärer Werte

Die Addition und die Subtraktion im Dualsystem erfolgt wie in jedem Stellenwertsystem, zum Beispiel dem Dezimalsystem. Von der Stelle mit dem niedrigsten Stellenwert bis zur Stelle mit dem höchsten Wert werden die Ziffernwerte addiert oder subtrahiert. Überträge bei Über- oder Unterschreiten des Wertebereichs einer Stelle dienen zum Ausgleich.

Je ein Beispiel soll die Zusammenhänge verdeutlichen. Wir haben, quasi zur Kontrolle, die Berechnung jeweils auch im Dezimalsystem dazugeschrieben. Bitte beachten Sie, daß wir für die duale Darstellung in unseren Beispielen die Stellenzahl auf fünf begrenzt haben. Dies ist von besonderer Bedeutung bei der mit Hilfe des Komplements auf die Addition zurückgeführten Subtraktion.

1. Addition

	01001	9	
+	00111	7	
	-------	---	
Merke	111		
	10000	16	2^5

2. Subtraktion

	01001	9	
-	00111	7	
	---------	---	
Merke	11		
	00010	2	2^1

3. Subtraktion als Addition mit Komplement

	01001	9
+	11000	24
	---------	--
	100001	33

Die jetzt vorn überschießende 1 wird an der Stelle mit dem niedrigsten Stellenwert (ganz rechts) addiert. Das Ergebnis ist:

	00001	1
+	1	1
	----------	--
	00010	2

Als Komplement bezeichnet man den Differenzwert zwischen dem Zahlenwert und der bei der maximalen Stellen-

```
Zahl 1 =          123       &B00000000 01111011
Zahl 2 =          13        &B00000000 00001101

Zahl1 + Zahl2     136       &B00000000 10001000
Zahl1 - Zahl2     110       &B00000000 01101110
Zahl1 * Zahl2     1599      &B00000110 00111111
Zahl1 \ Zahl2     9         &B00000000 00001001 + (.4615384615384617)
NOT Zahl1        -123       &B11111111 10000100
NOT(NOT Zahl1)    123       &B00000000 01111011
Zahl1 AND Zahl2   9         &B00000000 00001001
Zahl1 OR  Zahl2   127       &B00000000 01111111
Zahl1 XOR Zahl2   118       &B00000000 01110110
Zahl1 IMP Zahl2  -114       &B11111111 10001101
Zahl1 EQV Zahl2  -118       &B11111111 10001001


Auf Drucker ausgeben <D>  Abbruch <Q> oder <Esc>
```

Binärzahl und Komplement

zahl größtmöglichen Zahl. Im Dualsystem kann das Komplement ganz einfach gebildet werden, indem anstelle einer 0 eine 1 und umgekehrt gesetzt wird.

In unserem Bild "Binärzahl und Komplement", wird eine Bildschirmausgabe des Programms BINDEMO gezeigt. Dort sehen Sie die 20 ersten Binärzahlen und ihr jeweiliges Komplement.

Multiplikation und Division im Dualsystem

Man kann mit Dualzahlen nicht nur addieren und subtrahieren, sondern auch multiplizieren und dividieren. Diese Rechenvorgänge werden im Computer zwar auf die Addition zurückgeführt. Wir wollen sie aber dennoch der Vollständigkeit halber hier aufführen, und weil das Ergebnis insbesondere der Division zeigt, welche Probleme im Genauigkeitsbereich von Zahlen in einem Rechner auftreten können. Auch für diese beiden Operationen zeigen wir Beispiele. Beide Vorgänge laufen genauso ab wie beim Dezimalsystem, nur daß eben der maximale Ziffernwert 1 ist.

1. Multiplikation

```
  01001 * 1011          9*11
  ------------          ----
  01001                 9
   00000                 9
    01001
     01001
  --------              ----
Merke 1
  01100011              99
  2^6+2^5+2^1+2^0
```

2. Division

```
  11001 : 1010 = 10,1   25:10 = 2,5
  ------------          -----
  1010                  20
  ----                  --
    101                  50
    ---                  50
    1010                 --
    1010                  0
    ----
       0
```

Bei der Division ergibt sich ein binärer Bruch, da die erste Zahl nicht ohne Rest durch die zweite geteilt werden kann.

Es ist wichtig bei Divisionen daran zu denken, daß im Rechner nur 1 und 0 als Schaltzustände gewertet werden und Divisionen zwangsläufig zu Rundungsproblemen führen, wenn die Zahlen nicht ohne Rest teilbar sind.

Aussagenlogik mit Booleschen Operatoren

Der englische Mathematiker George Boole (1814-1865) hat die Grundlagen für die Schaltalgebra erarbeitet, die nach ihm auch Boolesche Algebra genannt wird. Die dafür eingesetzten Operatoren nennt man logische oder auch Boolesche Operatoren. Bei deren Verwendung erhält man als Ergebnis immer die Aussage in Form einer Booleschen Variablen, die den Wert "Wahr" (englisch: true) für eine erfüllte Bedingung oder "Falsch" (englisch: false) für eine nicht erfüllte Bedingung annimmt.

Als Werte werden in den Programmiersprachen die beiden Zahlenwerte -1 für "Wahr" und 0 für "Falsch" verwendet. Da das Ergebnis immer eine Aussage ist, wird das logische System, das die Regeln zusammenfaßt, auch als Aussagenlogik bezeichnet. Die (auch in BASIC verfügbaren) Booleschen Operatoren faßt unsere Tabelle zusammen. Da das Verständnis der Wirkung der logischen Operatoren wichtig für das Verständnis der sogenannten Bitoperationen ist, wollen wir hier darauf etwas näher eingehen.

Aussageverbindungen

Aus einzelnen Aussagen kann man durch Verknüpfung miteinander Aussagenverbindungen gewinnen von der Art wie: Wenn A wahr ist und B wahr ist, ist auch A UND B wahr. Der hier eingesetzte Operator wird zur besseren Unterscheidung vom sprachlichen "und" in Großbuchstaben geschrieben.

Diese Schreibweise wenden wir grundsätzlich in diesem Heft an, wenn wir logische Operatoren darstellen. Da mit den Operatoren Verbindungen von Aussagen zu neuen Aussagen zusammengefaßt werden, bezeichnet man die Operatoren auch als Junktoren (lateinisch: iunctura = Verbindung) und die Ergebnisse als Junktionen.

NOT Negation, Verneinung NOT A
Die Verneinung einer Aussage ergibt eine Umkehrung von deren Wahrheitswert. Dieser Operator ergibt somit das Komplement einer Zahl.

AND Konjunktion A AND B
ergibt wahr, wenn A und B zugleich wahr sind.

OR Disjunktion A OR B
ergibt wahr, wenn A oder B oder beide wahr sind.

XOR Exklusives ODER A XOR B
ergibt nur dann wahr, wenn eine der Aussagen wahr ist, die andere aber falsch.

IMP Implikation A IMP B
ergibt falsch, wenn A wahr, B falsch ist. In allen anderen Fällen ergibt sich wahr.

EQV Äquivalenz A EQV B
ergibt wahr, wenn beide Aussagen den gleichen Wahrheitswert haben.

A	NOT A
w	f
f	w

Wahrheitstafel für die Negation

A	B	A AND B
w	w	w
w	f	f
f	w	f
f	f	f

Wahrheitstafel für die Konjunktion

A	B	A OR B
w	w	w
w	f	w
f	w	w
f	f	f

Wahrheitstafel für die Disjunktion

A	B	A XOR B
w	w	f
w	f	w
f	w	w
f	f	f

Wahrheitstafel für das Exclusive ODER

A	B	A IMP B
w	w	w
w	f	f
f	w	w
f	f	w

Wahrheitstafel für die Implikation

A	B	A EQV B
w	w	w
w	f	f
f	w	f
f	f	w

Wahrheitstafel für die Äquivalenz

Die Überschrift dieses Unterabschnitts lautet: Das Rechnen mit Binärzahlen. Bisher haben wir zwar noch keine Berechnungen angestellt, aber mit der Darstellung der Wirkung der Operatoren eine wichtige Grundlage für die weiteren Betrachtungen geschaffen. Es läßt sich nämlich jetzt anhand von "Wahrheitstabellen" auf einfache Weise das Ergebnis einer Operation mit einem der Junktoren ablesen. Wir haben in unserem Bild die Wahrheitstabellen aller dargestellten Operatoren zusammengefaßt.

Sie finden darin den jeweiligen Operator, das mathematische Formelzeichen für ihn und dann eine Matrix von Verknüpfungen. Sehen Sie sich z.B. die Matrix für XOR, das "Exclusive ODER" an. Es ergibt immer dann "Wahr", wenn die beiden Aussagen einander nicht entsprechen.

Alle Rechenvorgänge im Computer erfolgen, wie Sie wissen, bitweise. Ein Bit kann den Wert 0 oder den Wert 1 erhalten. Wenn wir jetzt 0 = Falsch und 1 = Wahr setzen, können wir auch die Kombination zweier Binärwerte mit Hilfe der dargestellten Operatoren vornehmen.

Einige Beispiele sollen dies verdeutlichen. Es ergeben:

Aussage	Bitmuster	Dezimalwert
A	1010	10
B	0101	5
C	1001	9
NOT A	0101	5
A AND B	0000	0
A OR B	1111	15
A XOR B	1111	15
A XOR C	0011	3
A IMP B	0101	5
A IMP C	0010	4
A EQV B	0000	0
A EQV C	1100	12

Wenn Sie bei der Überprüfung der Ergebnisse jeweils die Wahrheitstafeln zu Rate gezogen haben, sollten Sie sich diese vielleicht jetzt kopieren und für künftige Bitoperationen neben Ihren Rechner legen.

Eine weitere Möglichkeit bieten wir mit dem Programm BINDEMO.EXE, das Sie auf der Diskette finden. Mit diesem Programm können Sie sich für Ganzzahlen die jeweiligen Bitmuster, Komplemente und Hexwerte sowie im Untermenü Bitoperationen die Ergebnisse bei Anwendung der verschiedenen in BASIC verfügbaren logischen Operatoren ansehen und, nach Bedarf, ausdrucken.

Ein paar Kernprozeduren aus dem Programm haben wir wieder in Pseudocode gesetzt. Im Programm ist ein vollständiges System von Funktionen für Bitmanipulationen integriert.

Diese Funktionen, von denen wir hier vier etwas näher ansehen wollen, ergänzen die Funktionen von QBASIC, QuickBASIC oder Visual Basic1.0. Die Funktionen können in Form von Unterprogrammen auch in GW-BASIC-Programme eingebaut werden.

Die erste Funktion BinAND$ steht exemplarisch für die ganze Gruppe zur Verwendung der logischen Operatoren. Durch die Einfügung der logischen Operationen in Funktionen gewinnt man zwar nur wenig Speicherplatz, Programme werden aber dadurch transparenter und leichter zu warten. In der zweiten Funktion BitZuBIN$ wird in der zweiten Zeile nach der Funktionsöffnung auf eine zusätzliche Funktion NurBIT$ zugegriffen.

In dieser Funktion, die in unserem Programm natürlich enthalten ist, werden die Binärkennzeichnung "&B" und das durch unsere Funktion BIN$ in der Mitte eingefügte Leerzeichen entfernt.

Als letzte in diesem Ausschnitt aus dem Programm dargestellte Prozedur sehen Sie eine Funktion, die gleichfalls als Beispiel für eine ganze Gruppe steht. In der Funktion BINdurchBIN$ wird aus zwei, als binäre Zeichenkette übergebenen Werten das Ergebnis der Division und der Rest der Ganzzahldivision ermittelt und als BINdurch-BIN$ zurückgegeben. Analog sind im beschriebenen Programm Addition, Subtraktion und Multiplikation von Binärzeichenketten in Funktionen eingefügt.

Wir sind an dieser Stelle deshalb so detailliert auf das Dualzahlensystem eingegangen, weil alle Informationen, die der Rechner von außen über Zuleitungen erhält oder nach außen sendet, binär codiert sind. Dies gilt also auch für alle Signale, die etwa von Sensoren an den Rechner oder vom Rechner an ein Interface gehen. Sie sind immer digitalisiert, also in einzelne Bit zerlegt.

Das Hexadezimalsystem

Da die Codierung im Binärsystem, das der Rechner unmittelbar versteht, langwierig ist und leicht zu Fehlern führen kann, wurde das Hexadezimalsystem eingeführt. Dieses hat die Basis 16 und ergibt wesentlich kürzere und damit überschaubarere Codierungen.

Da die 16 eine gerade Zahl des binären Systems ist (2^4 = 16) lassen sich hexadezimal codierte Informationen mit geringem Aufwand in die binären Codes der Maschinensprache übersetzen.

```
Zahl1 + Zahl2      123     &B00000000 01111011
Zahl1 - Zahl2      77      &B00000000 01001101
Zahl1 * Zahl2      2300    &B00001000 11111100
Zahl1 \ Zahl2      17      &B00000000 00000100 +(.3478260869565215)
NOT Zahl1         -100     &B11111111 10011011
NOT(NOT Zahl1)     100     &B00000000 01100100
Zahl1 AND Zahl2    4       &B00000000 00000100
Zahl1 OR  Zahl2    119     &B00000000 01110111
Zahl1 XOR Zahl2    115     &B00000000 01110011
Zahl1 IMP Zahl2   -96      &B11111111 10011111
Zahl1 EQV Zahl2   -115     &B11111111 10001100


Auf Drucker ausgeben <D>  Abbruch <Q> oder <Esc>
```

Das BINDEMO-Programm

Für die Umrechnung von Dezimalzahlen in Hexzeichen verwendet man folgende Formel:

DezWert = Ziffernwert * 16^Stellenwert

In der Tabelle "Binär und Hexadezimal" haben wir die ersten 16 Binärwerte von 0 bis 15 und die dazugehörende Hexadezimalschreibweise zusammengefaßt.

Für die Ziffern ab 10 werden in der Hexadezimalschreibweise die Buchstaben A bis F verwendet.

Dez	Binär	Hex
0	0000	00
1	0001	01
2	0010	02
3	0011	03
4	0100	04
5	0101	05
6	0110	06
7	0111	07
8	1000	08
9	1001	09
10	1010	0A
11	1011	0B
12	1100	0C
13	1101	0D
14	1110	0E
15	1111	0F

Auch Hexwerte werden wie unsere Binärwerte als Zeichenketten dargestellt. Die Vorzeichenkombination ist "&H". In BASIC stehen zur Umwandlung folgende Befehlskombinationen zur Verfügung:

Dez zu Hex	H$ = HEX$(Dezimalwert)
Hex zu Dez	Wert = VAL("&H"+H$)

Wichtig ist, daß bei der Rückverwandlung einer Hex-Zeichenkette in den Dezimalwert die Vorzeichenkombination für den VAL-Befehl mit angegeben sein muß.

In unserem Bild sehen Sie, daß das Programm BINDEMO.EXE neben den Binärwerten auch die Hex-Werte ausgibt. Sie können das Programm als Hilfsprogramm sowohl unter DOS als auch, im Fenster gestartet, unter Windows sicher immer wieder gebrauchen.

```
DECLARE FUNCTION BINplusBIN$ (B1$, B2$)
DECLARE FUNCTION BINminusBIN$ (B1$, B2$)
DECLARE FUNCTION BINmalBIN$ (B1$, B2$)
DECLARE FUNCTION BINdurchBIN$ (B1$, B2$)
DECLARE FUNCTION ShiftR$ (D$, Stellen%)
DECLARE FUNCTION ShiftL$ (D$, Stellen%)
DECLARE FUNCTION NurBIT$ (D$)
DECLARE FUNCTION BitZuBIN$ (D$)
DECLARE FUNCTION BinNOT$ (Z&)
DECLARE FUNCTION BinAND$ (Z1&, Z2&)
DECLARE FUNCTION BinOR$ (Z1&, Z2&)
DECLARE FUNCTION BinXOR$ (Z1&, Z2&)
DECLARE FUNCTION BinIMP$ (Z1&, Z2&)
DECLARE FUNCTION BinEQV$ (Z1&, Z2&)
DECLARE SUB DemoBitOperation ()
DECLARE SUB MiniMenu ()
DECLARE FUNCTION Not$ (Z&)
DECLARE SUB DemoBinaer ()
DECLARE FUNCTION BINVAL& (Z$)
DECLARE FUNCTION BINKompl$ (Wert$)
DECLARE FUNCTION BIN$ (Zahl&)

MiniMenu
END

FUNCTION BIN$ (Z&)
  Zahl& = Z&
  IF Zahl& > 32767 THEN BIN$ = "Zahl zu groß": EXIT FUNCTION
  IF Zahl& < -32768 THEN BIN$ = "Zahl zu klein": EXIT FUNCTION
  Minus$ = "0"
  IF Zahl& < 0 THEN Minus$ = "1"
  FOR m = 14 TO 0 STEP -1
    IF (Zahl& AND (2 ^ m)) THEN
      A$ = A$ + "1"
      Zahl& = Zahl& MOD (2 ^ m)
    ELSE
      A$ = A$ + "0"
    END IF
    IF m MOD 8 = 0 THEN A$ = A$ + " "
  NEXT
  BIN$ = "&B" + Minus$ + A$: A$ = ""
END FUNCTION

FUNCTION BinAND$ (Z1&, Z2&)
  BinAND$ = BIN$(Z1& AND Z2&)
END FUNCTION

FUNCTION BINdurchBIN$ (B1$, B2$)
  Z1& = (BINVAL&(B1$) \ BINVAL&(B2$))
  Z2# = (BINVAL&(B1$) / BINVAL&(B2$))
  Rest# = Z2# - Z1&
  BINdurchBIN$ = BIN$(Z1&) + "+ (" + LTRIM$(STR$(Rest#)) + ")"
```

```
END FUNCTION

FUNCTION BinEQV$ (Z1&, Z2&)

 BinEQV$ = BIN$(Z1& EQV Z2&)
END FUNCTION

FUNCTION BinIMP$ (Z1&, Z2&)
  BinIMP$ = BIN$(Z1& IMP Z2&)
END FUNCTION

FUNCTION BINKompl$ (W$)
  BINKompl$ = BIN$(NOT (BINVAL&(W$)))
END FUNCTION

FUNCTION BINmalBIN$ (B1$, B2$)
  BINmalBIN$ = BIN$(BINVAL&(B1$) * BINVAL&(B2$))
END FUNCTION

FUNCTION BINminusBIN$ (B1$, B2$)
  BINminusBIN$ = BIN$(BINVAL&(B1$) - BINVAL&(B2$))
END FUNCTION

FUNCTION BinNOT$ (Z&)
  BinNOT$ = BIN$(NOT Z&)
END FUNCTION

FUNCTION BinOR$ (Z1&, Z2&)
  BinOR$ = BIN$(Z1& OR Z2&)
END FUNCTION

FUNCTION BINplusBIN$ (B1$, B2$)
  BINplusBIN$ = BIN$(BINVAL&(B1$) + BINVAL&(B2$))
END FUNCTION

FUNCTION BINVAL& (Ziffern$)
  Z$ = NurBIT$(Ziffern$)
  Minus = (LEFT$(Z$, 1) = "1")
  FOR m = 16 TO 2 STEP -1
    IF MID$(Z$, m, 1) = "1" THEN
      B& = B& + 2 ^ (16 - m)
    END IF
  NEXT
  IF Minus THEN B& = -32767 + B&
  BINVAL = B&
END FUNCTION

FUNCTION BinXOR$ (Z1&, Z2&)
  BinXOR$ = BIN$(Z1& XOR Z2&)
END FUNCTION
```

```
FUNCTION BitZuBIN$ (D$)
  Merk$ = "&B"
  D$ = NurBIT$(D$)
  FOR m = 1 TO LEN(D$)
    IF INSTR("01", MID$(D$, m, 1))  0 THEN
      Merk$ = Merk$ + MID$(D$, m, 1)

    END IF

    IF m = 8 THEN Merk$ = Merk$ + " "
  NEXT
  BitZuBIN$ = Merk$
END FUNCTION

SUB DemoBinaer
GOSUB Kopf
DO
  LOCATE AZeile, ASpalte
  PRINT SPACE$(80 - ASpalte);
  LOCATE AZeile, ASpalte
  INPUT "", ant$
  IF UCASE$(ant$) = "Q" THEN EXIT DO
  IF BZeile > 24 THEN GOSUB Kopf
  Zahl& = VAL(ant$)
  B$ = BIN$(Zahl&)
  LOCATE BZeile, 1
  PRINT BINVAL&(BIN$(Zahl&)), "  ";
  PRINT B$,
  PRINT BINKompl$(BIN$(Zahl&)); "  ";
  A$ = HEX$(Zahl&)
  A$ = "&H" + STRING$(LEN(A$) MOD 2, "0") + A$
  PRINT A$;
  BZeile = CSRLIN + 1
LOOP WHILE UCASE$(ant$) <> "Q" AND ant$ <> CHR$(27)
EXIT SUB

Kopf:
  COLOR 15, 2
  CLS
  PRINT "Ende mit <Q>"
  PRINT "Bitte <Q> oder Zahl eingeben:";
  AZeile = CSRLIN
  ASpalte = POS(0)
  PRINT " Zahl              Bin_r                    Komplement       Hex";
  PRINT STRING$(80, "-")
  BZeile = CSRLIN
RETURN
END SUB
SUB DemoBitOperation
DO COLOR 0, 7
  CLS
  PRINT "Ende mit <Q>"
```

```
   INPUT "Bitte erste Zahl eingeben:", ant$
   IF UCASE$(ant$) = "Q" THEN EXIT DO
   Zahl1& = VAL(ant$)
   INPUT "Bitte zweite Zahl eingeben:", ant$
   IF UCASE$(ant$) = "Q" THEN EXIT DO
   Zahl2& = VAL(ant$)
   CLS
   B1$ = BIN$(Zahl1&)

  B2$ = BIN$(Zahl2&)
   PRINT "Zahl 1 =          ";
   PRINT BINVAL&(BIN$(Zahl1&)),
   PRINT B1$
   PRINT "Zahl 2 =          ";
   PRINT BINVAL&(BIN$(Zahl2&)),
   PRINT B2$
   PRINT
   PRINT "Zahl1 + Zahl2     "; BINVAL&(BINplusBIN$(B1$, B2$)), BINplusBIN$(B1$,
     B2$)
   PRINT "Zahl1 - Zahl2     "; BINVAL&(BINminusBIN$(B1$, B2$)), BINminusBIN$(B1$,
     B2$)
   PRINT "Zahl1 * Zahl2     "; BINVAL&(BINmalBIN$(B1$, B2$)), BINmalBIN$(B1$, B2$)
   PRINT "Zahl1 \ Zahl2     "; BINVAL&(BINdurchBIN$(B1$, B2$)), BINdurchBIN$(B1$,
     B2$)

   PRINT "NOT Zahl1         "; BINVAL&(BinNOT$(Zahl1&)), BinNOT$(Zahl1&)
   PRINT "NOT(NOT Zahl1)    "; BINVAL&(BinNOT$(NOT Zahl1&)), BinNOT$(NOT Zahl1&)
   PRINT "Zahl1 AND Zahl2   "; BINVAL&(BinAND$(Zahl1&, Zahl2&)), BinAND$(Zahl1&,
Zahl2&)
   PRINT "Zahl1 OR  Zahl2   "; BINVAL&(BinOR$(Zahl1&, Zahl2&)), BinOR$(Zahl1&,
Zahl2&)
   PRINT "Zahl1 XOR Zahl2   "; BINVAL&(BinXOR$(Zahl1&, Zahl2&)), BinXOR$(Zahl1&,
Zahl2&)
   PRINT "Zahl1 IMP Zahl2   "; BINVAL&(BinIMP$(Zahl1&, Zahl2&)), BinIMP$(Zahl1&,
Zahl2&)
   PRINT "Zahl1 EQV Zahl2   "; BINVAL&(BinEQV$(Zahl1&, Zahl2&)), BinEQV$(Zahl1&,
Zahl2&)
   PRINT : PRINT : PRINT

   PRINT "Auf Drucker ausgeben <D>  Abbruch <Q> oder <Escape>"
   DO: ant$ = INKEY$: LOOP WHILE ant$ = ""
   IF UCASE$(ant$) = "D" THEN GOSUB Drucker
LOOP WHILE UCASE$(ant$) <> "Q" AND ant$ <> CHR$(27)
EXIT SUB
Drucker:
   LPRINT "Zahl 1 =         ";
   LPRINT BINVAL&(BIN$(Zahl1&)),
   LPRINT B1$
   LPRINT "Zahl 2 =         ";
   LPRINT BINVAL&(BIN$(Zahl2&)),
   LPRINT B2$
   LPRINT
```

```
LPRINT "NOT Zahl1          "; BINVAL&(BinNOT$(Zahl1&)), BinNOT$(Zahl1&)
  LPRINT "NOT(NOT(Zahl1))   "; BINVAL&(BinNOT$(NOT Zahl1&)), BinNOT$(NOT Zahl1&)
  LPRINT "Zahl1 AND Zahl2   "; BINVAL&(BIN$(Zahl1& AND Zahl2&)), BinAND$(Zahl1&,
Zahl2&)
  LPRINT "Zahl1 OR  Zahl2   "; BINVAL&(BinOR$(Zahl1&, Zahl2&)), BinOR$(Zahl1&,
Zahl2&)
  LPRINT "Zahl1 XOR Zahl2   "; BINVAL&(BinXOR$(Zahl1&, Zahl2&)), BinXOR$(Zahl1&,
Zahl2&)
  LPRINT "Zahl1 IMP Zahl2   "; BINVAL&(BinIMP$(Zahl1&, Zahl2&)), BinIMP$(Zahl1&,
Zahl2&)
  LPRINT "Zahl1 EQV Zahl2   "; BINVAL&(BinEQV$(Zahl1&, Zahl2&)), BinEQV$(Zahl1&,
Zahl2&)
  LPRINT : PRINT : PRINT
RETURN
END SUB

SUB MiniMenu
  DO
    St$ = STRING$(50, 196)
    COLOR 15, 1
    CLS
    LOCATE 7
    PRINT TAB(10); "        Demoprogramm aus Chip-Special"
    PRINT TAB(10); St$
    PRINT TAB(10); "        Rechnen mit Bits"
    PRINT TAB(10); St$
    PRINT TAB(10); "1       Binärwerte ausgeben"
    PRINT
    PRINT TAB(10); "2       Bitoperationen"
    PRINT TAB(10); St$
    PRINT TAB(10); "        Wahl mit <Ziffer>  Ende mit <ESC> oder <Q>"
    PRINT TAB(10); St$
    DO: ant$ = INKEY$: LOOP WHILE ant$ = ""
    IF ant$ = CHR$(27) OR UCASE$(ant$) = "Q" THEN EXIT DO
    A = VAL(ant$)
    ON A GOSUB Wahl1, Wahl2
  LOOP
 EXIT SUB
Wahl1:
  DemoBinaer
RETURN
Wahl2:
  DemoBitOperation
RETURN
END SUB
FUNCTION NurBIT$ (D$)
  Merk$ = ""
  I = INSTR(D$, "+")
  IF I > 0 THEN D$ = LEFT$(D$, I - 1)
  D$ = RTRIM$(D$)
  FOR m = 1 TO LEN(D$)
```

```
    IF INSTR("01", MID$(D$, m, 1)) > 0 THEN
      Merk$ = Merk$ + MID$(D$, m, 1)
    END IF
  NEXT
  L = LEN(Merk$)
  IF L < 16 THEN Merk$ = STRING$(16 - L, "0") + Merk$
  IF L > 16 THEN Merk$ = RIGHT$(Merk$, 16)
  NurBIT$ = Merk$
END FUNCTION

FUNCTION ShiftL$ (D$, Stellen%)
  ShiftL$ = NurBIT$(NurBIT$(D$) + STRING$(Stellen%, "0"))
END FUNCTION

FUNCTION ShiftR$ (D$, Stellen%)
  ShiftR$ = LEFT$(STRING$(Stellen%, "0") + NurBIT$(D$), 16)
END FUNCTION
```

Die Funktionen BIN$ und BINVAL

Wie man Computer programmiert

Dieser Abschnitt richtet sich an den Leser, der noch keine Erfahrung im Programmieren seines Rechners hat. In kurzen Einführungen wird der Umgang mit Pseudocode und das Programmieren in BASIC dargestellt.

Im Abschnitt "Schlüssel zur Computertechnik" haben wir aufgezeigt, daß vom Rechner nur binär (Werte 0 und 1) codierte Programme direkt "verstanden" werden. Alle anderen Codierungen, ob im maschinennahen Assembler oder in einer Hochsprache wie BASIC geschrieben, müssen dem Rechner übersetzt werden. Diese Übersetzung kann auf zwei unterschiedliche Arten erfolgen, deren Besonderheiten wir in der Tabelle "Programmiersysteme" zusammengefaßt haben. Der Interpreter, das interpretierende Programmiersystem, übersetzt die Programmbefehle Zeile für Zeile in dem Augenblick, in dem das unter der Programmieroberfläche gestartete Programm die Zeile erreicht. Wird diese Zeile im Programm mehrmals angesteuert, erfolgt auch die Übersetzung jeweils neu. Diese Abarbeitungsart erfordert deshalb einen relativ hohen Zeitaufwand während des Programmlaufs. Das ist anders beim Compiler. Der Compiler übersetzt das Programm zunächst einmal komplett in Maschinencode. Hier ist gegenüber dem Interpreter, der das Programm direkt starten läßt, eine gewisse Zeit für den Compiliervorgang erforderlich. Erst dann wird das Programm abgearbeitet.

Programmiersysteme

Aber da jetzt keine Übersetzung mehr erforderlich ist, läuft das compilierte Programm wesentlich schneller ab als das Programm im Interpreter. Die meisten modernen Programmiersysteme mit Compiler arbeiten mit einer quasi-compilier-Vorstufe. Die Programmzeilen werden schon in einen maschinennahen Code übersetzt.

Bei Microsoft wird dies Vorcodierung (in QBASIC, QuickBASIC, Visual Basic für DOS und Visual Basic für Windows "threaded P-Code" (deutsch etwa: Aufgefädelter Programmcode) genannt.

Dadurch können auch Programmteile unter einem Compiler so getestet werden wie sonst nur bei einem Interpreter.

Interpreter	Nachteile:	Übersetzt Zeile für Zeile, Befehl für Befehl immer neu Programme brauchen immer die Entwicklungsoberfläche
	Vorteile:	Direktstart ohne Übersetzungszeit Programmtests können auch mit Programmteilen erfolgen.
Compiler	Nachteil:	Benötigt am Anfang Übersetzungszeit
	Vorteile:	Reduzierter Zeitbedarf beim Ablauf Programme laufen auch ohne die Oberfläche, eventuell mit einem sogenannten Laufzeitmodul.

Hochsprache oder Pseudocode

Will man die gleichen Programme für die unterschiedlichen Programmiersprachen schriftlich darstellen, kann man auf zweierlei Art vorgehen:

1. Programmdarstellung parallel in den unterschiedlichen Programmiersprachen
2. Programmdarstellung in Pseudocode

Beide Vorgehensweisen haben Vor- und Nachteile. Wir haben in unserer folgenden Tabelle die wesentlichen Merkmale beider Vorgehensweisen dargestellt.

Codierung in:	Merkmale
Progr.-Sprache	Vorteile: Alle Feinheiten wie Variablendeklaration, besondere Befehle können erfaßt werden. Das Wissen des Programmentwicklers über die jeweilige Hochsprache kann direkt vermittelt werden. Die Programme können so, wie sie dastehen, übernommen werden. Nachteile: Algorithmen sind oft nicht auf Anhieb zu erkennen. Programmcode muß bei paralleler Betrachtung für unterschiedliche Sprachen mehrfach, zum Teil nur minimal unterschiedlich, dargestellt werden.
Pseudocode	Vorteile: Programmcode für unterschiedliche Hochsprachen ist gleich und muß nur einmal dargestellt werden. Der wesentliche Algorithmus wird transparenter und damit für den Programmierer unabhängig von der Sprache direkt verwendbar. Nachteile: Keine direkte Verwendung des Codes möglich. Der "übersetzende" Programmierer muß die Feinheiten der Programmiersprache selbst kennen.

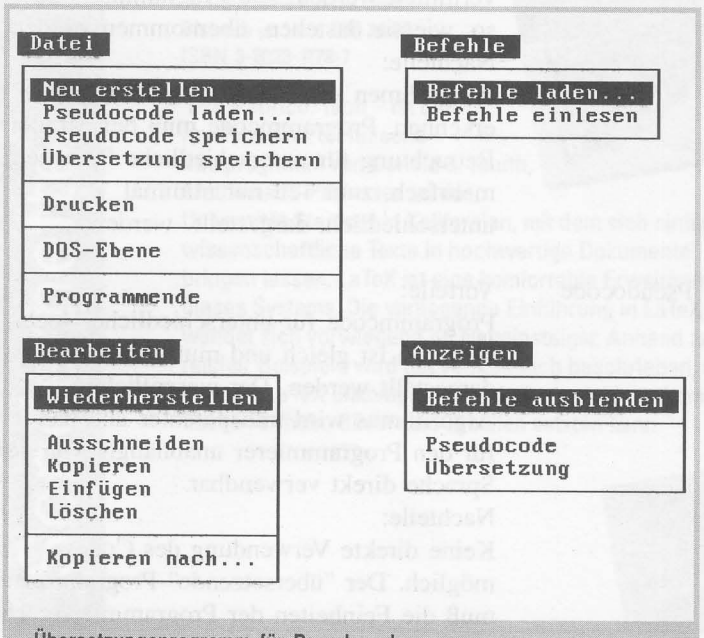
Das Übersetzungsprogramm für Pseudocode

Wir haben die schon immer vorhandene Idee mit dem Pseudocode in diesem Heft aufgegriffen. Mit angeregt wurde dies durch das von Wolfgang Link in seinem Buch "Messen, Steuern, Regeln mit PCs" verwendete. Dabei sind wir einen Schritt weiter gegangen. Auf der Diskette zu diesem Heft finden Sie unter dem Dateinamen "BASICTRS.EXE" ein kompiliertes QuickBASIC-Programm, das Sie unter DOS starten können. Sie können es natürlich auch unter Windows in einem Fenster laufen lassen. Das Programm übersetzt den Pseudocode in BASIC-Code. Um diesen Code in möglichst vielen BASIC-Varianten verwenden zu können, wird im Pseudocode in diesem Heft auf Programmteile zur Bildschirmgestaltung verzichtet.

Durch das Programm wird der Pseudocode in eine beliebig durch die Befehlsbibliothek bestimmbare Programmiersprache übersetzt und als ASCII-Programmfile abgespeichert. Es kann danach problemlos unter jeder BASIC-Oberfläche in die eigenen Programme eingebunden werden. Das Übersetzungsprogramm kann natürlich nicht alle Besonderheiten der jeweiligen Programmiersprache kennen. Aus diesem Grund erfolgt immer eine Basisübersetzung, die dann im integrierten Editor vor dem Abspeichern noch individuell bearbeitet werden kann.

Kleiner BASIC-Kurs

Ein Teil unserer Leser kennt BASIC noch nicht oder hat nur geringe Erfahrungen in dieser Programmiersprache. Für diese Leser ist der kleine BASIC-Kurs gedacht. Er vermittelt Ihnen, wenn Sie zu dem Kreis der Neulinge in BASIC gehören, die zum Verständnis der Programmiertechniken von BASIC erforderlichen Grundkenntnisse. Er kann und will nicht Ersatz für umfassendere Lehrbücher sein.



Übersetzungsprogramm für Pseudocode

Die Entwickler von BASIC, Kemeny und Kurtz, haben für die Befehlsbezeichnungen ihrer Programmiersprache die englische Sprache als Grundlage genommen. Ihnen ging es ja darum, eine auch dem Programmieranfänger verständliche Hilfe für das Computerprogrammieren anzubieten. Wenn Sie Englisch können, sind Ihnen deshalb viele der Begriffe aus dem allgemeinen Sprachgebrauch ableitbar.

BASIC-Programme laufen linear ab

Bis zum Erscheinen von DOS 5.0 gehörte zum Lieferumfang des Betriebssystems der BASIC-Interpreter GW_BASIC. Seit der Version DOS 5.0 ist es ein QBasic genanntes Programmiersystem, das auf dem erfolgreichen QuickBASIC basiert.

Wenn Sie sich zum Beispiel ein mit GW-BASIC erstelltes Programm ansehen, so finden Sie als Hinweis auf den linearen Programmablauf noch Zeilennummern vor jeder Programmzeile.

Ein solches Programm sieht beispielsweise so aus:

```
10  CLS
20  PRINT "Das Programm für die Namensausgabe"
30  PRINT "Bitte hier Ihren Namen eingeben :";
40  SPALTE = POS(0)
50  ZEILE = CSRLIN
60  PRINT: PRINT
70  PRINT " Der Name erscheint zwei Zeilen über der
    Eingabe"
80  INPUT"", NAME$
90  LOCATE ZEILE, SPALTE
100 PRINT NAME$ 110 END
```

Mit dem Programm wird zunächst der Bildschirm für Ein- und Ausgabe frei gemacht, dann werden Texte (in Anführungszeichen) ausgegeben, ein Name vom Benutzer eingegeben.

Dieses Programm enthält genau sieben unterschiedliche BASIC-Befehle, die alle versal, in Großbuchstaben, geschrieben sind.

Befehle sind in einer Programmiersprache geschützte Wörter mit eindeutiger Funktion, sogenannte Schlüsselwörter. Dabei können Befehle anweisen, bestimmte Aktionen zu veranlassen, die Anweisungen (Statements), oder sie können bestimmte Werte oder Daten zurückgeben, sogenannte Funktionen (Functions).

Vier der Befehle versteht man mit etwas Englischkenntnissen auf Anhieb:

PRINT=	Drucke (im übertragenen Sinn) auf dem Bild schirm
INPUT=	Eingabe, etwas eingeben
LOCATE=	etwas auf einen Platz bringen
END=	etwas (hier das Programm) beenden

Außerdem kann man sich vorstellen, daß in Anführungsstrichen stehender Text eben Text wie bei einer persönlichen Rede ist.

Die anderen Befehle sind Abkürzungen von englischen Bezeichnungen:

CLS= clear Screen = Bildschirm säubern
POS= Position
CSRLIN= Cursorline = Linie des Cursors

Zusätzlich zu diesen sechs Befehlen finden wir drei Bezeichnungen, bei denen die Worte sogar deutsch geschrieben sind:

SPALTE
ZEILE
NAME$

Diese drei Wörter sind sogenannte Variablen, vom Programmierer für dieses Programm so festgelegte (individuelle Schlüssel-)Wörter. Variablen dienen dazu, in einem Programm veränderliche Texte und Zahlen, also Daten, aufzunehmen.

Ihr Name wird vom Programmierer für das spezielle Programm festgelegt. Die Variablennamen können vom Programmierer individuell festgelegt werden. Sie haben nur in diesem Programm die einmal festgelegte Bedeutung.

Programmabläufe werden von Befehlen bestimmt

Wir haben jetzt die drei möglichen Bestandteile eines Programms herausgearbeitet:

Befehle
Variablen
Texte

Befehle und Variablen erscheinen in Großbuchstaben, der Text kann in beliebiger Schreibweise erscheinen. Sehen wir uns das Programm jetzt einmal zeilenweise an, so wie es abgearbeitet wird.

10 CLS
In dieser Zeile wird der Bildschirm "gereinigt", klar gemacht.

20 PRINT "Das Programm für die Namensausgabe"
Der Text "Das Programm für die Namensausgabe" wird "gedruckt" ausgegeben

30 PRINT "Bitte hier Ihren Namen eingeben:";
Der Text "Bitte hier Ihren Namen eingeben :" wird ausgegeben; Bei dieser Zeile ist das Semikolon am Ende von besonderer Bedeutung. Normalerweise springt der Cursor nach einer Ein- oder Ausgabe auf dem Bildschirm in die nächste Zeile. Hier wird durch das Semikolon erreicht, daß der Cursor in dieser Zeile bleibt.

40 SPALTE=POS(0)
Die Spalte des Cursors wird ermittelt und der Variablen SPALTE übergeben. (die Null steht hier wegen der Kompatibilität zu früheren Versionen, sie hat sonst keine Bedeutung)

50 ZEILE=CSRLN
Hier wird die Zeile des Cursors an die Variable ZEILE übergeben.

60 PRINT:PRINT
Mit dem ersten PRINT-Befehl (ohne Semikolon) wird in die nächste Zeile geschaltet. Das zweite PRINT erzeugt dann eine Leerzeile, es wird wieder in die nächste Zeile geschaltet.

70 PRINT "Der Name erscheint zwei Zeilen über der Eingabe"
Wieder wird Text "Der Name erscheint zwei Zeilen über der Eingabe" ausgegeben.

80 INPUT"", NAME$
Eine Eingabe wird vom Benutzer erwartet. Die Eingabe wird der Variablen NAME$ zugewiesen. Bei dieser Variablen steht am Schluß ein Dollarzeichen ($). Dieses Zeichen gibt an, daß die Variable eine Textvariable ist, alle über die Tastatur hier eingegebenen Zeichen werden als Text angesehen. Die Anführungszeichen nach INPUT und das Komma sind aus syntaktischen Gründen erforderlich. Mit dieser Zeichenkombination wird erreicht, daß das üblicherweise von der INPUT-Anweisung ausgegebene Fragezeichen unterdrückt wird.

90 LOCATE ZEILE, SPALTE
Der Cursor wird zurückbewegt in die mit den Variablen ZEILE und SPALTE bezeichnete Position auf dem Bildschirm.

100 PRINT NAME$
Dort wird der Inhalt der Variablen NAME$ ausgegeben.

110 END
Das Programm wird beendet

An diesem Programm lernt man schon einige für die Arbeit mit BASIC (und jeder anderen Programmiersprache) wichtige Regeln kennen:

Alle Teile eines Computerprogramms müssen entsprechend den Regeln der Programmiersprache gestaltet sein. Vorgeschrieben sind die genaue Schreibweise der Schlüsselwörter (Syntax) und deren vorgeschriebene Verwendungsart und Bedeutung (Semantik).

Ein Programm wird verändert

Es ist immer wieder erforderlich, Programme an neue Anforderungen anzupassen. Die Änderung und Ergänzung von Programmen ist bei dem mit Zeilennummern arbeiten-

den BASIC möglich, indem man im Programmeditor (siehe Editor) einfach eine Zeile mit der passenden Zeilennummer eingibt.

Wir wollen in unserem Falle dann, wenn kein Name eingegeben wurde, einen Signalton ausgeben und die erneute Eingabe fordern.

Dazu fügen wir folgende Zeilen in das Programm ein, indem wir sie einfach mit den Zeilennummern eingeben. Der Editor fügt sie dann automatisch an der richtigen Stelle in das Programm ein.

Unser Ziel ist es, zunächst zu prüfen, ob eine Eingabe erfolgt ist. Ist dies der Fall, soll die Ausgabe der eingegebene Text ausgegeben und das Programm beendet werden. Ist eine Eingabe ohne Textinhalt (man spricht auch von Leertext) erfolgt, so soll ein Signalton gegeben werden und dann die erneute Eingabe erfolgen.

Die entsprechenden Programmzeilen sind:

```
85 IF NAME$ = "" THEN GOSUB
200 GOTO 80 200 LOCATE 10, 20
210 PRINT "Bitte korrekte Eingabe! Weiter mit
    <Return>-Taste"
220 BEEP
230 INPUT "", ANT$
240 LOCATE 10, 20
250 PRINT STRING$(60, " ");
260 RETURN
```

Zu den uns bekannten BASIC-Befehlen kommen hier ein paar wichtige dazu. Auch sie sind mit Englischkenntnissen verständlich.

IF...THEN	Wenn...dann

Etwas wird ausgeführt, wenn eine Bedingung erfüllt ist. Man bezeichnet die IF-THEN-Anweisung deshalb auch als bedingte Anweisung im Gegensatz zu unbedingten Anweisungen wie PRINT, CLS.

GOSUB	Gehe zu Sub = Gehe in die Subroutine

Als Subroutine werden Programmteile bezeichnet, die mit der GOSUB-Anweisung angesprungen und die mit der RETURN-Anweisung (siehe unten) wieder verlassen werden. Wenn kein Text (die direkt aneinandergefügten Anführungszeichen) eingegeben wurde, wird also die Subroutine ab Zeile 200 angesprungen.

Wichtig ist, daß das Programm nach Rückkehr aus der Subroutine direkt nach der GOSUB-Anweisung fortfährt. Dies ist, nach einem Trennungszeichen (Doppelpunkt).

GOTO 80	Gehe zu (in unserem Falle zu Zeile 80)
LOCATE 10, 20	In der Subroutine wird zunächst der Cursor für die anschließende Textausgabe in Zeile 10 und Spalte 20 positioniert.
PRINT "..."	Dort wird der Text "Bitte korrekte Eingabe! Weiter mit <Return>-Taste"
BEEP	Ein Signalton wird ausgegeben.
INPUT "", ANT$	Dann wird eine Tastatureingabe in die Variable ANT$, also zumindest der Tastendruck auf die <Return>-Taste, verlangt.
LOCATE 10, 20	Der Cursor wird in Zeile 10, Spalte 20, also an der gleichen Stelle wie schon zuvor positioniert.
PRINT STRING$...	Ist der Tastendruck erfolgt, wird der zuletzt ausgegebene Text mit Leerzeichen überschrieben.

Ist die Bedingung dagegen erfüllt, ist also ein Text eingegeben, so geht das Programm ohne Umweg wie vorher zu Ende. Es ist in GW-BASIC also mit wenig Aufwand eine Änderung und Ergänzung von Programmen möglich. Die Abstände von zehn Zeilennummern haben hier den Vorteil, daß keine Umnummerierung erforderlich ist.

Die neuen BASIC-Dialekte

In den neuen BASIC-Dialekten (zum Beispiel QuickBASIC, QBASIC sowie Visual Basic für DOS und für Windows) sind auch die Zeilennummern nicht mehr erforderlich. Hier wird neuer Programmtext dadurch eingefügt, daß man an der entprechenden Stelle die Folgezeilen im Editor nach unten schiebt. Anstelle der Zeilennummern, die aus Gründen der Kompatibilität noch erlaubt sind, aber keine ordnende Funktion mehr haben, treten als Adressen für die Sprungbefehle Sprungmarken.
Es heißt dann nicht mehr

```
GOSUB 200
```

sondern beispielsweise

```
GOSUB  Subroutine1
```

und die Zeilennummer 200 wird durch die Sprungmarke Subroutine1: (beachten Sie den Doppelpunkt !) gekennzeichnet.

Die Befehle sind in den neuen Dialekten die gleichen wie vorher, nach Bedarf ergänzt und erweitert. Aus diesem Grund sind GW-BASIC-Programme ohne große Änderungen auch in den anderen Microsoft-BASIC-Versionen ver-

wendbar. Nur Visual Basic macht hier eine Ausnahme. Bei GW-BASIC sind zur besseren Gliederung eines Programms Teile, auf die von unterschiedlichen Programmstellen und mehrfach zugegriffen wird, in sogenannten Subroutinen zusammengefaßt.

Diese werden mit dem GOSUB-Befehl angesprungen und mit RETURN wieder verlassen. Mit den neuen Dialekten wird das prozedurale Programmieren auch unter BASIC realisiert. Darunter versteht man die Möglichkeit, Programmteile in in sich abgeschlossenen Prozeduren zusammenzufassen.

Diese Prozeduren werden durch ihren Namen aufgerufen. Es wird zwischen Sub-Prozeduren und Function-Prozeduren unterschieden. Der Unterschied liegt vor allem darin, daß Function-Prozeduren in ihrem Namen einen Wert zurückgeben können. Dieser Name hat eine dem Variablennamen vergleichbare Wirkung: Er ist ein Zeiger auf eine bestimmte Adresse im Speicher.

Dort wird zum Beispiel das Ergebnis einer Berechnung in der Function-Prozedur dann plaziert, wenn dieses Ergebnis in der Prozedur an den Prozedurnamen übergeben wurde. Damit sind Function-Prozeduren genau wie Variablen verwendbar. Sub-Prozeduren dagegen können nur aufgerufen werden. Ihre Namen sind keine Variablen-Zeiger, sie geben deshalb auch kein Ergebnis zurück.

Eine Sub-Prozedur hat folgende Struktur:

```
Sub NamenEingeben (X$)
   ...    'Hier steht der eigentliche Programmtext
End Sub
```

Nach dem Prozedurnamen steht in einer Klammer ein Parameter als Variablenname (X$). Dies ist der Platz, an den ein Wert oder ein Variableninhalt übergeben werden kann.

Der Aufruf dieser Sub-Prozedur könnte beispielsweise aussehen wie die zweite Zeile des nachfolgenden Programmtextes.

N$ = "Mein Name"	Der Variablen N$ wird der Text zugewiesen.
NamenEingeben N$	Die Prozedur wird angesprungen und dabei N$ als Argument an den Parameter X$ der Prozedur übergeben.
Print N$	Beim Rücksprung aus der Prozedur ist jetzt N$ der Parameter, der den Inhalt der eventuell geänderten Variablen X$ übergeben bekommt.

Bitte beachten Sie: Beim Aufruf einer Sub-Prozedur steht das Argument nach dem Prozedurnamen nicht in einer Klammer. Die Variable N$ übergibt ihren Inhalt an den Parameter X$ in der Prozedur. Diesen Inhalt nennt man Argument. Wird ein Argument als Variable übergeben, dann wird, wenn deren Inhalt sich in der Prozedur ändert, die Adresse zu diesem geänderten Inhalt an die aufrufende Stelle zurückgegeben.

Damit ändert sich dieser Wert auch für alle Programmzeilen, die nach dem Prozeduraufruf angesteuert werden. In der Print-Zeile würde dann der geänderte Inhalt auf dem Bildschirm ausgegeben.

Eine Function-Prozedur sieht zunächst ähnlich aus wie eine Sub-Prozedur.

```
Function Pi( )
   Pi = 4 * Atn( )
End Function
```

Function-Prozeduren haben aber wie oben dargelegt eine andere Funktion als Sub-Prozeduren. Der Aufruf erfolgt deshalb im Rahmen eines Ausdrucks. Dieser Aufruf könnte zum Beispiel so aussehen:

```
Print Pi( )
```

Bitte beachten Sie: Beim Aufruf einer Function-Prozedur muß dem Namen immer eine Klammer folgen, auch wenn gar kein Argument übergeben wird.

Mehrmodulige Programme

Die Beschränkung der älteren BASIC-Versionen auf die 64-KByte-Grenze ist bei modular arbeitenden neueren Versionen (QuickBASIC, beide Visual Basic) aufgehoben. Das neue Visual Basic für Windows (Version 2.0) durchbricht zusätzlich auch die Grenzen bei den Größen der in den Programmen verarbeitbaren Textdateien.

Als Module bezeichnet man Programmteile, die getrennt abgespeichert werden können. In jedem Modul kann nun Programmtext stehen. Damit dieser Text aber im ganzen Programm, auch als Projekt bezeichnet, verfügbar und bekannt ist, müssen alle Prozeduren deklariert werden.

Bei mehrmoduligen Programmen gibt es zwei Arten der Deklaration der in den Modulen enthaltenen Prozeduren. Bei QuickBASIC und Visual Basic für DOS müssen alle Prozeduren in den Modulen deklariert sein, aus denen heraus sie aufgerufen werden.

Erst wenn sie deklariert sind, sind Prozeduren, die außerhalb des aktuellen Moduls stehen, in diesem bekannt. Die Deklaration erfolgt bei den beiden BASIC-Versionen zum Teil auch automatisch durch die Benutzeroberfläche.

Bei Visual Basic für Windows (und für DOS) gibt es drei unterschiedliche Typen von Modulen:

Global Modul — In diesem werden im ganzen Projekt geltende Variablen und Konstanten mit dem Befehl Global ... deklariert.

Form-Modul — (bei Microsoft nur Form genannt) Diese Module sind die einzigen, die eine Ausgabe auf dem Bildschirm erlauben. Sie bestehen aus zwei verknüpften Teilen, einer Form als Bildschirmelement und dem dazugehörenden Programmcode. Der Programmcode kann in der Deklarationsebene (Generell-Ebene), in Ereignisprozeduren, die auf Ereignisse wie einen Mausclick reagieren oder in benutzerdefinierten Sub- oder Function-Prozedur en stehen.

Benutzerdef. Modul — Die Prozeduren dieses Moduls sind (bei Visual Basic für Windows) ohne zusätzliche Deklaration im ganzen Projekt bekannt. Soviel zur allgemeinen Einführung in die BASIC-Programmierung.
Wir haben darauf verzichtet, gleichfalls eine Einführung in (Turbo) Pascal einzufügen, da dies einfach den Rahmen dieses Heftes sprengen würde. Auf das Programmieren von fischertechnik-Steuerprogrammen mit dem Lucky-Logic-System gehen wir später noch detailliert ein.

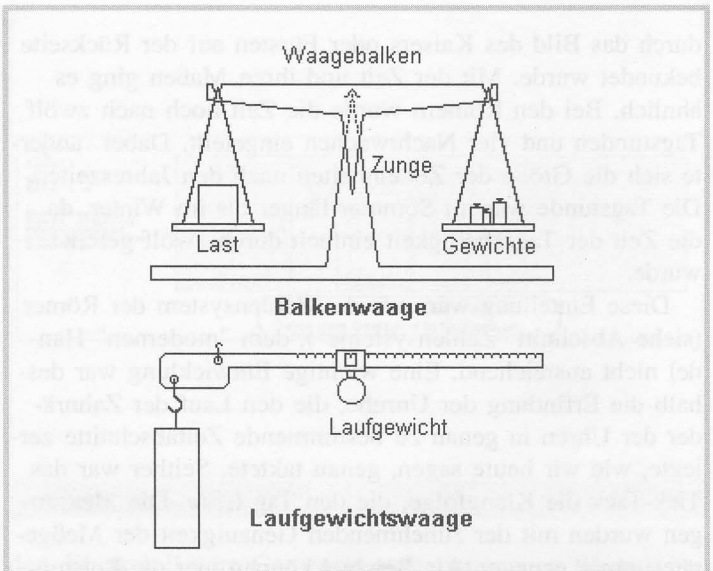
Signale in Informationen umsetzen

Den fischertechnik-Proficomputing-Baukasten kann man auch als völliger Laie im Bereich der Steuer- und Regeltechnik dazu benutzen, die im Handbuch dargestellten Modelle zu bauen und zu steuern. Man kann das dann als eine etwas anspruchsvollere Spielerei ansehen und betreiben. Aber der Baukasten ist ebenso wie der fischertechnik-Profi-Sensorikbaukasten nicht nur zum Betrieb der jeweiligen Modelle geeignet. Wie wir im Abschnitt "Fuzzy Logic ..." an dem Beispiel der Verwendung der Turtle sehen, kann man auch durchaus anspruchsvollere Experimente mit den Modellen betreiben. Aber dafür benötigt man zumindest ein gewisses Grundlagenwissen über die Vorgänge beim Messen, Steuern und Regeln. Mehr darüber finden Sie in diesem Abschnitt.

Am Anfang auch dieses Beitrags sollen zunächst ein paar wichtige Begriffe in Kurzform erklärt werden. Wir sprechen vom Messen, Steuern und Regeln. Was ist das? Was ist vor allem der Unterschied zwischen Steuern und Regeln? Messen ist die Umsetzung von Sensorsignalen in Informationen. Als Steuern bezeichnet man die Ingangsetzung und Beeinflussung (über Stellglieder) eines Ablaufs ohne Rückmeldung. Regeln dagegen bezeichnet den gleichen Vorgang wie beim Steuern mit dem Unterschied, daß Informationen vom Ablauf dazu benutzt werden können, über die entsprechenden Stellglieder eine Korrektur des Ablaufs vorzunehmen, Regeln ist also eine Steuerung mit "Feedback".

Messen

Die ersten "Meßgeräte" der Menschheit waren um 5000 v.Chr. in Ägypten die Balkenwaagen, mit denen die ersten Händler ihre Waren abwogen. Diese Waagenform ist uns heute noch erhalten mit der sogenannten Apothekerwaage (siehe "Die ersten Waagen"). Kurz danach müssen gleichfalls die Ägypter die ersten Sonnen-"Uhren" entwickelt haben, bei denen man an der Veränderung der Länge des Schattens eines in die Erde gesteckten Stabes die Zeit ermitteln konnte. Nach und nach wurden diese Meßinstrumente verfeinert und weitere dazu gefunden und erfunden. So wurde aus der Sonnenuhr, deren Schattenlänge das



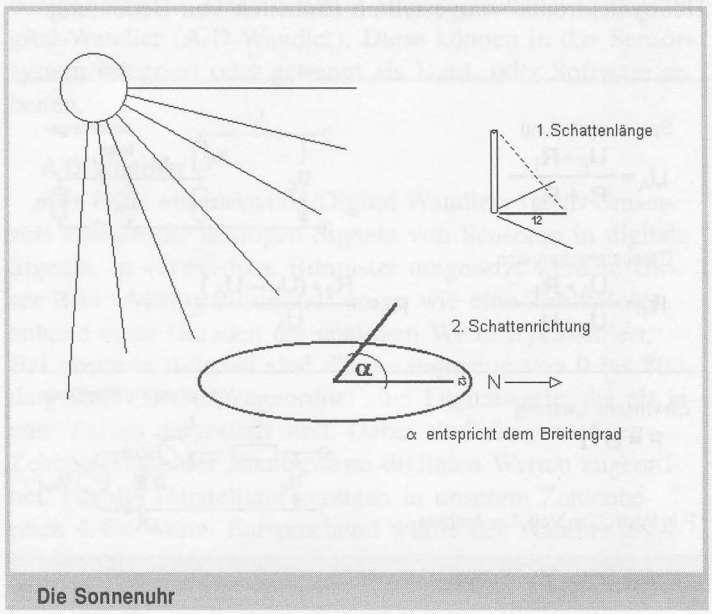
Die ersten Waagen

Maß der Zeit war, die Sonnenuhr, bei der die Richtung des Schattens die Zeit angab (siehe "Die Sonnenuhr").

Heute haben wir Meßgeräte für die unterschiedlichsten Meßbereiche. Sie arbeiten alle mit dem Grundprinzip des Vergleichs. Es wird eine zu messende Größe mit einer Maßgröße in Bezug gesetzt. Um zu gleichen Ergebnissen zu kommen, müssen die Maßeinheiten standardisiert sein.

Maßeinheiten

Mit der Entwicklung der Meßgeräte lief die Festlegung der dazu gehörenden Maßeinheiten parallel. Zu Anfang wandte jeder einzelne seine eigenen Körpermaße an, maß jeder mit seiner "persönlichen Elle". Aber von der individuellen Elle, dem Maß vom Ellenbogen bis zu den Fingerspitzen kam man bald zur für einen bestimmten Ort und später für bestimmte Regionen festgesetzten Elle. Gewichte wurden ebenso "genormt", Münzen aus Edelmetallen erhielten ein festgelegtes Gewicht, dessen Einheitlichkeit



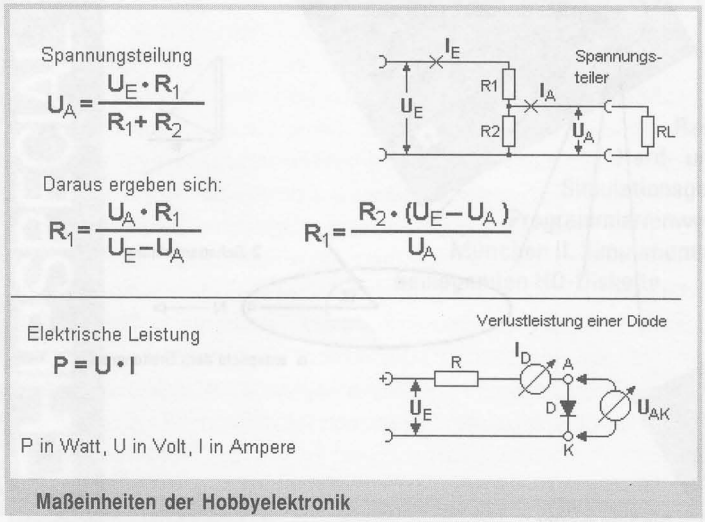
Die Sonnenuhr

durch das Bild des Kaisers oder Fürsten auf der Rückseite bekundet wurde. Mit der Zeit und ihren Maßen ging es ähnlich. Bei den Römern wurde die Zeit noch nach zwölf Tagstunden und vier Nachtwachen eingeteilt. Dabei änderte sich die Größe der Zeiteinheiten nach den Jahreszeiten. Die Tagstunde war im Sommer länger als im Winter, da die Zeit der Tageshelligkeit einfach durch zwölf geteilt wurde.

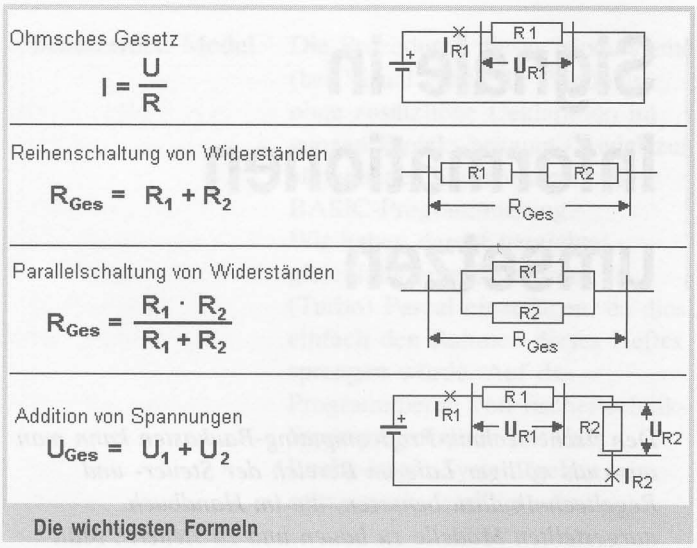
Diese Einteilung war, wie das Zahlensystem der Römer (siehe Abschnitt "Zahlensysteme"), dem "modernen" Handel nicht ausreichend. Eine wichtige Entwicklung war deshalb die Erfindung der Unruhe, die den Lauf der Zahnräder der Uhren in genau zu bestimmende Zeitabschnitte zerlegte, wie wir heute sagen, genau taktete. Seither war das Tick-Tack die Klangfolge, die den Tag teilte. Die Messungen wurden mit der zunehmenden Genauigkeit der Meßgeräte immer genauer. Als Beispiel können hier die Leistungen der Sportler bei Olympischen Spielen der Altzeit und der Neuzeit dienen. Bei den alten Griechen war man erster, zweiter oder dritter Läufer, der im Ziel ankam. Heute messen wir die Zeiten in Hundertstel-Sekunden, und man stellt sich die Frage, ob das noch sinnvoll und der Sache gerecht ist. Oder wollen wir künftig Entscheidungen im Sport nach Tausendstel-Sekunden treffen?

An diesem Beispiel zeigt sich eine wichtige Sicht allen Messens: Mit der Verfeinerung der Maßeinheiten, die sich aus den technischen Möglichkeiten der immer weiter verbesserten Meßgeräte ergeben, sollte man sich immer die Frage stellen, ob die gewählte Einheit der zu messenden Sache dem Sachverhalt im wahrsten Sinne des Wortes "angemessen" ist.

So könnte man als Elektronik-Amateur sich die tollsten wissenschaftlichen Meßgeräte zulegen. Nur, ist das sinnvoll? Wir werden weiter unten noch kurz auf die Meßgeräte eingehen, die Sie sich als Neuling anschaffen sollten, wenn Sie vertieft in die Materie der Meß-, Steuer- und Regelungstechnik einsteigen wollen. In der Hobbyelektronik sind insbesondere die in der Tabelle "Maßeinheiten der Hobbyelektronik" dargestellten Einheiten von Bedeutung.



Maßeinheiten der Hobbyelektronik



Die wichtigsten Formeln

Das Ohmsche Gesetz

"Bitte keine Theorie!" ... mag mancher Leser bei dieser Überschrift denken. Wir wollen das Thema deshalb auch nicht zu breit abhandeln. Aber wer sich wirklich ernsthaft mit der Materie elektrischen und elektronischen Messens beschäftigt, muß auch das Ohmsche Gesetz kennen. Mit ihm wird der Zusammenhang zwischen Spannung, Stromstärke und Widerstand sichtbar und berechenbar.

Das Ohmsche Gesetz ist nach dem deutschen Physiker Georg Simon Ohm benannt, der das Gesetz 1827 wie folgt formulierte:

"Die elektrische Stromstärke ist direkt proportional zur Potentialdifferenz und umgekehrt proportional zum Widerstand."

Das klingt für den Laien etwas kompliziert. Wesentlich transparenter ist da die Formel:

$I = U/R$

I ist die Strömstärke, gemessen in Ampere (A), U ist die Spannung gemessen in Volt (V) und R ist der Widerstand gemessen in Ohm (W), also Spannung geteilt durch Widerstand ergibt die Stromstärke.

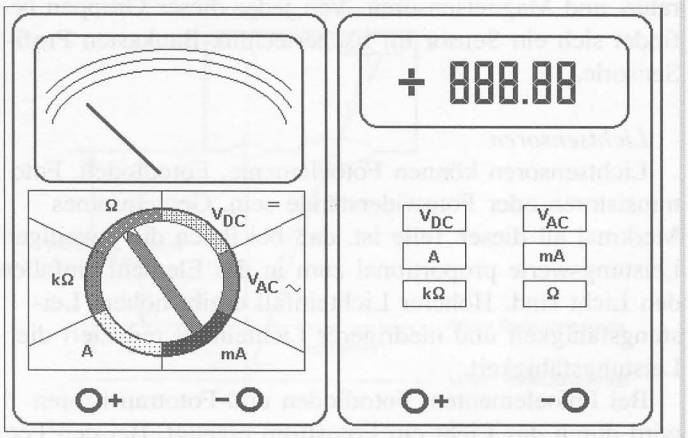
Wir haben in einer Tabelle die sechs wichtigsten Formeln zusammengefaßt, die der Hobbyelektroniker kennen sollte. Die symbolisierten Schaltungen zeigen jeweils dazu die Zusammenhänge und die Meßstellen.

Damit soll es hier mit der Theorie auch schon genug sein.

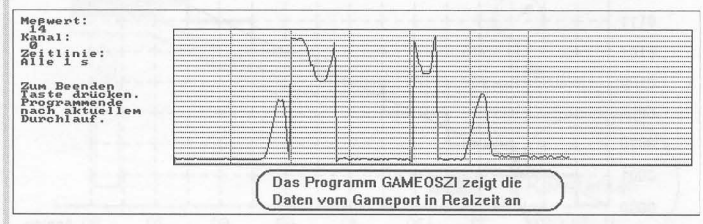
Meßgeräte

Die Darstellung von Meßwerten kann analog (Ablesen auf einer Skala), digital (mit Ziffernanzeige) oder grafisch erfolgen. Als Grundlage für die vom Amateur vorzunehmenden Messungen kann ein sogenanntes Vielfachmeßgerät dienen.

Wir haben in unserer Abbildung "Vielfachmeßgeräte" die zwei Typen von Geräten schematisiert dargestellt.



Vielfachmeßgeräte



GAMESOSZI in Aktion

Die Wiedergabe der Meßwerte kann analog über Drehzeiger oder digital über Ziffernanzeige erfolgen. Es ist aber wichtig zu wissen, daß auch Meßgeräte Toleranzbereiche haben, das heißt, in definierten Grenzen liegende Abweichungen vom angezeigten Wert. Das Digitalgerät zeigt zwar exakte Ziffernfolgen an, diese repräsentieren aber in der Regel einen Mittelwert. Wenn Sie sich mit dem Thema der Unschärfe auseinandersetzen möchten, sollten Sie auch den Abschnitt "Fuzzy Logic ..." durchlesen. Wählen Sie also den Gerätetyp, der Ihrer Arbeitsweise am meisten entgegenkommt.

Für die Arbeit im Elektronikbereich sollte Ihr Vielfachmeßgerät folgende Meßbereiche aufweisen:

Gleichspannung:	R = 50 kW Innenwiderstand mindestens 50 kW mind. fünfMeßbereiche ab ca. 1 - 2 V
Wechselspannung:	Meßbereiche ab etwa 3 V
Widerstand:	etwa vier Meßbereiche
Stromstärke:	etwa vier Meßbereiche ab 100 mA bis ca. 1 A

Der schon etwas geübtere Leser kann sich auch seine Meßgeräte selbst bauen. Hierfür gibt das Buch "Messen - was, wie, womit" aus dem Elektorverlag (näheres siehe unten) direkt verwertbare Anregungen. Interessant ist dort das Konzept, Meßgeräte modular zusammenzufügen.

Als eine wichtige Erweiterung der Meßdatenerfassung und Verarbeitung kann in einem nächsten Schritt die Anschaffung eines Oszilloskops folgen. Mit ihm lassen sich dann zum Beispiel Kennlinien von Transistoren und vieles andere mehr grafisch leichter aufarbeiten.

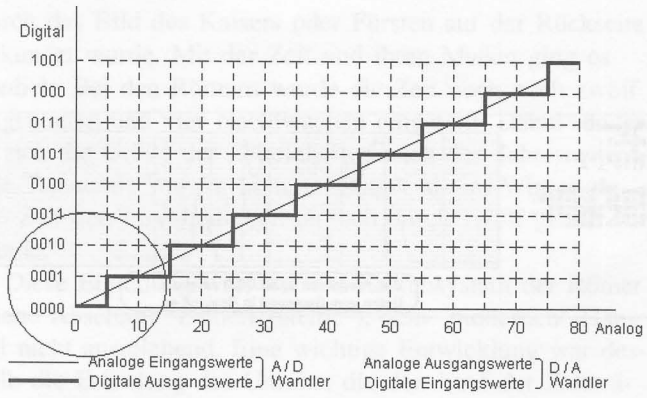
Die grafische Aufarbeitung kann natürlich auch mit Hilfe des Computers erfolgen. Auf unserer Diskette finden Sie ein Programm, das wir mit freundlicher Genehmigung des Markt&Technik-Verlags aus dem "PC-Bastelbuch" von Kai Hamann entnommen haben. Das Programm kann man auf einfache Weise den eigenen Anforderungen anpassen, da es auch im Quelltext vorliegt. In dem Programm werden die aktuell vom Gameport aufgenommenen Werte wie bei einem Oszilloskop auf den Bildschirm gebracht. Wenn Sie die im Programm gestellte Frage "Soll eine Meßdatenreihe vor Ausgabe beendet werden?" mit "N" beantworten, werden die vom Gameport eingelesenen Daten in Realzeit auf dem Bildschirm in eine Kurve übertragen. Unser Bild "GAMEOSZI in Aktion" zeigt einen solchen Ablauf.

Mit Sensoren messen

Physikalische Größen wie Temperatur, Licht (elektromagnetische Strahlung), Druck, Beschleunigung, chemische Größen (Zusammensetzung von Gasen und Flüssigkeiten) sowie elektrochemische Größen kann man mit Sensoren messen. Als Sensoren bezeichnet man Meßfühler, Meßwertgeber die Meßwerte in ein proportionales, das heißt, ein der zu messenden Größe entsprechendes elektrisches Signal umsetzen. Das elektrische Signal ist in der Regel ein analoges, das heißt, sich kontinuierlich änderndes Signal. Sensoren sind also Wandler, die Größen des zu messenden Systems in Analogsignale umsetzen. Die analogen Signale können aber in der Regel nicht direkt verarbeitet werden, sie müssen für die Verwendung im jeweiligen System aufbereitet werden. Dazu benötigt man sogenannte Analog/Digital-Wandler (A/D-Wandler). Diese können in das Sensorsystem integriert oder getrennt als Hard- oder Software arbeiten.

A/D-Wandler

Mit Hilfe eines Analog/Digital-Wandlers (auch Umsetzer) können die analogen Signale von Sensoren in digitale Signale, in verwertbare Bitmuster umgesetzt werden. Unser Bild "Analog zu digital" zeigt, wie eine Zahlenfolge anhand einer Geraden die analogen Werte repräsentiert. Bei unserem Beispiel sind die Analogwerte von 0 bis 80 dargestellt. Diesen zugeordnet sind Digitalwerte, die als binäre Zahlen dargestellt sind. Dabei sind die jeweiligen Zehnerschritte der Analogwerte digitalen Werten zugeordnet. Für die Darstellung genügen in unserem Zahlenbereich 4-Bit-Werte. Entsprechend würde der Wandler als 4-Bit-Wandler bezeichnet.



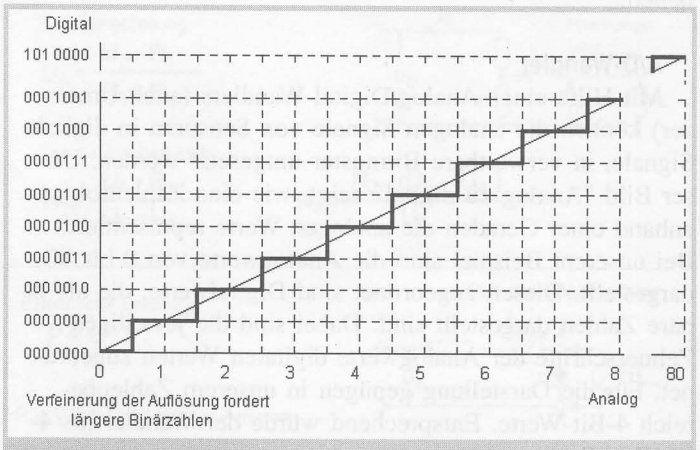
Analog zu digital

Da die durch die Treppenkurve dargestellten Binärwerte nur im jeweiligen Schnittpunkt der Geraden mit der Linie des jeweiligen Binärwertes den Analogwerten genau entspricht (in unserem Beispiel bei 10, 20 ...), ergibt sich eine als Auflösung bezeichnete Abstufung der exakt durch Binärwerte repräsentierten Analogwerte. Eine Erhöhung der Bitzahl ergibt somit eine höhere Auflösung. Die Abweichung der dazwischenliegenden Werte erreicht, wie leicht aus dem Bild "Höhere Auflösung" zu ersehen ist, maximal die Hälfte des niederwertigsten Bit (LSB = Least Significant Bit), das ja die Höhe der Treppenstufen bestimmt. Die Genauigkeit der digitalen Ausgabe eines A/D-Wandlers wird also durch die Auflösung bestimmt.

Außer der Auflösung und der damit verbundenen Abweichung der Digital- von den Analogwerten ist eine weitere Abweichungsmöglichkeit von großer Bedeutung. Für jede Operation, also auch für das Umwandeln von Analog- in Digitalwerte, wird eine bestimmte Zeit benötigt. Diese Zeit, die man Wandlungszeit (engl.: Conversion Time) nennt, beeinflußt bei Wandlungen die Genauigkeit des verarbeiteten Meßwertes.

Sensoren

Bei den Sensoren wollen wir uns in diesem Heft nur mit drei Gruppen beschäftigen, nämlich mit Licht-, Temperatur- und Magnetsensoren. Von jeder dieser Gruppen befindet sich ein Sensor im fischertechnik-Baukasten Profi-Sensoric.



Höhere Auflösung

Lichtsensoren

Lichtsensoren können Fotoelemente, Fotodioden, Fototransistoren oder Fotowiderstände sein. Gemeinsames Merkmal all dieser Teile ist, daß bei ihnen die jeweiligen Leistungswerte proportional zum in das Element einfallenden Licht sind. Höherer Lichteinfall ergibt höhere Leistungsfähigkeit und niedrigerer Lichteinfall reduziert die Leistungsfähigkeit.

Bei Fotoelementen, Fotodioden und Fototransistoren wird durch das Licht ein Fotostrom erzeugt. Bei den Fotoelementen kann wie bei einem Generator elektrische Leistung an einen Verbraucher abgegeben werden. Fotoelemente werden, da sie Licht in elektrische Energie umwandeln, zumeist als Solarzellen zur Stromversorgung eingesetzt. Fotodioden (vergleiche "Dioden") benötigen im Gegensatz zu Fotoelementen eine von außen angelegte Spannung.

Sperrstrom und Beleuchtungsstärke stehen in linearem Zusammenhang, weshalb Fotodioden zu Meßzwecken eingesetzt werden. Wird eine Fotodiode mit einer zusätzlichen Sperrschicht versehen, so hat man das Prinzip des Fototransistors. Da dieser (wie Transistoren) zusätzliche Verstärkerwirkung hat, ist ein Fototransistor wesentlich empfindlicher als eine Fotodiode.

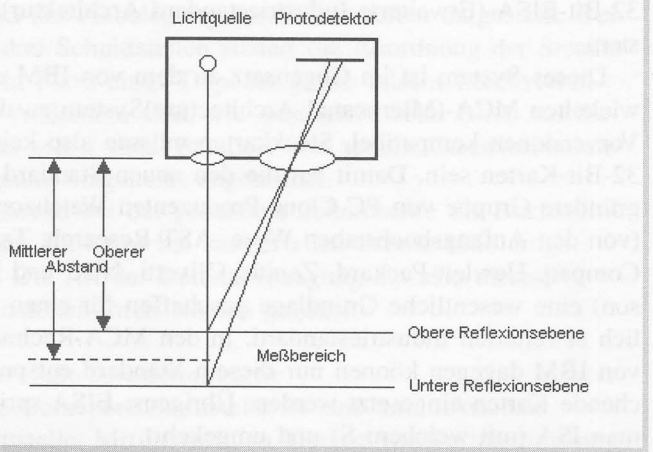
In den beiden in diesem Heft beschriebenen fischertechnik-Profi-Baukästen (Profi-Sensoric und Profi-Computing) finden Sie Fototransistoren als Lichtsensoren. Fotowiderstände sind Halbleiterelemente, deren elektrischer Widerstand bei Beleuchtung abnimmt. Sie benötigen eine Spannungsquelle.

In unseren zwei Bildern "Schichtdicke messen" und "Materialstärke messen" werden Fotodetektoren für die in den Untertiteln stehenden Aufgaben benutzt. Wir verwenden hier die allgemeinere Bezeichnung Fotodetektoren (siehe Fachbegriffe ...), da die Messungen zwar optoelektronisch, aber auf unterschiedliche Weise erfolgen können.

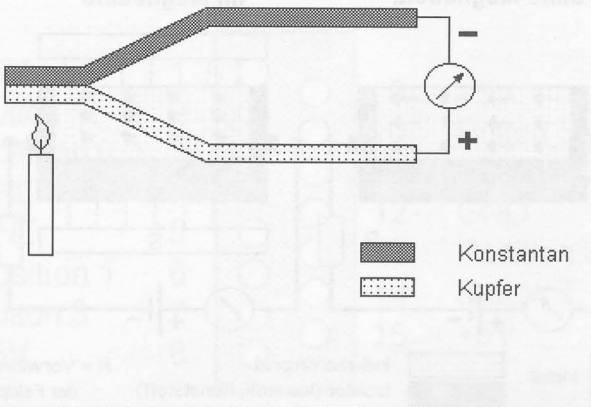
Die Schichtdickenmessung erfolgt so, daß ein Lichtstrahl von einer exakten Lichtquelle, die in einem genau definierten Abstand zur Oberfläche eines Werkstücks steht, senkrecht auf die zu messende (lichtdurchlässige) Schicht trifft.

Dabei erfolgen Reflexionen an den beiden Oberflächen, deren Strahlen über ein Linsensystem zu einem Fotodetektor gelangen. Über die im Detektor ermittelten Werte kann dann, da oberer Abstand und Winkel der einfallenden Strahlen als Werte vorliegen, im Wege der Dreiecksrechnung die Schichtdicke errechnet werden.

Die zweite Meßmethode nutzt denselben Typ von Fotodetektor, nur daß jetzt aus dem exakt festzustellenden Abstand zweier Detektoren und dem nach dem vorstehend beschriebenen System ermittelten Abstand jedes einzelnen Fotodetektors zur Materialoberfläche direkt der Differenzwert, also die Materialstärke, berechnet werden kann.



Schichtdicke messen



Thermoelektrische Fühler

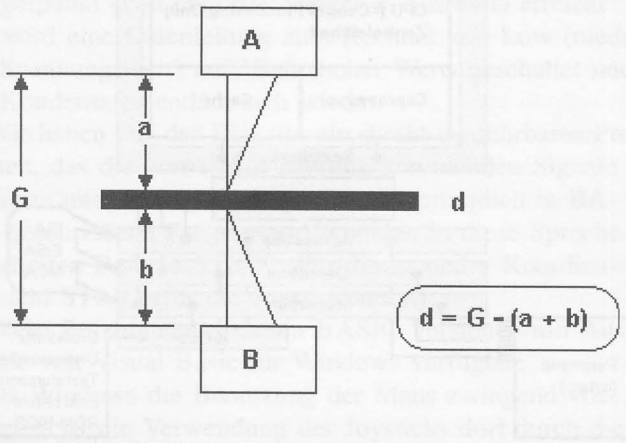
Temperatursensoren

Temperatursensoren erfassen und messen Temperaturen und Temperaturunterschiede. Temperatursensoren können auf unterschiedlichen physikalischen Reaktionen auf Temperaturunterschiede basieren.

Die Thermofühler, genauer gesagt, thermoelektrischen Fühler nutzen den Potentialunterschied, der entsteht, wenn zwei unterschiedliche Metalle miteinander fest verbunden sind, und ein Unterschied zwischen den Temperaturen der Verbindungs-(Schweiß-)stelle und den Metallen besteht. Einen Vertreter dieses Typus zeigen wir in dem Bild "Thermolelektrischer Fühler". Thermoelektrische Fühler werden vor allem in der Hochtemperaturmessung eingesetzt.

Typische Repräsentanten der Gruppe der Temperatursensoren sind die Heißleiter (NTC-Widerstände, siehe Fachbegriffe) und Kaltleiter (PTC-Widerstände) deren innerer elektrischer Widerstand in Abhängigkeit von der Umgebungswärme unterschiedliche Werte annehmen kann.

In den beiden Profi-Baukästen findet man je einen NTC-Widerstand. Die Nennwiderstände sind unterschiedlich. Bei Profi-Sensoric ist dies ein Widerstand mit 60 kW, im Profi-Computing ein solcher mit 1,5 kW.



Materialstärke messen

Infrarot-Wärmesensoren geben zwar als Meßwerte auch Temperaturwerte zurück, sie gehören aber zu den Lichtsensoren, da sie den unsichtbaren Infrarotanteil des Lichts für die Wärmemessung nutzen.

Magnetsensoren

Ein auch im Hobbybereich verwendeter Magentsensortyp ist die Feldplatte. Bei ihr wächst der Widerstand gegen einen an der Feldplatte angelegten Strom entsprechend einem einwirkenden Magnetfeld. Das Prinzip dieses Sensortyps zeigt unser Bild "Feldplatte als Magnetsensor".

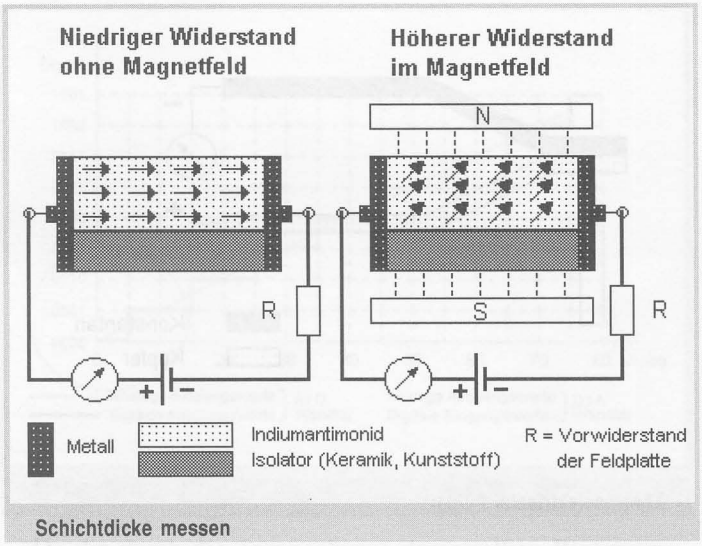
Im Abschnitt "Fachbegriffe ..." weiter unten finden Sie eine ausführliche Beschreibung der Hall-Sensoren und Reed-Kontakte, zweier Sensortypen, die gleichfalls auf magnetische Felder reagieren. Während beim Reed-Kontakt bei einer bestimmten Magnetfeldstärke ein Kontakt hergestellt wird, kann im Hall-Sensor an einer durch den Hall-Effekt erzeugten Spannungsdifferenz die Stärke des Magnetfeldes ermittelt werden. Im Profi-Sensoric-Baukasten finden Sie einen Reed-Kontakt, dessen Kontakt mit dem mitgelieferten Dauermagneten geschlossen werden kann.

Im Abschnitt "fischertechnik Profi-Sensoric" gehen wir auf Einzelheiten der Verwendung der Sensoren in den Modellen noch näher ein.

Die Datenübergabe

Wir hatten zu Beginn dieses Abschnitts die Begriffe Steuern und Regeln kurz definiert. Der Unterschied liegt in der Rückkopplung. Während beim Steuern vorgegebene Werte eingestellt und in immer gleicher Weise ohne Korrektur vom Prozeß her verwendet werden, werden bei der Regelung durch Meßgeräte gelieferte Daten aus dem laufenden Prozeß dazu benutzt, Korrekturen vorzunehmen. Da es bei uns um die Möglichkeiten der Steuerung und Regelung mit dem Computer geht, ist es von besonderer Bedeutung festzustellen, wie - bei der Regelung - die Daten vom Prozeß in den Rechner und vom Rechner in den Prozeß gelangen.

Voraussetzung dafür ist natürlich, daß relevante Daten aus dem Prozeß ermittelt und an den Rechner übergeben werden. Aus den Daten werden dann im Rechner Schluß-



Schichtdicke messen

folgerungen gezogen und das Ergebnis dieser Schlußfolgerungen an den Prozeß wieder zurückgegeben. Man bezeichnet die Stellen, wo Daten von einem Medium in das andere, von einem Gerät zu einem anderen oder vom Benutzer zum Gerät oder umgekehrt gelangen, als Schnittstellen oder mit dem englischen Fachterminus "Interface". So besteht zum Beispiel bei einem PC die Benutzerschnittstelle einerseits aus Tastatur und Maus für die Dateneingabe, andererseits aus Bildschirm und Drucker für die Datenausgabe. Im Rechner sorgen entsprechende Programme (Treiber) dafür, daß die ein- oder ausgehenden Signale in die jeweils andere "Sprache" übersetzt werden.

Wir wollen zunächst die Aufgabe näher betrachten, bei der es darum geht, Daten von einem Sensor oder ähnlichem an den Computer weiterzugeben. Eine der Voraussetzungen dafür, daß unser Rechner überhaupt etwas mit den Daten anfangen kann, die von außerhalb an ihn übermittelt werden, ist, daß beide die gleiche Sprache sprechen. Aus diesem Grund müssen Sensorsignale aus Analogwerten in digitale Signale umgewandelt werden. Das Prinzip haben wir schon oben bei der Beschreibung des A/D-Wandlers dargestellt. Die dann in Bit umgewandelten Analogwerte können über Datenleitungen an den Computer gesandt, diesem übergeben werden.

Die Erweiterungssteckplätze des PC

Im PC werden alle Verbindungen von der Hauptplatine mit dem Prozessor nach außen über die sogenannten Erweiterungssteckplätze geschaffen. Zum Verständnis des Steckplatzsystems soll unser Bild "Die Bussysteme des PC" dienen. Sie sehen darin, vereinfacht dargestellt, wie die Signale in einem Rechner über Bus genannte mehrpolige Leitungsverbindungen zwischen den einzelnen Komponenten eines Systems ausgetauscht werden.

Dabei stehen für den PC zur Zeit drei verschiedene Steckplatzsyteme zur Verfügung. Die Entwicklung der Bussysteme ging vom ursprünglichen 8-Bit-Synchronbus des ersten PC über das auch heute noch am weitesten verbreitete 16-Bit-ISA-(Industriestandard Architektur)System zum 32-Bit-EISA-(Erweiterte Industriestandard Architektur)System.

Dieses System ist im Gegensatz zu dem von IBM entwickelten MCA-(Microcanal Architecture)System zu den Vorversionen kompatibel, Steckkarten müssen also keine 32-Bit-Karten sein. Damit hat die den neuen Standard begründete Gruppe von PC-Clone-Produzenten Watchzone (von den Anfangsbuchstaben Wyse, AST Research, Tandy, Compaq, Hewlett-Packard, Zenith, Olivetti, NEC und Epson) eine wesentliche Grundlage geschaffen für einen wirklich erweiterten Industriestandard. In den MCA-Rechnern von IBM dagegen können nur diesem Standard entsprechende Karten eingesetzt werden. Übrigens: EISA spricht man ISA (mit weichem S) und umgekehrt.

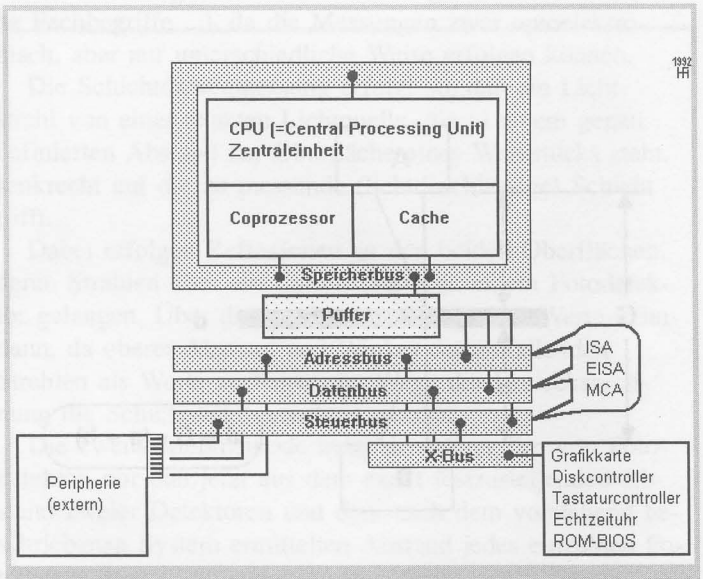
Die Schnittstellen des PC

Die Übergabe von Signalen kann nun parallel (mehrere Bits gleichzeitig) oder seriell (Bit nach Bit) erfolgen. Entsprechend stehen für die Übertragung zwei verschiedene Arten von Schnittstellen zur Verfügung, die parallele und die serielle Schnittstelle. Beide Schnittstellentypen sind normalerweise in einem PC vorhanden.

Typische parallele Schnittstellen sind z.B. die 25-polige Druckerschnittstelle, häufig auch Centronics-Schnittstelle genannt und der 15-polige Joystickanschluß. Die Druckerschnittstelle ist in der Regel gemeint, wenn man von der parallelen Schnittstelle spricht. Parallel ist bei ihr die Übergabe der Daten Byte für Byte auf acht Datenleitungen und der zusätzlichen Informationen über mindestens.

Als serielle Schnittstelle hat jeder Rechner heute primär für den Anschluß der Maus die ebenfalls 25-polige V24-(auch RS-232-)Schnittstelle. An diese Schnittstelle werden auch Modems oder andere Datenübertragungseinrichtungen angeschlossen.

Diese Schnittstelle überträgt die eigentlichen Daten seriell, also Bit für Bit nacheinander. Wir haben in unseren



Die Bussysteme des PC

Bildern die Pinbelegung der Schnittstellen dargestellt. Bei allen drei Schnittstellen sichert die Anordnung der Steckerstifte in Form eines Trapezes gegen falsche Steckverbindung. Außerdem sind, wie man leicht beim Blick auf die Stecker selbst sieht, parallele und serielle Steckverbindungen genau umgekehrt angeordnet.

Während bei der parallelen Schnittstelle die Buchse im Rechner ist, ist es bei der seriellen Schnittstelle umgekehrt. Die Art der Datenübertragung hat also diesen Schnittstellen ihren Namen gegeben.

Die Gameport-Schnittstelle

Die Gameportschnittstelle ist eine unidirektionale Schnittstelle. Mit ihr werden Signale von Peripheriegeräten (Joysticks) an den Computer gesandt. Wie der Name es schon sagt, ist diese Schnittstelle vor allem für die Übertragung von Daten im Spielbereich gedacht.

Die Schnittstelle ist deshalb zwar in den meisten Homecomputern vorhanden, gehört aber nicht zur Standardausstattung der PC. Man muß sie in der Regel nachrüsten. Dazu gibt es Schnittstellenkarten, die ohne Probleme in einen der Erweiterungssteckplätze auf der Hauptplatine des PC gesteckt werden können.

Die Belegung der Gameport-Schnittstelle zeigt unser Bild. Die für die Pins verwendeten Bezeichnungen ergeben sich direkt daraus. Die Abkürzung GND steht für Ground (deutsch: Grund, also Gerätemasse, Abschirmung). An den Gameport werden als Joystick bezeichnete Steuerknüppel angeschlossen, mit deren Hilfe normalerweise die Steuerung von Spielprogrammen erfolgt.

Die Joysticks der Homecomputer und der PC haben unterschiedliche Schaltungssysteme. Während bei den meisten Homecomputern digitale Joysticks angeschlossen werden, sind dies bei den PC analoge Joysticks.

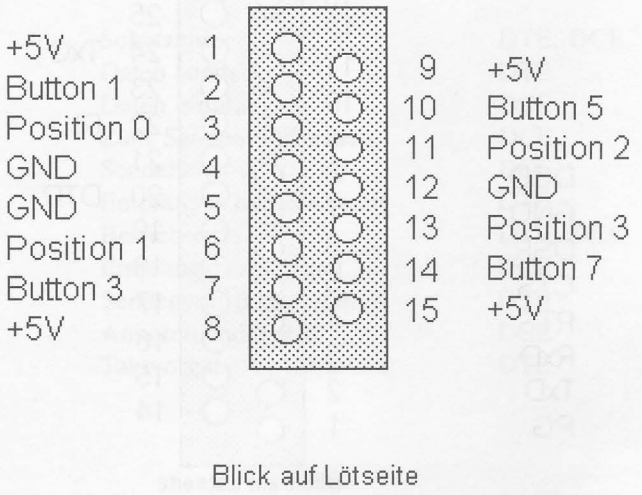
Diese haben, im Gegensatz zu den digitalen Joysticks, bei denen Einzelschalter durch die Bewegungen des Knüppels geschaltet werden, zwei Potentiometer (X-,Y-Position) integriert, die je nach Stellung des Knüppels den Stromfluß zu einem Kondensator beeinflussen. Dieser wird entsprechend schnell oder langsam aufgeladen.

Wenn die Ladung des Kondensators den sogenannten Triggerpunkt (von englisch: trigger = Auslöser) erreicht hat, wird eine Datenleitung zum Rechner von Low (niedriger Spannungswert) auf High (hoher Wert) geschaltet und der Kondensator entlädt sich wieder.

Wir haben auf der Diskette ein direkt ausführbares Programm, das die von einem Joystick kommenden Signale in Werte umsetzt. Dazu benutzt das ursprünglich in BASIC geschriebene Programm die beiden in diese Sprache integrierten Befehle STICK (Für die x- und y-Koordinaten) und STRIG (für die Joystickbutton).

Diese Befehle sind in allen BASIC-Versionen mit Ausnahme von Visual Basic für Windows verfügbar.

Da Windows die Benutzung der Maus zwingend vorschreibt, ist die Verwendung der Joysticks dort durch die Maus ersetzt. Der Gameport wird unter Windows nicht abgefragt.



Die Gameport-Schnittstelle

Die serielle Schnittstelle

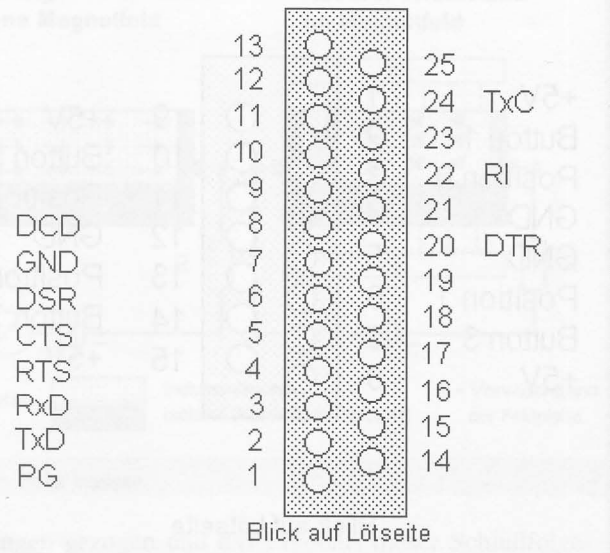
Ein früher Vorläufer unserer seriellen Datenübertragung war die Telegrafenleitung des Samuel Morse. Auch das Morsealphabet wird seriell übertragen. Zeichen setzen sich aus langen und kurzen Einzelsignalen, unseren Bits, zusammen. Die sich aus der Erfindung Morses entwickelnde Technik der Datenübertragung der Fernschreiber wird heute noch in der TTY-Schnittstelle (Teletype, deutsch: Fernschreiben) für Optokoppler genutzt.

Für unsere heutige serielle Schnittstelle werden zwei Bezeichnungen synonym verwendet: RS232 und V.24. Diese Bezeichnungen kommen von der amerikanischen RS232-Norm der EIA (Electronic Industrie Association) beziehungsweise der europäischen V.24-Norm des CCITT (Comite Consultatif International Telegrafique et Telefonique, Genf). Beide definieren das, was wir als serielle Schnittstelle kennen.

Die serielle Schnittstelle kann als 25-poliger oder auch als neunpoliger Subminiatur-D-Stecker vorliegen. Wie die beiden Pinzahlen schon zeigen, wird von den Pins der 25-poligen Schnittstelle nur eine geringer Teil genutzt. Wir haben in unseren Bildern die genutzten Leitungen und deren Bezeichnungen dargestellt. Dabei ist wichtig, daß, aus welchen Gründen auch immer, die beiden Pins 2 und 3 beim 25-poligen Stecker an TD und RD, beim neunpoligen aber genau umgekehrt an den Busleitungen liegen.

Für die Übertragung werden also nur wenige Leitungen genutzt. Die Kurzbezeichnungen dieser Leitungen, ihre englischen Namen und ihre Funktion ersehen Sie aus unserer Tabelle "Die benutzten seriellen Leitungen".

Die Daten werden bei der seriellen Schnittstelle nacheinander Bit für Bit übertragen. Dies hat den Vorteil, daß nur eine Datenleitung erforderlich ist, erfordert aber gegenüber der parallelen Übertragung höhere Übertragungszeiten. Für einen bidirektionellen Datentransfer genügt eine zweite Leitung. Die serielle Schnittstelle ist deshalb bisher allein ge-

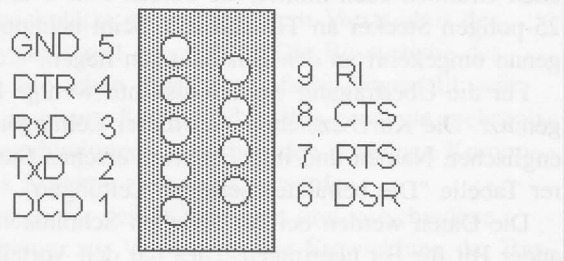


Die 25-polige serielle Schnittstelle

eignet zur Datenfernübertragung über die Telefonleitungen. Der Vorgang bei der Übertragung von Daten über die serielle Schnittstelle verläuft so, daß die 8-, 16- oder 32-Bit breiten Daten aus dem Bussystem in ein sogenanntes Schieberegister geladen werden.

Dort wird das mehrere Bit breite Byte in die Einzelbits zerlegt. Auf Kommando verlassen die Bits dann nacheinander das Schieberegister und gehen zum Empfänger. Hier wiederum sorgt ein umgekehrt arbeitendes Schieberegister dafür, daß das Byte wieder zusammengefügt an das Bussystem übergeben wird.

Die Verbindung zweier DEE über die serielle Schnittstelle kann direkt oder über je ein Modem erfolgen. Ein Modem ist ein Gerät, das die Bit-Daten in die für die Übertragung erforderlichen Tonfrequenzen umwandelt. Das Modem wird direkt mit der Telefonleitung verbunden, während der früher häufiger verwendete Akustikkoppler keinen solchen Anschluß verlangte, dafür aber auch wesentlich geringere Übertragungsgeschwindigkeiten erlaubte. Zur Übertragung von Modem zu Modem werden die im



Die neunpolige Schnittstelle

oberen Teil unseres Bildes "Elektrische Verbindung der Übertragungseinrichtungen" gezeigten Verbindungen hergestellt. Die Übertragung der Daten allein genügt noch nicht. Es muß vielmehr dem Empfänger bekannt sein, in welcher Geschwindigkeit die Daten aus dem Schieberegister des Senders kommen. Diese Information kann auf zweierlei Art erfolgen:

1. Synchrone serielle Übertragung

Bei dieser schickt der Sender den Takt, unter dem die Datenbits sein Schieberegister verlassen, über eine Synchronisationsleitung an den Empfänger. Dieser arbeitet dann im Gleichtakt mit dem Sender. Das ermöglicht hohe Übertragungsgeschwindigkeiten, verteuert aber durch die zusätzlich erforderliche Datenleitung die Einrichtung. Sie ist deshalb nur sinnvoll bei ständigen Verbindungen wie etwa der Steuerung und Regelung im industriellen Bereich.

2. Asynchrone serielle Übertragung

Hier erzeugt der Empfänger seinen eigenen Takt. Dessen Frequenz muß deshalb ungefähr der des Senders entsprechen. Damit der Empfänger erkennt, daß eine Datenübertragung folgt, werden die Daten in Datenworte zerlegt, denen jeweils ein Startbit vorausgeschickt wird und ein bis zwei Stopbits folgen. Die Stopbits geben dem Empfänger etwas Zeit zur korrekten Datenübernahme. Die Bitzahl der Datenworte hängt jeweils von der Struktur der zu sendenden Daten ab. ASCII-Texte benötigen nur 7-Bit, Zahlen dagegen mindestens 8-Bit Breite. Zur Sicherheit gegen Fehler im Leitungssystem kann bei ASCII-Textübertragungen (7-Bit) zusätzlich zu Startbit, Datenbits, Stopbit(s) das achte Bit als "Parity-Bit" gesandt werden. Dieses Bit kann gesetzt oder nicht gesetzt sein. Der Sender gibt bei der Öffnung des Datenkanals vor, ob und welche Art Parity bei der Aussendung der Daten angewandt wird. Ist die Parity mit "even" das heißt, "gerade" vorgegeben, dann setzt der Sender das Parity-Bit, wenn die Quersumme der Datenbits ungerade ist, das heißt, die Quersumme der Datenbits wird gerade gesetzt.

Bei Parity gleich "odd" (ungerade) sorgt der Sender dafür, daß die Quersumme ungerade wird. Da jedes Datenwort ein Zeichen repräsentiert, kann auf der anderen Seite der Empfänger ermitteln, ob dieses Zeichen korrekt ist oder nicht. Die Information darüber, mit welcher Geschwindigkeit, ob ein Parity-Bit, wie lang das Datenwort und ob ein oder zwei Stopbits gesendet werden, wird beim Öffnen des Datenkanals gleichfalls angegeben. In der folgenden Programmzeile sehen Sie, wie man unter BASIC einen Datenübertragungskanal auf COM1, der ersten seriellen Schnittstelle, öffnet, dessen Übertragungsrate 1200 Baud (Bit pro Sekunde) umfaßt und dessen Datenstruktur mit einem Parity-Bit, sieben Datenbits und einem Stopbit definiert wird.

```
OPEN "COM1:1200,E,7,1" AS #Nr
```

Pin	Kürzel	Name	Funktion	Wer
1	PG	Protective Ground	Schutzerde	DTE, DCE
2	TD auch TxD	Transmit Data	Daten senden	DTE
3	Rd auch RxD	Receive Data	Daten empfangen	DCE
4	RTS	Request To Send	Zum Senden auffordern	DCE
5	CTS	Clear To Send	Sendebereit	DCE
6	DSR	Data Set Ready	Empfänger betriebsbereit	DTE
7	SG	Signal Ground	Betriebserde	DTE, DCE
8	DCD	Data Carrier Detect	Empfangsbereutpegel	DCE
20	DTR	Data Terminal Ready	Sendegerät betriebsbereit	DTE
22	RI	Ring Indicator	Ankommender Ruf	DCE
24	TC	Transmit Clock	Taktvorgabe für Empfang	DTE

In der Spalte Wer werden folgende Abkürzungen verwendet:

DTE	Data Terminal Equipment	deutsch: Datenübertragungs-Endeinrichtung
DCE	Data Communication Equipment	deutsch: Datenübertragungs-Einrichtung

Die benutzten seriellen Leitungen

Die Übertragungsrate (auch Baudrate genannt) gibt an, wieviel Zeichen (Parity-, Start- und Stop-Bits eingerechnet) pro Sekunden gesendet werden. Eine Übertragungsrate von 1200 Baud ergibt in unserem Beispiel die Übertragung von 1200/10 (Startbit, 7 Datenbits, Stopbit und Paritybit = 10 Bit) also 120 Zeichen pro Sekunde.
E steht für Parity = Even. In BASIC kann die Parity-Information folgende Werte annehmen:

E	Even	gerade Quersumme
O	Odd	ungerade Quersumme
N	Non keine Parität, dieser Wert muß bei Datenwortlängen von 8 Bit immer vorgegeben werden.	
S	Space	Leerzeichen
M	Mark	Kennzeichen

Weitere Einzelheiten zum COM-Befehl entnehmen Sie bitte Ihrem BASIC-Handbuch.

Die Datenübertragung mit der seriellen Schnittstelle

Die serielle Datenleitung führt im logischen 0-Zustand eine positive Spannung zwischen +3 und +25 Volt und im logischen 1-Zustand (High) eine negative Spannung von -3 bis -25 Volt. Diese Pegelung mit negativer Logik (siehe: Fachbegriffe, kurz erklärt: Logikpegel) wird auch als V.24-Pegelung bezeichnet. Die Werte liegen über den +5 Volt der TTL (Transistor-Transistor-Logik) des Computers (siehe unten, im Unterabschnitt "Die parallele Schnittstelle"). Dadurch liegen die möglichen Übertragungsentfernungen zwischen Rechner und Modem bei etwa 30 Metern und die maximal mögliche Übertragungsgeschwindigkeit bei maximal 20 kByte = 1024*8 * 20 = 163840 Bit/s), entsprechend unserem Beispiel also 163840/10 = 16384 Zeichen pro Sekunde.

Realistisch sind zur Zeit Übertragungsgeschwindigkeiten zwischen 75 und 19200 Baud. Die Daten können zwischen zwei Rechnern bei geringer Entfernung direkt oder bei größeren Entfernungen über Modem ausgetauscht werden. In diesem Falle sind die Rechner das DTE (Data Terminal Equipment) oder deutsch die DEE (Datenübertragungs-Endeinrichtung) und die Modems sind das DCE (Data Communication Equipment) oder in deutsch die DÜE (Datenübertragungs-Einrichtung). Im folgenden verwenden wir die schon in unseren Schnittstellenbildern und der Tabelle "Die benutzten seriellen Leitungen" verwendeten Abkürzungen.

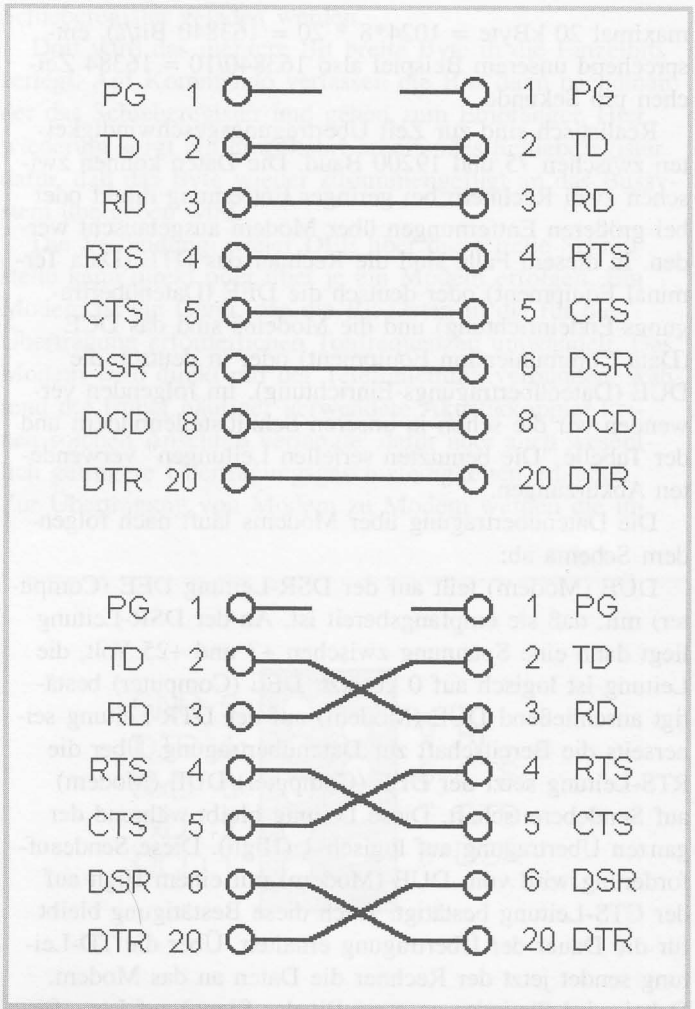
Die Datenübertragung über Modems läuft nach folgendem Schema ab:

DÜE (Modem) teilt auf der DSR-Leitung DEE (Computer) mit, daß sie empfangsbereit ist. An der DSR-Leitung liegt dann eine Spannung zwischen +3 und +25 Volt, die Leitung ist logisch auf 0 gesetzt. DEE (Computer) bestätigt anschließend DÜE (Modem) auf der DTR-Leitung seinerseits die Bereitschaft zur Datenübertragung. Über die RTS-Leitung setzt der DEE (Computer) DÜE (Modem) auf Sendebereitschaft. Diese Leitung bleibt während der ganzen Übertragung auf logisch 1 (High). Diese Sendeaufforderung wird vom DÜE (Modem) mit einem High auf der CTS-Leitung bestätigt. Auch diese Bestätigung bleibt für die Dauer der Übertragung erhalten. Über die TD-Leitung sendet jetzt der Rechner die Daten an das Modem. Dabei wird für jedes gesetzte Bit das Signal auf Low, für jedes nicht gesetzte Bit auf High gesetzt.

Dies ist genau umgekehrt wie bei den Steuersignalen. Man sagt, Steuersignale seien High-aktiv und Datensignale Low-aktiv. Nach Abschluß der Datenübertragung wird vom DEE sein Bereit-Signal auf der RTS-Leitung auf Low gesetzt. Dies wird von DÜE mit dem Gegensignal CTS gleich Low bestätigt. Den gesamten Komplex der gegenseitigen Anforderungs- und Bestätigungssignale bezeichnet man als Handshaking (deutsch: Händeschütteln). Mit dem Handshaking informieren sich die beiden an der Übertragung beteiligten Geräte gegenseitig über ihren Zustand. Das Handshaking ist für die Übertragung größerer Datenmengen zu zeitraubend. In solchem Fall werden deshalb ganze Datenpakete übersandt, die je nach verwendetem Protokoll durch spezielle Zeichen begrenzt werden.

In unserem Bild sehen Sie bei den Leitungsverbindungen die Leitung PG zwischen den beiden Übertragungseinrichtungen unterbrochen. Das ist wichtig. Mit der Abschirmung des Datenkabels darf nur eines der Geräte über PG verbunden sein, um die Störsicherheit zu erhöhen. Sind beide angeschlossen, so kann dies zusätzliche Störungen verursachen.

Im unteren Teil des gleichen Bildes finden Sie die Kabelverbindung zweier Rechner mit dem sogenannten Nullmodem. Diese Verbindung weist gegenüber der zwischen zwei Modems liegenden Leistungsbelegung drei gekreuzte Anschlußpaare auf. Durch diese Kreuzverbindung wird die sonst von den Modems vorgenommene Adressierung der Schieberegister hergestellt. Man kann sich also sehr leicht selbst ein Nullmodem herstellen.



Elektrische Verbindungen der Übertragungseinrichtung

Ein Tip zum Programmieren der seriellen Schnittstelle sei noch nachgetragen. Bei den mit MS-DOS mitgelieferten Programmen finden Sie auch das Programm MODE, das den entsprechenden externen Befehl repräsentiert. Analog zu der Programmierung unter BASIC mit OPEN COM kann man mit dem MODE-Befehl auch serielle Schnittstellen konfigurieren. Mit Hilfe des Mode-Befehls ist auch der Anschluß eines Druckers an die serielle Schnittstelle programmierbar. Bitte lesen Sie im DOS-Handbuch die entsprechenden Hinweise.

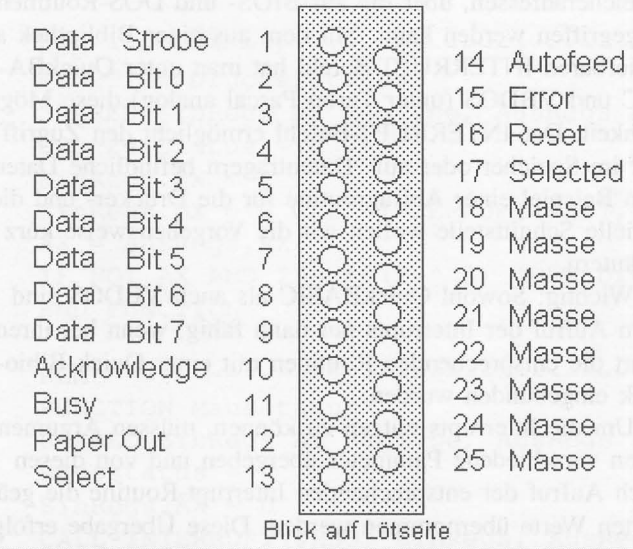
Die parallele Schnittstelle

Im Gegensatz zur V.24-Pegelung (-25 V bis -3V und +3V bis +25 V) der seriellen Schnittstelle (siehe oben) verwendet die parallele Schnittstelle die TTL-Pegelung. Dabei entsprechen Werte gleich oder unterhalb 0,8 Volt dem Low-Pegel, Werte gleich oder über 2,4 Volt dem High-Pegel. Die TTL-Pegelung ist eine mit positiver Logik, wie sie von den meisten Bauteilen des Rechners verwendet wird (siehe Fachbegriffe, kurz erklärt: Logikpegel). Die parallele Schnittstelle wird auch häufig Centronics-Schnittstelle genannt. Die Pin-Belegung wurde nämlich von der durch den Druckerhersteller Centronics kreierten (36-poligen) Schnittstelle auf die 25-polige übernommen. Unser Bild "Die parallele Schnittstelle" zeigt die Pinbelegung. Auch hier werden wieder englische Bezeichnungen verwendet, die in nachstehender Tabelle "Die Pinbelegung" erläutert werden.

In unserer Tabelle finden Sie in der letzten Spalte die Hinweise High- oder Low-Aktiv. Dies zeigt an, welchen Wert der davor stehende Zustand zurückgibt. Ein Online-Fehler beim Drucker zum Beispiel gibt auf Pin 15 eine logische 1 zurück, auf Pin 13 dagegen eine 0. Diese beiden Werte sagen also aus: Die serielle Schnittstelle meldet einen Fehler und der Drucker ist nicht Online.

Die parallele Schnittstelle verwendet wie die serielle das D-Sub-25-Steckersystem. Die Anschlüsse sind aber genau umgekehrt. Im Rechner sitzt die Steckerbuchse, der Leitungsanschluß hat den Stecker. Auch das Interface für den fischertechnik-Profi-Computing-Baukasten wird über den parallelen Anschluß mit dem Rechner verbunden. Wir gehen im Abschnitt fischertechnik Profi-Computing auf Einzelheiten ein. Wichtig ist, daß im Gegensatz zu unserer Lucky-Logic-Demoversion bei der Erstinstallation der Lukky-Logic-Vollversion auf Ihrem Rechner das Interface an der parallelen Schnittstelle angeschlossen sein muß.

Da Lucky Logic außerdem nur mit Maus und Tastatur zusammen verwendet werden kann, beschreiben wir im nachfolgenden Unterabschnitt ein kleines BASIC-Programm, das sowohl den Mausanschluß als auch den des Druckers prüft und die Ergebnisse verwendet, um Warn-



Die parallele Schnittstelle

meldungen auszugeben. Wir haben die entsprechenden Funktionen auch in unser Startprogramm eingefügt.

Allgemeines zu den Schnittstellen

Es gibt immer wieder das Problem, daß man vor Start oder am Anfang eines Programms die serielle und die parallele Schnittstelle darauf prüfen will, ob der Anschluß in Ordnung ist. Da dies eine immer wieder gestellte Frage ist, wollen wir hier die Möglichkeiten zur Lösung des Problems etwas näher betrachten.

Es geht darum, den Zustand der im Rechner integrierten seriellen und parallelen Schnittstellen zu ermitteln. Dazu sollte man zunächst wissen, welche Schnittstellen überhaupt angesprochen werden können. Wir haben in den vorherigen Unterabschnitten die physikalische Ausbildung der Schnittstellen erläutert. Dabei haben wir festgestellt, daß zum Beispiel zwischen Rechner und Drucker zusätzlich zu den übertragenen Daten auch Informationen über den Zustand der Schnittstelle ausgetauscht werden.

Diese Informationen gilt es zu nutzen. Logischerweise werden alle diese Informationen an genau definierten Stellen im Arbeitsspeicher abgelegt, wo sie dann von jedem, den es angeht, abgefragt werden können. Wo liegen also die Speicheradressen der Schnittstellen? Wir haben in der Tabelle "Die Portadressen" die Adressen aller unter DOS möglichen parallelen und seriellen Schnittstellen aufgeführt. Die Adressen sind jeweils 2-Byte-Werte, deshalb stehen in der Tabelle jeweils zwei Wertangaben. Dabei ist der erste Wert der des Highbyte, des höherwertigen Bytes. Dieser Wert ist also immer mit 256 zu multiplizieren, bevor er zu dem nachfolgenden Lowbyte (niederwertigem Byte) addiert wird. Die Bezeichnung Port ist übrigens eine alternative Bezeichnung für Schnittstellen zwischen den Bussen des Mikroprozessors und der Außenwelt.

Wie Sie aus der Tabelle der Portadressen ersehen, können je bis zu vier serielle und parallele Schnittstellen eingebaut sein:

Adresse		Schnittstelle		
DEZ	HEX	Typ	Bezeichnung	Inhalt
1032/1033	400/401	ser	COM1	I/O-Basisadresse
1034/1035	402/403	ser	COM2	I/O-Basisadresse
1036/1037	404/405	ser	COM3	I/O-Basisadresse
1038/1039	406/407	ser	COM4	I/O-Basisadresse
1040/1041	408/409	par	LPT1	I/O-Basisadresse
1042/1043	40A/40B	par	LPT2	I/O-Basisadresse
1044/1045	40C/40D	par	LPT3	I/O-Basisadresse
1046/1047	40E/40F	par	LPT4	I/O-Basisadresse

An den in der Tabelle angegebenen Adressen steht jeweils die Eingabe-/Ausgabe-Basisadresse der Schnittstellen. Der Wert dieser Adresse ergibt die in unserem Bild "Die I/O-Bits der Schnittstellen" dargestellten Bits. In der ersten Zeile finden Sie alle Bits auf True High gesetzt. Dieses Bild ergibt sich, wenn das fischertechnik-Interface angeschlossen ist. In diesem Fall wird kein Fehler (Bit 3) gemeldet, obwohl kein Papier (Bit 5) bei einem Drucker ein Fehler sein würde. Die beiden darunterstehenden Zei-

1	Data Strobe	Rechner an Drucker: Daten übernehmen	Low
2-9	Data Bit 0 bis Bit 7	Rechner an Drucker: Data-Bits 0 bis 7	Low
10	Acknowledge	Drucker an Rechner: Bestätigen der Daten	Low
11	Busy	Drucker an Rechner: Empfangsbereit	High
12	Paper Out	Drucker an Rechner: Kein Papier	High
13	Select	Drucker an Rechner: Drucker Online	High
14	Autofeed	Rechner an Drucker: Seitenvorschub durch Rechner	Low
15	Error	Drucker an Rechner: Fehler beim Drucken	Low
16	Reset	Rechner an Drucker: Neustart des Druckers	Low
17	Selected	Rechner an Drucker: Druckeranwahl	Low
18-25	Masse		

Die Pinbelegung

len zeigen die Biststruktur bei eingeschaltetem beziehungsweise ausgeschaltetem Drucker (Bit 4).

Um diese Ausgabe zu erhalten, müssen wir die entsprechende Speicheradresse abfragen und die zurückgegebenen Werte in Bits zerlegen. Zunächst also muß die Adresse abgefragt werden. Dazu gibt es in den verschiedenen Programmierhochsprachen die entsprechenden Befehle. In BASIC ist es der Befehl PEEK. Bitte beachten Sie aber, daß dieser Befehl unter Windows nicht mehr verfügbar ist, da Windows den Zugriff auf die Betriebssystem-Ebene selbst verwaltet. Wir haben in unserem Listing "Die Schnittstellenabfrage" die vollständige Lösung unter BASIC aufgezeigt. Bei genauer Betrachtung dieses Listings sind zwei in einer Schleife stehende Programmzeilen von besonderer Bedeutung:

```
FOR m =  1 TO 4
   X = &H6
   lpt(m) = PEEK(X + (2 * m + 1)) * 256
                + PEEK(X + (2 * m))
NEXT
```

Mit der PEEK-Funktion können Speicheradresseninhalte ermittelt werden. Die erste Verwendung von PEEK ermittelt den Wert am High-Byte-Speicherplatz, die zweite am Low-Byte-Platz. Wichtig ist aber, daß die in der Schleife verwendeten Werte nicht die Adressen sind, sondern diese Werte nur den sogenannten Offset wiedergeben. Eine vollständige Speicheradresse setzt sich nämlich aus dem Segmentwert und dem Offset, also der Stelle ab Segmentanfang, zusammen.

Unser Segment beginnt bei Hex 40 (Dezimal 64). Die Anweisung DEF SEG = &H40 besorgt die Umschaltung in das neue Speichersegment. Um Fehler zu vermeiden, wird am Schluß des Programms der Zeiger auf die Basic-Segmentadresse mit DEF SEG (ohne Argument) zurückgesetzt.

Die erste Druckeradresse hat ein Offset von &H8 für das Low- und &H9 für das High-Byte. Das finden Sie bestätigt, wenn Sie sich unsere Tabelle "Die Portadressen" nochmals ansehen. Die anderen Offsets schließen sich an, so daß eine Aufbereitung in einer Schleife möglich ist. Das Äquivalent zum PEEK-Befehl ist der POKE-Befehl. Mit ihm kann man gezielt Werte an bestimmte Adressen geben.

Wir haben mit der Abfrage von Adressen eine Möglichkeit der Speicherabfrage kennengelernt. Eine weitere Möglichkeit liegt in einer Besonderheit der DOS-Speicherverwaltung begründet.

Die Interrupt-Abfrage

Was den Windows-Programmierern die Message-Queue (Informationen-Schlange) ist, ist für die DOS-Programmierer der Zugriff auf die Interrupt-Adressen. Das sind spezielle, vor dem Zugriff durch andere Programme gesicherte Speicheradressen, über die auf BIOS- und DOS-Routinen zugegriffen werden kann. Mit dem aus einer Bibliothek aktivierbaren INTERRUPT-Befehl hat man unter QuickBASIC und VBDOS (unter Turbo Pascal analog) diese Möglichkeit. Der INTERRUPT-Befehl ermöglicht den Zugriff auf im Speicher oder auf Datenträgern befindliche Daten. Am Beispiel einer Abfragroutine für die Drucker- und die serielle Schnittstelle wollen wir die Vorgehensweise kurz erläutern.

Wichtig: Sowohl QuickBASIC als auch VBDOS sind zum Aufruf der Interrupts nur dann fähig, wenn bei ihrem Start die entsprechenden Routinen mit einer Quick-Bibiothek eingebunden wurden.

Um die Interrupts nutzen zu können, müssen Argumente an verschiedene Parameter übergeben und von diesen nach Aufruf der entsprechenden Interrupt-Routine die geänderten Werte übernommen werden. Diese Übergabe erfolgt an dafür reservierte Speicherplätze, die sogenannten Register. Wichtig sind zum Beispiel

AX der Akkumulator
BX das Basisregister
CX das Zählerregister
DX das Datenregister.

Diese Register übernehmen 2-Bytewerte. Daher wird zum Beispiel der Akkumulator auch häufig als AH (High-Byte von AX) und AL (Low-Byte von AX) aufgeschlüsselt. Analog geschieht das mit den anderen Registern. In den genannten BASIC-Versionen werden zunächst die Register mit der TYPE-Anweisung als RegType deklariert und danach mit DIM SHARED die Variablen EinRegs und AusRegs als RegType dimensioniert.

Diese Variablen übergeben oder übernehmen im Programm die Werte der Register. Mit SHARED dimensionierte Variablen sind bei QB und VBDOS in jeder Prozedur des aktuellen Moduls bekannt.

Wir haben für unser Programm zwei FUNCTION-Prozeduren geschrieben, die Sie in Ihre eigenen Programme einbauen können. FUNCTION-Prozeduren können wie Variablen benutzt werden.

Sie geben über ihren Namen Werte oder Adressen auf Werte zurück. Sie finden die beiden Prozeduren nachstehend unter "Drucker und Maus" und in unserem Listing SCHNITT.BAS am Ende des Heftes. Das ausführbare Programm finden Sie als SCHNITT.EXE auf der Diskette.

Wir verwenden die zwei Interrupts 17h(Func 01h) und 33h(Func 00h) für unsere Interruptabfragen. (h steht für Hexadezimal). Welche Werte übergeben und welche zurückgegeben werden, sehen Sie in unserer Übersicht "Zwei Interrupts".

Int 17h
Interrupt für parallele Schnittstelle Funktion 01h
Eingabe: AH = 1

```
REM ------------------------------------------------------------
FUNCTION DruckerStatus
SHARED EinRegs AS RegType, AusRegs AS RegType
DruckerStatus = 0
EinRegs.AX = 2
EinRegs.DX = 0 CALL INTERRUPT(&H17, EinRegs, AusRegs)
AX = AusRegs.AX
IF NOT AX AND 256 * (2 ^ 3) THEN DruckerStatus = -1
END FUNCTION

REM ------------------------------------------------------------
FUNCTION Mausstatus
SHARED EinRegs AS RegType, AusRegs AS RegType
Mausstatus = 0
EinRegs.AX = 0
CALL INTERRUPT(&H33, EinRegs, AusRegs)
IF AusRegs.AX = 0 THEN
 EXIT FUNCTION
END IF
Mausstatus = -1
END FUNCTION
```

Drucker und Maus

DX = Nr der parallelen Schnittstelle (von 0 zählend)
Rückgabe: AH = Status der Schnittstelle

Bit	Zustand
7	Beschäftigt
6	Empfangsbestätigung
5	kein Papier/Papier vorhanden
4	Online/Offline
3	I/O-Fehler
2	reserviert
1	reserviert
0	Time Out-Fehler

Int 33h
Mausinterrupt
Funktion 00h
Eingabe: 0
Rückgabe: AX = 0 kein Maustreiber installiert
AX = -1 Treiber installiert
BX = Anzahl der Maustasten (in der Funktion nicht genutzt)

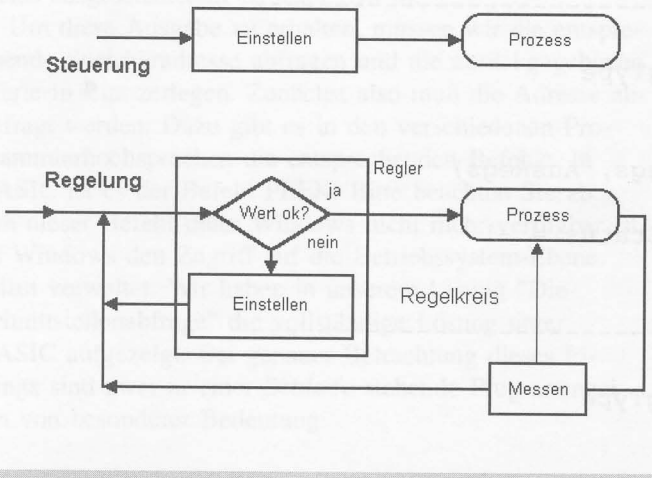
Steuern und Regeln

Wir wissen jetzt also, wie man Werte mit Sensoren ermittelt, wie analoge Signalwerte zu Digitalwerten werden und wie diese in den Rechner gelangen. Im folgenden Unterabschnitt wollen wir uns mit dem Thema "Steuern und Regeln" auseinandersetzen. Wir wissen, daß man mit Steuern die Abarbeitung von Vorgaben, mit Regeln dagegen die über Feedback korrigierbare Arbeit in einem Regelkreis bezeichnet. Das Symbol für das Steuern ist somit die Linie, das für das Regeln der Kreis. In unserem Bild "Steuern und Regeln" haben wir die Zusammenhänge grafisch dargestellt.

Jeden Tag bekommen wir es mit Steuerung und Regelung zu tun, wenn wir uns mit einem Fahrzeug in das Verkehrsgewühl begeben. Viele der Ampelanlagen sind auch heute noch zeitgesteuert. Sie geben in einem voreingestellten Zeittakt, der von einem Taktgeber (zum Beispiel einer Uhr) überwacht wird, die einzelnen Fahrtrichtungen und Überwege frei. Auch wenn dieser Zeittakt je nach Tageszeit unterschiedlich eingestellt wird, so reagiert das System nicht auf unterschiedlich starke Verkehrsströme. Hier liegt eine typische Verkehrssteuereung vor.

Etwas anderes ist dies bei Kreuzungen, bei denen jedes über die Kreuzung fahrende Fahrzeug über Induktionsschleifen oder Fußgänger und Radfahrer über eine Taste ein Signal an die Anlage absetzen, das in dieser verarbeitet wird. Die Induktionsschleife nutzt die von dem englischen Physiker Michael Faraday 1831 entdeckte elektromagnetische Induktion. Die Induktionsschleife ist eine Leiterschleife, die in die Fahrbahn eingelassen ist. Überfährt eine Fahrzeug diese Schleife, so ändert das magnetische Feld des Fahrzeugs die Spannung an den Anschlüssen der Schleife. Wird eine Induktionsschleife im Kreuzungsbereich überfahren, wird dieses Signal an die Anlage weitergeleitet.

Im Rechner der Anlage werden jetzt zum Beispiel Anzahl und (Zeit-)Abstände der Fahrzeuge in der gerade offe-



Regeln und Steuern

nen Fahrtrichtung geprüft. Gleichzeitig wird aufgenommen, ob an einer der anderen Stellen ein Verkehrsteilnehmer gemeldet wurde. Ist dies der Fall und unterschreitet die registrierte Zahl der Fahrzeuge in der offenen Richtung einen bestimmten Wert oder überschreiten die Zeitabstände einen vorgegebenen Wert, so wird die bisher freie Richtung über Gelb auf Rot und dann die neue Richtung auf Grün geschaltet. Diese Art der Beeinflussung der freigegebenen Richtungen durch die Fahrzeugzahl und die Fahrzeugabstände ist eine Verkehrsregelung.

Am Beispiel der Verkehrsregelung sieht man, welche Vorteile Regelung gegenüber Steuerung hat. Man sieht aber auch am Vergleich, daß Regelung immer einen höheren technischen Aufwand fordert als Steuerung. Es ist also immer eine Entscheidung zu treffen: Ist Regelung erforderlich, ist der Aufwand gerechtfertigt.

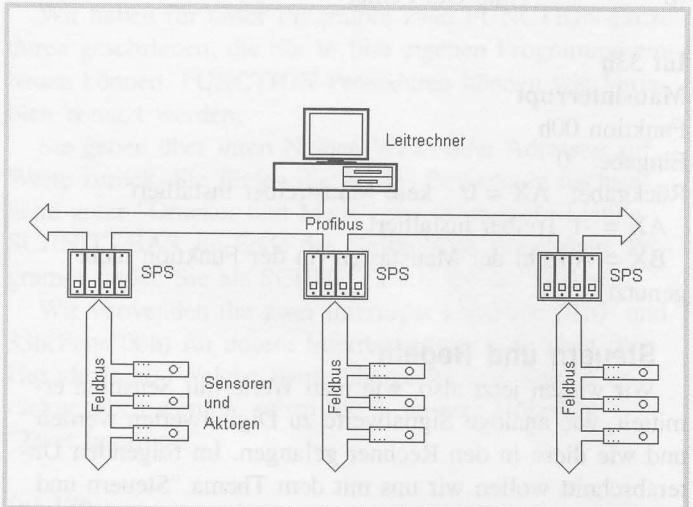
Ein typischer Fall von Steuerung ist im industriellen Bereich die Steuerung von Werkzeugmaschinen. Man kann eine Drehbank, auf der eine Metallspindel aus einem Metallblock gedreht werden soll, auf verschiedene Weise steuern. Die Steuerung kann von Hand vom Dreher selbst erfolgen und verlangt hohe Fertigkeit von diesem. Sie ist, da der Dreher auf unterschiedliche Gegebenheiten reagieren kann, eigentlich eine Form der Regelung. Als nächstes kann der Ablauf aller Vorgänge auf einem Datenträger (zum Beispiel Lochband) gespeichert und in die Steuerung der Drehbank eingespeist werden. Damit ist sichergestellt, daß die Arbeiten in immer gleicher Weise Werkstücke exakt reproduzieren. Eine dritte Möglichkeit ist die sogenannte Computerized Numerical Control-Steuerung (CNC-Steuerung), die computerunterstützte numerische Steuerung von Werkzeugmaschinen. CNC-Maschinen enthalten einen Computer, der vor Ort programmierbar ist.

In Kombination mit entsprechenden Datenträgern ist dieses System schnell anpassungsfähig an unterschiedliche Anforderungen. Als Nachteil dieser Maschinen zeichnet sich ab, daß sie lediglich bestimmte vorhersehbare Arbeiten verrichten und zu wenig flexibel sind. Die Übertragung der zunächst auf CNC-Maschinen realisierten Steuerung mit Hilfe des Computers auf die Automatisierung und Steuerung von technischen Vorgängen in anderen Bereichen (zum Beispiel Montagetechnik, Verfahrenstechnik) wird als SPS (Speicherprogrammierbare Steuerung) bezeichnet.

Das Herz dieser Steuerungen ist der Computer. Aus der Koppelung von Einzelmaschinen entsteht bei der Serienproduktion die Fertigungsstraße. Bei ihr werden Einzelmaschinen von einer zentralen Steuerung aus überwacht und gesteuert. Fertigt eine Produktion zwar viele, aber unterschiedliche Teile, dann ist es sinnvoll, die Steuerung aufzuteilen in die zentrale Ablaufsteuerung und die Maschinensteuerung vor Ort. Dies gilt auch, wenn sich Produktionsprozesse schnell anpassen müssen. In unserem Bild "Zentral und und dezentral" haben wir die Zusammenhänge schematisch dargestellt.

Ein Leitrechner steuert über ein Bussystem (beispielsweise Profibus DIN 19245) die speicherprogrammierbaren Steuerungseinheiten (SPS). Diese erhalten ihre Informationen über Prozeßzustände von Sensoren über ein eigenes Feldbusnetz und geben Steuersignale an die Aktoren, also elektronische oder mechanische Schalter, Ventile und ähnliches, weiter.

Wie man an dem letzten Beispiel sehen kann, werden in der Praxis die Begriffe Regelung und Steuerung nicht so scharf getrennt, wie in der Theorie. SPS, also speicherprogrammierbare Steuerung geht nahtlos durch die Rückkopplung über Sensoren in speicherunterstützte Regelung über und umgekehrt. Aber darauf kommt es auch nicht an. In diesem Abschnitt ging es uns darum, Ihnen die wichtigsten Informationen zum Gesamtkomplex Messen, Steuern, Regeln zu geben. Zum Abschluß haben wir die wichtigsten Begriffe noch einmal zusammengefaßt. Im Anschluß daran finden Sie noch Hinweise auf empfehlenswerte Literatur.



Zentral und dezentral

Fachbegriffe, kurz erklärt

Diode

Dioden sind Halbleiter (siehe dort). Sie lassen den Strom nur in einer Richtung durch und dienen deshalb als elektronische Schalter oder zum Gleichrichten von Wechselspannung. In der Praxis werden Silizium- oder Germanium-Dioden eingesetzt. Besondere Formen der Dioden sind LED (= light emitting diods, deutsch: Leuchtdioden) für Leuchtanzeigen und die Z-Dioden (Zener-Dioden) mit festgesetzten Spannungswerten für die Spannungsstabilisierung.

Fotodetektoren

Als Fotodetektoren bezeichnet man elektronische Bauelemente, die eine auftreffende elektromagnetische Strahlung (ultraviolettes, sichtbares oder infrarotes Licht) in elektrische Signale umwandeln. Zu den Fotodetektoren gehören solche mit äußerem Fotoeffekt (Fotozelle und ähnliches) und solche mit innerem Fotoeffekt wie Fotodiode und Fotctransistor.

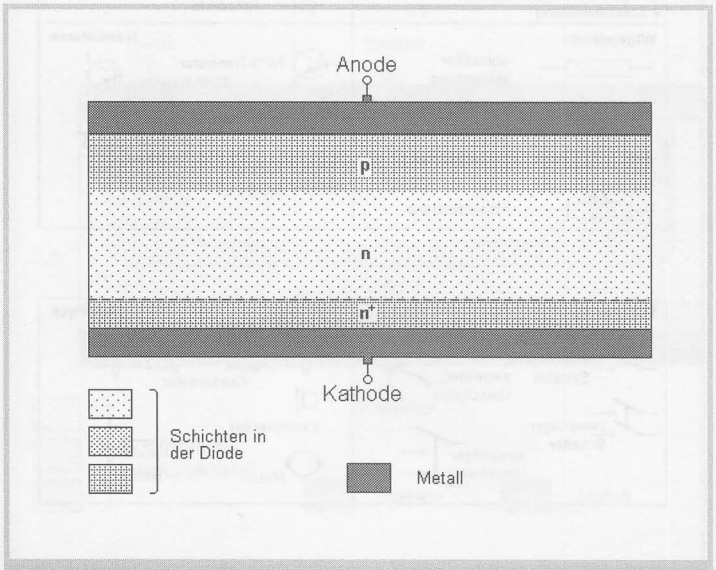
Fototransistoren

Wie der Name schon sagt, handelt es sich um Transistoren (siehe dort). Die Basis-Kollektor-Sperrschicht ist externem Licht zugänglich. Dadurch wird in Abhängigkeit von der Lichtintensität eine Spannung erzeugt, die den Fototransistor steuert.

Halbleiter

Als Halbleiter bezeichnet man Festkörper, deren elektrische Leitfähigkeit zwichen der von Metallen (Widerstand 10-6 W) und der von Isolatoren (Widerstand 1014 W) liegt. Halbleiter verlieren, im Gegensatz zu Metallen, mit abnehmender Temperatur an Leitfähigkeit.

Die Leitfähigkeit ist außer der Temperatur noch von der Zusammensetzung des Festkörpers abhängig. Insbesondere in der Elektronik werden zum Beispiel Siliciumkristalle (Si) als Halbleitermaterial eingesetzt. Mit der sogenannten Dotierung, das heißt, der genau bemessenen "dosierten" Zugabe von Fremdatomen zu reinem Halbleitermaterial, werden bei diesem Zonen unterschiedlicher elektrischer Leitfähigkeit geschaffen.



Die Diode

Je nach der Art der verwendeten Fremdatome wird unterschieden zwischen p-Dotierung mit Akzeptoren (ein Elektron weniger gegenüber dem reinen Halbleitermaterial, mit "Löchern") und n-Dotierung mit Donatoren (mit überzähligem Elektron). Bei Silizium werden Bor oder Aluminium als Akzeptoren und Phosphor oder Arsen als Donatoren verwendet.

Die Richtung, in der ein dotierter Halbleiter den elektrischen Strom durchläßt, ergibt sich aus der Art der verwendeten Dotierung. Wichtigste Anwendung der Halbleitertechnik im Elektronikbereich ist der Transistor. Um die Vorgänge in einem Halbleiter etwas zu verdeutlichen, soll hier das Prinzip der elektrischen Leitung vereinfacht dargestellt werden.

Wird eine elektrische Spannungsquelle an ein Metall angelegt, dann werden in den Metallatomen in den Außenschalen befindliche Elektronen von den von der Kathode (negativ) der Spannungsquelle kommenden Elektronen aus ihrer Position in Richtung Anode (positiv) gedrängt. In einem aus zwei (oder mehr) unterschiedlich dotierten Zonen bestehenden Halbleiter bestehen im einen Bereich (n) ein Elektronenüberschuß, im anderen (p) ein Elektronendefizit.

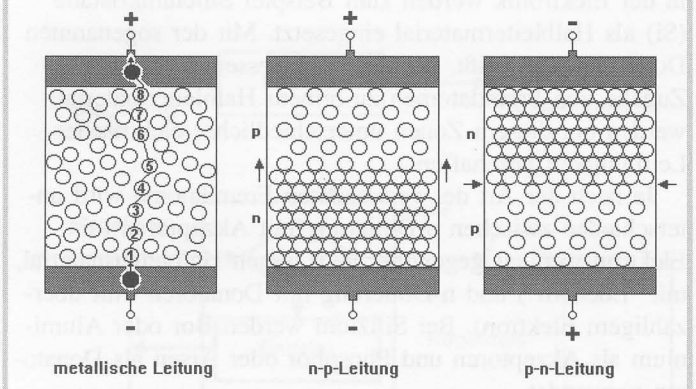
Bei ungleichen Ladungsverhältnissen besteht immer die Tendenz zum Ausgleich. Aus diesem Grund wandern überschüssige Elektronen in "Löcher" und, so sieht es aus, in Umkehrung die "Löcher" an die Stelle der Elektronen. Die Elektronen und Löcher diffundieren in benachbarte Gebiete. Dadurch entsteht im Grenzbereich zwischen vorher überversorgten und unterversorgten Bereichen ein allmählicher Ausgleich, man sagt, eine Raumladungszone.

Diese Zone, in der die Unterschiede ausgeglichen sind, wirkt immer stärker als Sperrschicht, da ihr Zustand ausgeglichen und damit stabil wird.

Wird jetzt an die Anschlüsse des Halbleiters ein Gleichstrom gelegt, so wird je nach Polung der Anschlüsse von einer Seite her die Anzahl der Elektronen erhöht, von der anderen Seite her reduziert. Es wandern Elektronen von dem negativen Pol der Stromquelle über den Halbleiter zum positiven Pol.

Diese Wanderung kann nun unterstützt oder gebremst werden von der Art der Dotierung der entsprechenden Zonen. Ist die Zonenfolge n-p, so wird die Raumladungszone reduziert und die Elektronen können direkt über die in der p-Schicht liegenden "Löcher" an die Anodenseite wandern. Ist die Schichtfolge dagegen p-n, so füllen die Eletronen zunächst die "Löcher" in der p-Schicht, treffen dann aber auf eine überfüllte n-Schicht.

Hier gelingt das Verdrängen erst dann, wenn eine gewisse Spannungsdifferenz aufgebaut, der Schwellwert erreicht ist. Aus diesem Grund sperren Dioden den Elektronenfluß in p-n-Richtung und verstärken ihn in n-p-Richtung.



Elektronenfluß in Metallen und Halbleitern

Hall-Sensoren und Reed-Kontakte

Mit diesen beiden Sensortypen können Stärken von Magnetfeldern festgestellt werden. Der Hall-Kontakt basiert auf einer Entdeckung des amerikanischen Physikers E.H.Hall. Dieser hat 1879 entdeckt, daß elektrischer Strom in Leitern, die in einem Magnetfeld angeordnet sind, eine Spannungsdifferenz senkrecht zu Strom und Magnetfeld erzeugt, die nach ihm benannte Hall-Spannung. Beim Hall-Kontakt verändert sich mit der Stärke des umgebenden Magnetfeldes die Spannung des Stromes, von dem der Hall-Kontakt durchflossen wird.

Es lassen sich mit dem Hall-Kontakt also unterschiedliche Magnetfeldstärken messen. Ein Reedkontakt (auch Reed-Relais) dagegen ist für eine bestimmte Magnetfeldstärke ausgelegt.

Der Reedkontakt besteht aus einem mit Schutzgas gefüllten Glasröhrchen, in dem zwei parallel zueinander liegende Kontaktzungen eingeschmolzen sind. Wird die vorgegebene Magnetfeldstärke erreicht, so werden die beiden Streifen aneinandergepreßt und damit ein Kontakt geschlossen.

Logikpegel

In der Digitaltechnik sind die einander jeweils entsprechenden beiden Werte 0 und 1, False und True, Low und High. Man kann diese Werte als, wie es zur Vereinfachung immer dargestellt wird, "Ausgeschaltet" und "Eingeschaltet" bezeichnen.

In der Praxis werden, da Spannungsschwankungen nicht auszuschließen sind, den beiden Logikpegeln Low und High Grenzwerte der Zugehörigkeit in einer Spannungsskala zugeordnet.

Dabei kann der höhere Spannungsbereich den Pegel High repräsentieren, man bezeichnet dies als positive Logik, oder der niedrigere Bereich, was man dann als negative Logik bezeichnet.

Die V.24-Pegelung mit High gleich oder größer als -3 Volt und Low gleich oder größer + 3 Volt entspricht also negativer Logik, die TTL-Pegelung mit High gleich oder größer als 2,4 Volt und Low gleich oder kleiner als 0,8 Volt entspricht positiver Logik. Zwischen den Grenzwerten liegt jeweils ein Sicherheitsbereich.

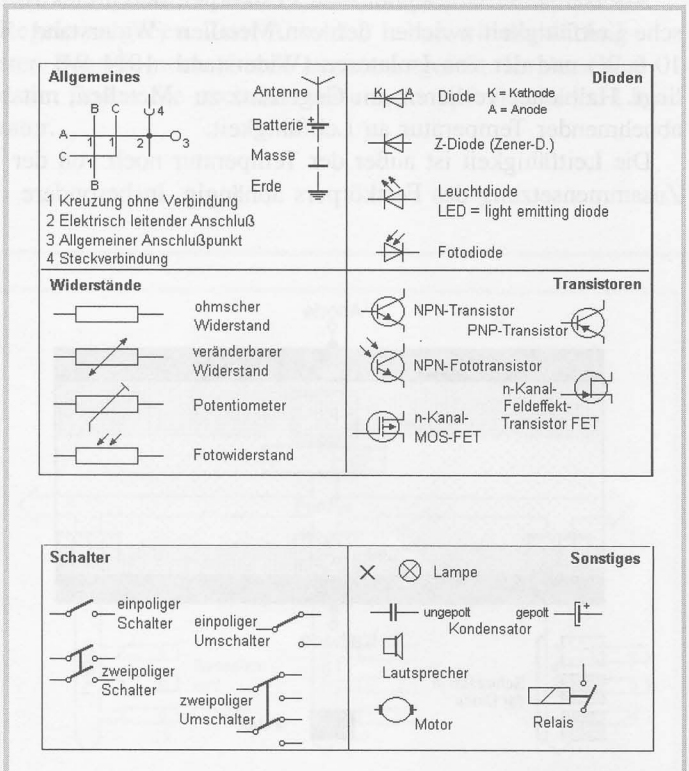
NTC-Widerstände

Als NTC-(= Negative Temperature Coefizient)Widerstände werden Halbleiterwiderstände mit negativem Temperaturkoeffizienten bezeichnet. Diese haben eine bei steigender Temperatur höhere Leitfähigkeit, das heißt, sie leiten dann den elektrischen Strom besser. Man bezeichnet sie deshalb auch als Heißleiter.

Im Gegensatz dazu sind Metalle sogenannte PTC-(= Positive Temperature Coeffizient)Widerstände, deren Widerstand mit zunehmender Temperatur steigt. Deshalb ist eine Glühlampe eine Art PTC-Widerstand. PTC-Widerstände sind zum Beispiel auch Silizium-Temperatorsensoren, bei denen die Eigenschaft des Siliziums, daß freie Elektronen in ihm mit zunehmender Temperatur zunehmend unbeweglicher werden, genutzt wird. Der innere Widerstand dieser Elemente wächst mit zunehmender Temperatur. PTC-Widerstände werden auch als Kaltleiter bezeichnet.

Schaltsymbole

Zur Vereinfachung und Vereinheitlichung der Darstellung von Schaltungen in Schaltplänen werden Symbole verwendet. Wir haben in dem Bild "Schaltsymbole" die wichtigsten Symbole für die Darstellung in elektronischen Schaltungen zusammengefaßt.



Schaltsymbole

SPS (speicherprogrammierbare Steuerung) SPS sind programmierbare elektronische Einrichtungen zur Automatisierung, Steuerung und Regelung von technischen Prozessen.

Transistor

Der Transistor wurde 1948 von dem Amerikaner William Shockley erfunden. Transistoren sind Halbleiterkristalle (siehe Halbleiter), die durch p-Dotierung und n-Dotierung Zonen unterschiedlicher elektrischer Leitfähigkeit erhalten. Unser Bild "Bipolarer Transistor" zeigt schematisch den Aufbau eines bipolaren Transistors (mit p- und n-Schichten). Er besteht aus drei Schichten, die abwechselnd p- oder n-dotiert sind.

Dabei kann die Schichtenfolge pnp (positiv-negativ-positiv) oder npn (negativ-positiv-negativ) sein. npn-Transistoren werden auch als komplementäre Transistoren bezeichnet. Zwei wichtige Repräsentanten dieser Art von Transistoren sind der Fototransistor und der Piezotransistor, dessen Leitfähigkeit sich in Abhängigkeit vom auf ihn ausgeübten Druck verändert. Außer bipolaren Transistoren gibt es unipolare Transistoren, deren wichtigste Gruppe die Feldeffekttransistoren (FET) sind.

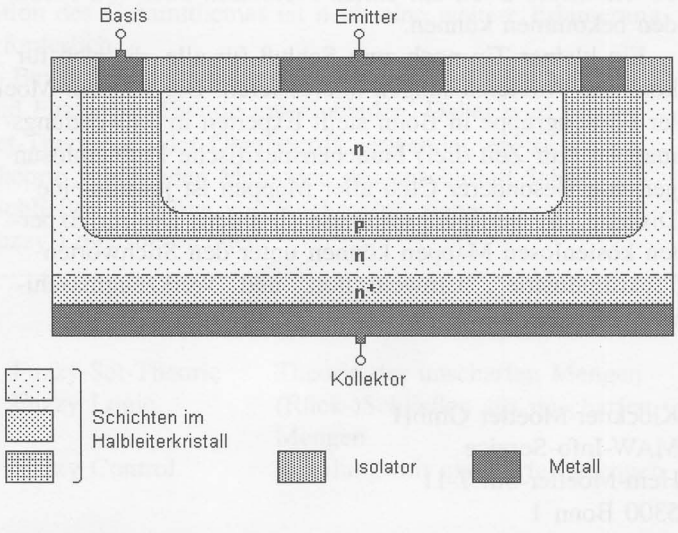
TTL (= Transistor-Transistor-Logik)

TTL-Schaltkreise sind Schaltkreise, bei denen direkt gekoppelte bipolare Transistoren außer den Verstärkerfunktionen auch die Verknüpfungsfunktionen übernehmen. Die Logikpegel werden mit einer Versorgungsspannung von +5 Volt übertragen.

Zur Vertiefung empfohlen

Wir hoffen, daß wir in diesem Abschnitt dem interessierten Laien einen kurzen Einblick in die Thematik "Messen, Steuern, Regeln" geben konnten. Wer sich intensiver damit beschäftigen möchte oder dies schon tut, dem seien vier Bücher empfohlen.

Die Bücher sind in der Reihenfolge des Schwierigkeitsgrads aufgeführt. Alle vier Bücher können unter Berücksichtigung der jeweiligen Voraussetzungen ohne Einschränkung empfohlen werden.



Bipolarer Transistor

1. (ohne Autorenangabe) Messen - was, wie, womit
Elector-Verlag, Aachen,

Wer schon seine ersten Versuche mit dem Lötkölben (Buch von Nührmann) hinter sich hat und etwas tiefer in die Thematik des Messens im Elektronik-Bereich einsteigen möchte, dem wird dieses Buch sehr hilfreich sein. Hier wird exakt beschrieben, wie elektronische Meßgeräte arbeiten, welche Meßgerätetypen es gibt und wie man damit mißt und wie man sich Meßgeräte selbst baut.

Stücklisten, genaue Baubeschreibungen und die erforderlichen (fast dreißig) Platinenlayouts im Anhang des Buches ermöglichen den Eigenbau, etwa eines Induktivitäts-Meßgerätes, eines kontaktlosen Drehzahlmessers oder eines Ultraschall-Entfernungsmessers.

2. Kai Hamann PC-Bastelbuch
Markt & Technik-Verlag, Haar bei München

Buch mit Diskette und bestückungsfertiger Platine Hamann beschreibt in seinem Buch die Möglichkeiten, den Gameport und die parallele Centronics-Schnittstelle für Messungen und Interface-Anschlüsse zu verwenden. Vom Bau einer Prüfkupplung für die Kontrolle eigener Schaltungen auf Kurzschlüsse über den zum Scanner umfunktionierten Drucker bis zum 8-Kanal-Analog/Digital-Wandler führt er den Leser und Mit-"Bastler" von einfachen zu sehr anspruchsvollen Einsatzmöglichkeiten des PC.

In diesem Buch findet der interessierte Leser viele Programmvorschläge in GW-BASIC. Die Diskette enthält die beschriebenen Programme als Text und in EXE-Form.

3. Wolfgang Link Messen, Steuern, Regeln mit PCs
Praxis der rechnergesteuerten Automatisierung
Franzis Verlag, München

Das Buch bietet eine Einführung in die Grundlagen der Automatisierung unter Einsatz des PC. Die Verwendung des PC zum Verarbeiten von Meßwerten, die Steuerung von Gleichstrom- und Schrittmotoren sowie die Regelung von Temperatur, Drehzahl oder Helligkeit sind ein paar herausgegriffene Beispiele aus den Anregungen dieses Buches.

Als Schnittstellen werden der IEC-625-Bus, die V.24-Schnittstelle und die Centronics-Schnittstelle beschrieben und verwendet. Interessant ist, daß Link, ähnlich wie wir es in diesem Heft versuchen, die Programmbeispiele im

Pseudocode angelegt hat. Dies verschafft dem Leser unabhängig von der von ihm verwendeten Programmiersprache einen schnellen Einblick in die zentralen Algorithmen.

4. Burkhard Kainka Messen, Steuern, Regeln über die RS 232 Schnittstelle
Franzis Verlag, München

Kainka beginnt seine Einführung in die Thematik mit der Erläuterung des Datenaustauschs zwischen zwei Rechnern mit dem Nullmodem. Das Buch gibt im Gegensatz zu den vorgenannten keine detaillierten Bauanleitungen, sondern bemüht sich darum, Wissen zu vermitteln und zu eigenen Versuchen anzuregen. Die Programme sind in GW-BASIC sowie zum größten Teil in Turbo Pascal geschrieben. In diesem Buch liegt der Schwerpunkt eindeutig bei der Vermittlung zusätzlichen Wissens. Es setzt deshalb solide Elektronic-Grundkenntnisse und einige brauchbare Meßgeräte (zum Beispiel Oszilloskop) voraus.

Ein weiteres Buch sei hier noch erwähnt, das Sie aber leider nicht mehr im Handel bekommen, da es schon Anfang der 80er Jahre erschienen ist und inzwischen aus dem Programm gestrichen wurde:

5. Dieter Nührmann Elektronik ohne Geheimnisse
Franzis Verlag, München
Eine Taschenbuchausgabe ist 1984 bei Knaur erschienen. Dieses Buch führt jeden, der keine zwei linken Hände hat, von den Grundlagen des Lötens über die theoretischen Grundlagen wie Anwendung des Ohmschen Gesetzes bis zum Bau eines kompletten Funkempfängers. Das alles wird in einer von vielen Abbildungen unterstützten kurzweiligen und doch exakten Art vermittelt, so daß dieses Buch für das Selbststudium jedes Laien nur wärmstens empfohlen werden kann.

Wenn Sie "blutiger Anfänger" sind, sollten Sie sehen, ob Sie dieses Buch auf einem Flohmarkt oder von Freunden bekommen können.

Ein kleiner Tip noch zum Schluß für alle, die sich für konkrete Problemlösungen interessieren. Die Klöcker-Moeller Firmengruppe in Bonn stellt Experten in Anwendungsmappen (zur Zeit drei) branchenspezifische Problemlösungen vor. Besonderer Clou aller Mappen ist jeweils eine Diskette, die Fertigungsprozesse simuliert. Die für Experten kostenlosen Mappen können unter den Stichworten "Textilindustrie", "Umweltschutz" und "Werkzeugmaschinen" bezogen werden bei:

Klöckner-Moeller GmbH
MAW-Info-Service
Hein-Moeller-Str. 7-11
5300 Bonn 1

Fuzzy Logic - eine faszinierende Logik

Das Wort Fuzzy Logic geht um in aller Munde. Nachdem einige japanische Unternehmen in Produkten der Consumer Electronic zum Beispiel in Videokameras Chips mit Fuzzy-Logic integriert haben und die Käufer feststellen konnten, daß es plötzlich möglich war, verwacklungsfrei zu filmen, seither kennt wohl jeder ein wenig technisch Interessierte dieses Thema. In diesem Abschnitt werden wir versuchen, in dieses wirklich faszinierende Thema einzuführen, ohne die Darstellung mit mathematischen Erklärungen zu überfrachten.

Zunächst soll noch ein wenig zu dem Begriff gesagt werden. Es ist immer problematisch, Begriffe aus fremden Sprachen zu übersetzen. Den Begriff Fuzzy Logic in "Unscharfe Logik" zu verwandeln, ist ähnlich problematisch wie die Übersetzung von "Artificial Intelligence" mit "Künstliche Intelligenz". Die wörtliche Übertragung kann alle im Original mitschwingenden Teilbedeutungen nicht annähernd erfassen. Es ist deshalb besser, diesen Begriff nicht zu übersetzen, sondern als Anglizismus zu erhalten.

Denn die "Unscharfe Logik" ist, da sie mathematisch definierbar ist, eben nicht unscharf. Wir werden den Begriff Fuzzy Logic ohne Übersetzung verwenden. Zur Definition des Gesamtthemas ist noch eine weitere Erläuterung erforderlich.

Bei den Mathematikern wird zwischen Fuzzy Logic und Fuzzy-Set-Theorie unterschieden. Dabei wird Fuzzy-Set-Theorie als der Oberbegriff verwendet, die zu dieser Theorie gehörenden Methoden des unscharfen Schließens (Schließen im Sinne von Rückschluß ziehen) werden als Fuzzy Logic bezeichnet.

Fuzzy-Set-Theorie	Theorie der unscharfen Mengen
Fuzzy Logic	(Rück-)Schließen aus unscharfen Mengen
Fuzzy Control	Regelung mit unscharfen Mengen

Es ist zu vermuten, daß diese Unterscheidung nur Mathematiker vornehmen werden. Der Begriff Fuzzy Logic wird sich wohl als Gattungsbegriff durchsetzen, so ähnlich wie das Tempo-Taschentuch für alle Papiertaschentücher. Interessant in diesem Zusammenhang ist, daß die zwei weiter unten noch näher beschriebenen Bücher von Thomas Tilli über das Thema ebenfalls in ihrem Titel die Bezeichnung Fuzzy Logic verwenden. Auch wir werden von diesem Brauch nicht abweichen.

Sind Sie Fuzzy-Logiker?

Bevor Sie sich im Detail mit dem Thema der Abschnittsüberschrift beschäftigen, sollten Sie unseren kleinen Uhrentest absolvieren. Er sagt Ihnen, aus welchem Grund Sie sich mit dem Thema Fuzzy Logic befassen sollten. Nehmen Sie diesen Test aber bitte nicht zu ernst. Denn punktuelle Genauigkeit ist, wie wir von der Fuzzy Logic lernen werden, nicht das letzte aller Ziele. Haben Sie eine Armbanduhr oder eine Taschenuhr? Dann nehmen Sie diese einmal zur Hand. Hat Ihre Uhr Zeiger oder zeigt sie die Zeit digital mit wechselnden Ziffern an? Oder tut sie sogar beides?

Wenn Ihre Uhr nur Zeiger hat, dann sind Sie Fuzzy-Logiker. Dann müssen Sie sich mit dem in diesem Abschnitt dargestellten Thema beschäftigen. Denn die Logik der fließenden Grenzen ist Ihre Lebensmaxime. Zeigt Ihre Uhr digital die Zeit an? Dann sollten Sie sich mit dem Thema Fuzzy Logic unbedingt beschäftigen. Sie sind ein Mensch, der (noch) daran glaubt, daß Pünktlichkeit und Genauigkeit die oberen Maximen des menschlichen Lebens sind oder zumindest sein sollten.

Sie werden erfahren, warum vieles auf dieser Welt starre Grenzen nicht verträgt. Zeigt Ihre Uhr auf beide Arten die Zeit an? Dann gehören Sie zu den Wanderern zwischen den Welten. Dann wollen Sie immer wieder exakt sein und sind sich gleichzeitig der menschlichen Grenzen bewußt. Auch Ihnen bringt die Beschäftigung mit der Fuzzy Logic wichtige zusätzliche Informationen. Und vielleicht fällt Ihnen am Ende die Entscheidung Analoguhr mit Zeigern oder Digitaluhr mit Ziffernanzeige doch etwas leichter.

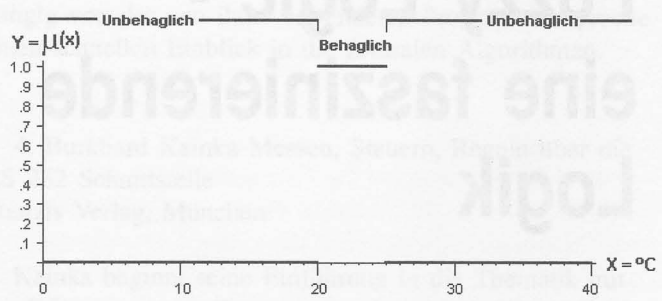
Schon Beaufort kannte Fuzzy Logic

In den einschlägigen Artikeln wird immer wieder das Beispiel der Temperatur und der menschlichen Begriffe "es ist kalt, "es ist warm" oder "es ist heiß" als Beispiel dafür verwendet, daß wir Menschen keine exakten Temperaturwerte, sondern bestimmte Temperaturbereiche meinen, wenn wir unsere Begriffe verwenden. Dieses Prinzip der Bereichsdefinition mit fließenden Grenzen hat der engliche Seemann Francis Beaufort in einer Tabelle der Windgeschwindigkeiten verwendet, die auch heute noch für die Bezeichnung von Windstärken verwendet wird. Die Tabelle ist ein typisches Beispiel dafür, wie sinnvoll es ist, anstelle der exakten Angabe Geschwindigkeit = Meter pro Sekunde "linguistische Werte" zu setzen wie Windstille, leichte Brise, Sturm, Orkan oder ähnliches.

Stärke nach Beaufort	Bezeichnung	Geschwindigkeit m/s	Auswirkung (im Binnenland)
0	Windstille	0 - 0,2	Rauch steigt senkrecht empor
1	leichter Zug	0,3 - 1,5	Windrichtung nur durch Rauch erkennbar
2	leichte Brise	1,6 - 3,3	Blätter säuseln
3	schwache Brise	3,4 - 5,4	Blätter und dünne Zweige bewegen sich
4	mäßige Brise	5,5 - 7,9	Bewegt dünne Äste, hebt Staub
5	frische Brise	8,0 - 10,7	Kleine Bäume schwanken
6	starker Wind	10,8 - 13,8	Pfeifen an Drahtleitungen
7	steifer Wind	13,9 - 17,1	Fühlbare Hemmung beim Gehen
8	stürmischer Wind	17,2 - 20,7	Bricht Zweige von Bäumen
9	Sturm	20,8 - 24,4	Kleinere Schäden an Häusern und Dächern
10	schwerer Sturm	24,5 - 28,4	Entwurzelt Bäume, schwere Schäden
11	orkanartiger Sturm	28,7 - 32,6	Verbreitete schwere Sturmschäden
12	Orkan	>=32,7	schwere Verwüstungen

Anmerkungen: Der Orkan wird inzwischen in den Windstärken 12 bis 17 angegeben. Der Bereich der Auswirkungen auf dem Meer wurde weggelassen.

Windskala



Scharfe Menge Behaglichkeit

Wie Sie aus der Tabelle "Windskala" sehen, ist jedem linguistischen Wert ein Geschwindigkeitsbereich zugeordnet. Damit hat Beaufort schon eine der Regeln der Fuzzy Logic vorweggenommen. Danach werden Grenzen zwischen Bereichen nicht nur durch einfache Zahlenwerte fixiert, sondern durch Zahlenwerte und zu ihnen gehörende Zugehörigkeitswerte. Die dem jeweiligen linguistischen Wert (Windstille ...) zugeordneten Zahlenbereiche kann man als unscharfe Zahlen (siehe unten) ansehen. Übrigens ist an dem Beispiel der Beaufort-Tabelle auch aufzeigbar, daß im menschlichen Maßbereich die beiden linguistischen Variablen Bezeichnung und Auswirkung wechselseitig verwendet werden können. Man kann also sagen: "Windstille ist , wenn..." oder "Wenn Windstille ist, dann...".

Eine zusätzliche Begriffsdefinition ist hier angebracht. Wir haben im vorhergehenden Text gelegentlich die Bezeichnung linguistischer Wert verwendet. Damit bezeichnen wir einen durch eine verbale Bezeichnung gekennzeichneten Wert (zum Beispiel Windstille), dem ein definierter Wertebereich (Windgeschwindigkeit 0,0 bis 0,2 m/s) zugeordnet wird. Linguistische Variable nennen wir eine verbale Bezeichnung, die linguistische Werte anstelle von Zahlenwerten repräsentiert.

Die Zugehörigkeit

In der Fuzzy Logic bilden alle Werte immer ein Wertepaar. Zu einem Wert gehört der Zugehörigkeitswert, der etwas darüber aussagt, wie nah der Wert an der Aussage "100prozentig dazugehörend" ist. Die Zugehörigkeitswerte können sich linear oder einer Funktionskurve folgend im Wertbereich zwischen 0 und 1, also zwischen 0% dazugehörend und 100% dazugehörend bewegen. Eine Unterteilung in 1/10er Einheiten ist allgemein üblich (0.1, 0.2 ...). Als Fuzzy-Set oder auch unscharfe Menge bezeichnet man eine geordnete Menge von Paaren aus Wert und Zugehörigkeitswert.
Die mathematische Schreibweise ist:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

Da in dieser Definition einige Zeichen stehen, die dem Nichtmathematiker nicht immer präsent sein müssen, folgt hier die verbale Zusammenfassung: Die Menge A besteht aus den Wertepaaren x und dem Zugehörigkeiswert My von x in A, wobei alle x Elemente der Menge X sind. Es geht also bei der Fuzzy Logic um Mengen und dazu gehörende Elemente. Die Definition sei hier nur dargestellt, um anzuzeigen, daß Fuzzy Logic ein mathematisch definiertes System ist.

Da dies jetzt ein wenig trocken war, wollen wir die Bildung von Fuzzy Sets einmal an einem Beispiel verständlich zu machen versuchen. Nehmen Sie einmal an, Sie würden gefragt, bei welcher Temperatur Sie sich am wohlsten fühlen. Ihre Antwort könnte lauten bei 20 bis 25 Grad Celsius. Hier haben wir es mit einer zunächst scharfen Grenzziehung zwischen den Bereichen der Behaglichkeit und denen der Unbehaglichkeit zu tun.

Das Bild "Scharfe Menge Behaglichkeit" zeigt die Zusammenhänge grafisch. Bitte beachten Sie, daß die Temperaturwerte auf der X-Achse, die entsprechenden Zugehörigkeitswerte auf der Y-Achse aufgetragen sind. Wenn Sie aber darüber nachdenken, werden Sie feststellen, daß für Sie diese Grenzen gar nicht so starr sind. Denn beim Übergang vom Winter zum Frühling oder beim Wechsel zwischen Außen und Innen werden Sie im Winter schon niedrigere Temperaturen als angenehm empfinden. Umgekehrt wird die obere Grenze im Sommer je nachdem, wie warm es draußen ist, auch fließend sein. Die Begrenzung zwischen unwohl- und wohlfühlen wird verwischt, wird unscharf.

Angemessener wäre die differenzierende Frage: "Wie wohl fühlen Sie sich bei...?" Jetzt sind Antworten zu erwarten wie "Ich fühle mich sehr wohl", "noch nicht so ganz wohl", "schon fast wohl", "unwohl" ... Nehmen wir an, Ihre Antworten weisen unterschiedliche Formulierungen und Behaglichkeitsstufen aus. Dann kann man diese Stufen zum Beispiel linear als Werte zwischen 0 = Unbehaglich bis 1 = Sehr behaglich darstellen. Mit anderen Worten, die Behaglichkeitstufen lassen sich prozentual bestimmten Temperaturwerten zuordnen. Wir erhalten also Wertepaare wie 150/0%, 160/20%, 170/40%, 180/60%, 190/80%, 200/100% ... Den linguistischen Werten "unbehaglich" ... zugeordnet könnte sich dann die in der folgenden Tabelle dargestellte Kombination ergeben.

Linguistischer Wert	Temperaturwerte von kalt zu warm	Temperaturwerte von warm zu kalt
Unbehaglich	15^0	30^0
etwas behaglich	16^0	29^0
es geht	17^0	28^0
na ja	18^0	27^0
nicht schlecht	19^0	26^0
Sehr behaglich	20^0	25^0

In der Fuzzy-Logic-Darstellung schreiben wir das ganze Fuzzy-Set

A = {(15,0.0), (16,0.2), (17,0.4), (18,0.6), (19,0.8), (20,1.0), (21,1.0) ... (29,0.2)(30,0.0)}

Die Werte aus dieser Menge von Paaren, die einen größeren Zugehörigkeitswert als 0.0 haben, werden als Support des Fuzzy-Sets bezeichnet.

$$S(A) = \{x \in X \mid \mu_A(x) \geq 0\} \qquad \text{Support}$$

Während das Fuzzy-Set die Wertepaare enthält, befinden sich im Support, da er für die Zugehörigkeit die Grenze 0 definiert hat, nur die Werte, die diesem Zugehörigkeitswert entsprechen. Es ist demnach

S(A) =
{16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29}

Wenn man alle im Fuzzy-Set enthaltenen Wertepaare in ein Diagramm einträgt und die Punkte verbindet, ergibt sich die im Bild "Fuzzy-Menge Behaglichkeit" dargestellte Trapezform. Man bezeichnet ein solches Fuzzy-Set als Fuzzy-Trapez-Set.

Der Support erfaßt, wie in dem Bild sehr deutlich wird, alle Temperaturwerte, die im weiteren Sinne zum Bereich der Behaglichkeit gezählt werden können. Dieser Support reicht von 160 bis 290 oder genauer, da wir die Werte als Punkte auf einer Linie ansehen, von 15,0010 bis 29,9990. In diesem Bild haben wir neben dem Support auch die Linie für den Alpha-Level eingetragen. Alpha-Level können für jeden Zugehörigkeitswert größer als 0 ermittelt werden. Wenn sie Werte zusammenfassen, die einen Zugehörigkeitswert haben, der gleich oder größer als ein als Untergrenze festgelegter Alphawert ist, dann spricht man von einem Alpha-Level-Set.

Werden nur über dem Grenzwert liegende Zugehörigkeitswerte berücksichtigt, spricht man von einem strengen Alpha-Level-Set. Die allgemeinen Definitionen für beide Formen von Alpha-Level-Sets sind:

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\} \qquad \alpha\text{-Level-Set}$$
$$A_{\alpha'} = \{x \in X \mid \mu_A(x) > \alpha\} \qquad \text{strenges } \alpha\text{-Level-Set}$$

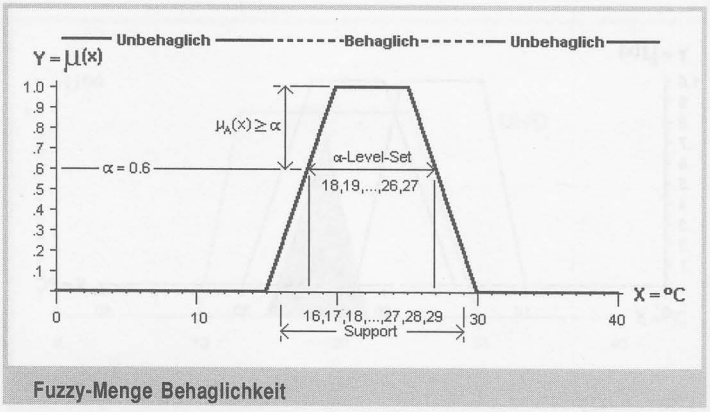
Auch für diese Definitionen soll hier wieder eine verbale Interpretation erfolgen: Ein Alpha-Set ist die Menge der Werte x, bei denen die Zugehörigkeitswerte gleich (oder größer) sind, als der vorgegebene Alpha-Wert. Die Werte für x müssen Elemente der Menge X sein.

Neben der Trapezdarstellung von Fuzzy-Sets gibt es weitere Darstellungsarten, die sich an den vorhandenen Wertepaaren orientieren. Werden die Wertepaare wie in unserem Beispiel diskret vorgegeben, so kann man statt der linearen auch die Array-Darstellung wählen.

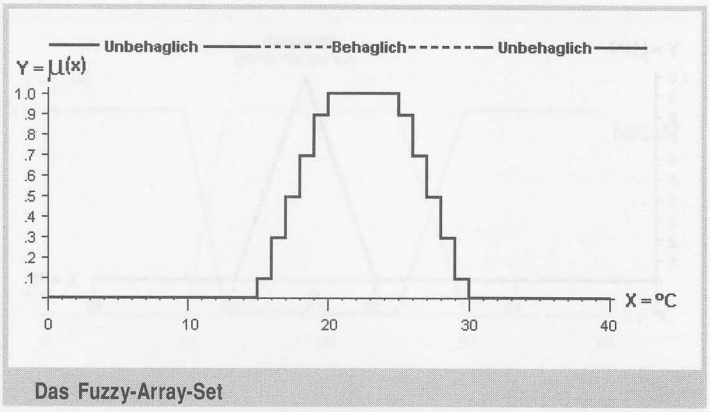
Diese zeigt unser Bild "Das Fuzzy-Array-Set". Die Werte in dieser Darstellung entsprechen denen im vorhergehenden Bild, nur daß hier alle Werte Ganzzahlwerte sind, also Werte wie 15.001 nicht zum Set gehören.

In einem anderen Fuzzy-Set können die Wertepaare vor und nach dem 100%-Bereich statt der linearen Folge jeweils einer Funktion folgen. Dann bezeichnet man dieses Fuzzy-Set als LR-Set, von Links-Rechts-Set (siehe Bild: Fuzzy-LR-Set).

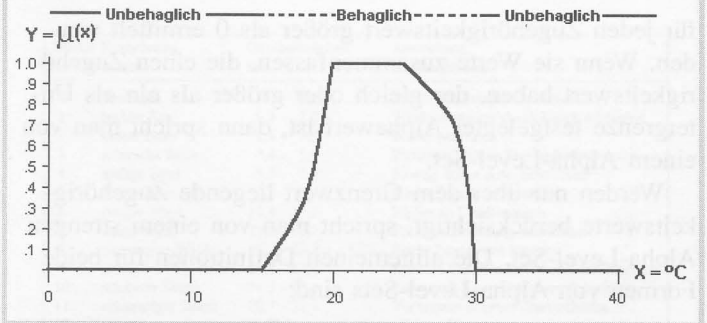
Ein weiterer Fuzzy-Set-Typ ergibt sich aus folgenden Überlegungen: Nehmen wir an, Sie haben exakt die Temperatur von 25 Grad als 100prozentig behaglich genannt. Dann ergibt die Darstellung ein Dreieck und das Fuzzy-Set ist ein Dreiecks-Set (siehe Bild: Das Fuzzy-Dreieck). Dessen grafische Darstellung zeigt gegenüber den anderen Darstellungen klar an, daß nur ein Wert den Anforderungen 100prozentig entspricht.



Fuzzy-Menge Behaglichkeit



Das Fuzzy-Array-Set



Fuzzy-LR-Set

Schauen Sie sich alle vier Abbildungen zu den Fuzzy-Sets noch einmal an. Und denken Sie bei der Betrachtung des Kurvenverlaufs daran, was die einzelnen Punkte repräsentieren: Die Stufen Ihrer Behaglichkeit. An manchen Stellen hat man das Gefühl, jetzt beginnt die Katze am Kamin zu schnurren. Und mit den Alpha-Level-Sets kann man sogar bestimmen, ab wann und bis wann sich die Katze am wohlsten fühlt.

Noch ein paar Grundlagen

Wir haben bisher nur ein einzelnes Fuzzy-Set jeweils betrachtet. Nun ist zum Beispiel die Behaglichkeit nicht nur von der Höhe der Umgebungstemperatur oder den (unscharfen) Temperaturbereichen abhängig. Die Luftfeuchtigkeit spielt dabei auch eine wichtige Rolle. Außerdem sind die Werte für die Wärmestrahlung und Windgeschwindigkeiten von Bedeutung. Alle Werte zusammen bezeichnet man als das Klima.

Klima = Temperatur
+ Feuchtigkeit
+ Wärmestrahlung
+ Windgeschwindigkeit

Es genügt also nicht, nur ein Fuzzy-Set für die Raumtemperatur zu erstellen. Auch die anderen Bereiche müssen in die Überlegungen einbezogen werden. Unterstellen wir einmal, es wären analog zu unserem Temperatur-Fuzzy-Set entsprechende für die anderen Behaglichkeitsparameter gefunden. Für diesen Fall gilt:

WENN Temperatur UND Feuchtigkeit UND Wärmestrahlung UND Windgeschwindigkeit jeweils im Behaglichkeitsbereich sind, DANN fühlen wir uns wohl.

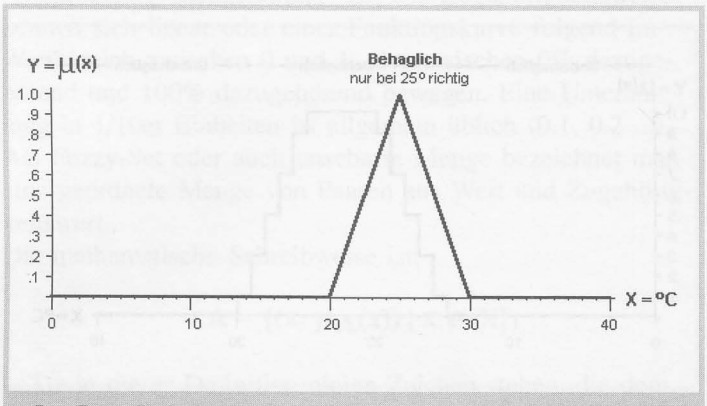
Wir haben es bei allen infragekommenden Parametern mit Fuzzy-Sets, also Mengen von Wertepaaren, zu tun. Und diese stimmen immer dort überein, wo die Teilmenge des einen Sets mit der eines anderen Sets übereinstimmt oder, wie in unserem Fall, wo die Teilmenge des einen Sets die Schnittmenge aller anderen Sets ist. Zur Verdeutlichung wollen wir unser Behaglichkeitsmodell um eine Person erweitern. Nehmen wir an, zwei Partner teilen sich eine Wohnung.

Nehmen wir weiter an, daß diese beiden Partner unterschiedliche Bereiche der Behaglichkeit haben. Der neue Partner, wir wollen ihn Partner 2 nennen, beginnt sich schon bei 100 langsam wohlzufühlen und mag maximal Temperaturen bis 220. Auch fühlt er sich schneller unbehaglich als Partner 1. Es ergeben sich dann die zwei in unserer Abbildung "Zwei Fuzzy-Sets" dargestellten Kurvenverläufe und Fuzzy-Sets.

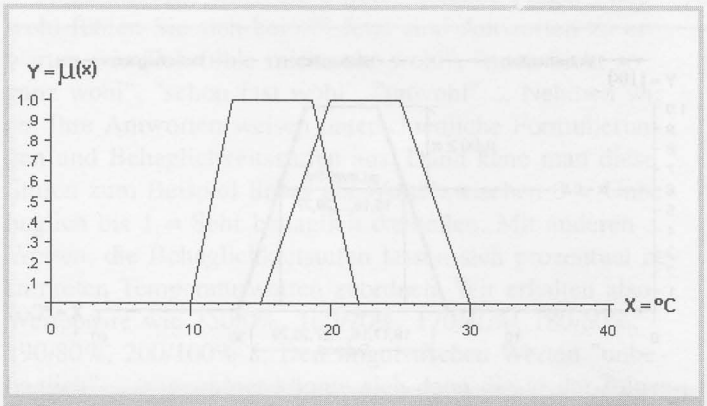
Man kann nun in der Grafik leicht feststellen, daß der Bereich, den beide Fuzzy-Sets bedecken, also das kleine Dreieck unten in der Mitte, der Temperaturbereich ist, in dem sich beide Partner noch wohlfühlen. Es zeigt sich aber auch, daß die Spitze dieser Teilmenge nicht den Zugehörigkeitswert von 1.0 erreicht. Das bedeutet, beide Partner werden sich einigermaßen, aber keiner der Partner wird sich wirklich 100prozentig wohlfühlen, wenn man die Raumtemperatur in diesem Bereich regelt.

Anders ist dies, wenn man eine Gesamtmenge aus beiden Fuzzy-Sets bildet. Dann ist das kleine Dreieck der Bereich, wo beide sich gerade so wohlfühlen, die Restmengen links und rechts davon aber die Bereiche, wo sich einer von beiden ganz wohl und der andere ganz unwohl fühlt.

Unter diesen Umständen sollten die beiden Partner prüfen, ob ihre Bindung von Dauer sein kann. Zur mathematischen Ermittlung, wann sich beide Partner wohlfühlen, wann sich einer von ihnen wohlfühlt oder wann sich gar keiner von ihnen wohlfühlt, bedarf es zusätzlicher Mengenoperatoren. Der in den USA lebende persische Wissenschaftler Lofti A. Zadeh hat schon 1965 die theoretischen



Das Fuzzy-Dreieck



Zwei Fuzzy-Sets

Grundlagen der Fuzzy-Theorie entwickelt. In unserem Bild am Ende dieses Kapitels sehen Sie Lofti A.Zadeh mit dem Fuzzy-Auto der INFORM (nähere Informationen siehe unten). Ihm ging es darum, all die Vorgänge mathematisch faßbar zu machen, bei denen nur unvollständige Informationen, unvollständige Datensätze ermittelbar sind oder vorliegen, aber dennoch Schlüsse gezogen werden sollen. Er entwickelte die mathematischen Grundlagen für die Arbeit mit Mengen, die er Fuzzy-Sets nannte. Es geht dabei immer um Mengen von Daten, die typisch aber nicht vollständig sein müssen. Es gelten deshalb die Regeln der Mengenlehre. Zadeh hat drei elementare Mengenopertoren vorgeschlagen.

Minimumoperator — Mit ihm wird der Durchschnitt

$$(C = A \cap B)$$

zweier Fuzzy-Sets ermittelt. Der Durchschnitt ist die Teilmenge eines Sets, die gleichzeitig Teilmenge des anderen ist. Der Durchschnitt wird als logisch

$$\mu_{A \cap B}(x) = \min \{\mu_A(x), \mu_B(x) \mid x \in X\}$$

UND interpretiert.

Maximumoperator — Mit ihm ermittelt man die Vereinigung zweier Fuzzy-Sets , also die Menge, die aus beiden Sets

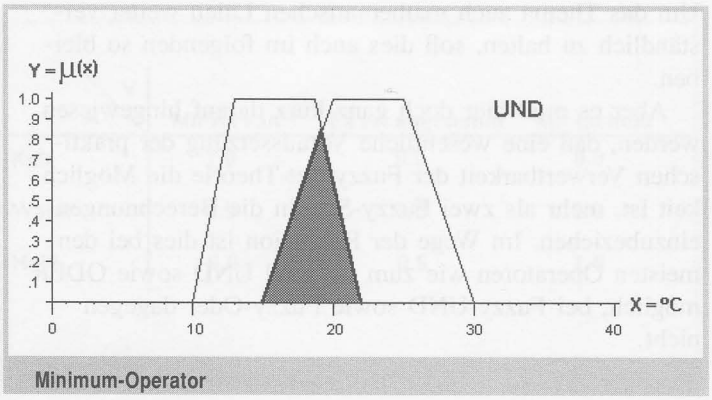
$$(C = A \cup B)$$

als Gesamtmenge entsteht. Die Vereinigung wird als logisch ODER interpretiert.

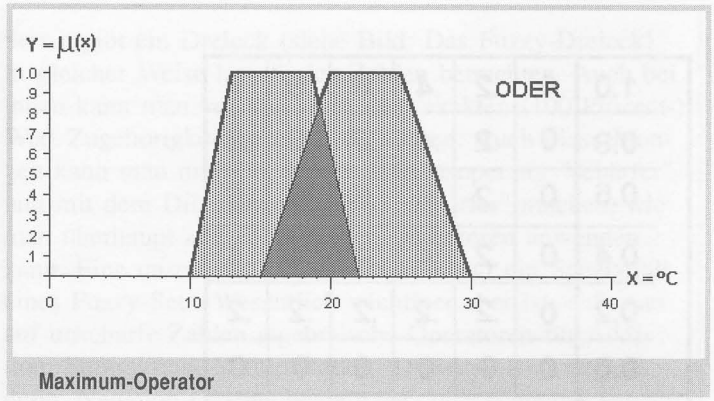
$$\mu_{A \cup B}(x) = \max \{\mu_A(x), \mu_B(x) \mid x \in X\}$$

Komplement — ist der Kehrwert eines Fuzzy-Sets. Das Komplement wird als logisch

$$\mu_C(x) = \{1 - \mu_A(x) \mid x \in X\}$$



Minimum-Operator



Maximum-Operator

NICHT interpretiert.

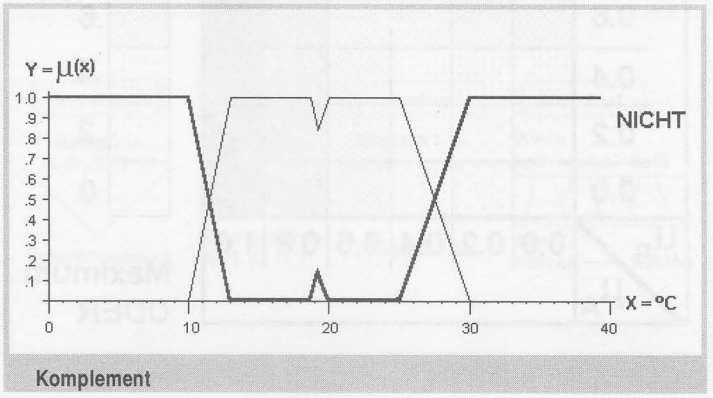
In unserem Beispiel haben wir zuvor schon die Bedeutung der Minimum- und Maximumwerte anhand der grafischen Darstellung festgestellt. Die Operatoren auf unserer Darstellung sind:

Minimumoperator — ergibt den Bereich, in dem sich der Eine UND der Andere wohlfühlen. Es ergibt sich das Fuzzy-Set P1_UND_P2

Maximumoperator — ergibt den Bereich, in dem sich der Eine ODER der Andere wohlfühlt oder wo sich beide wohlfühlen. Das Ergebnis ist das Fuzzy-Set P1_ODER_P2.

Komplement — ergibt den Bereich, wo sich beide NICHT wohlfühlen. Das Ergebnis ist t das Fuzzy-Set NICHT (P1_ODER_P2).

Wir haben mit den eben dargestellten Minimum- und Maximumoperatoren aus jeweils zwei Fuzzy-Sets ein neues Set geschaffen.

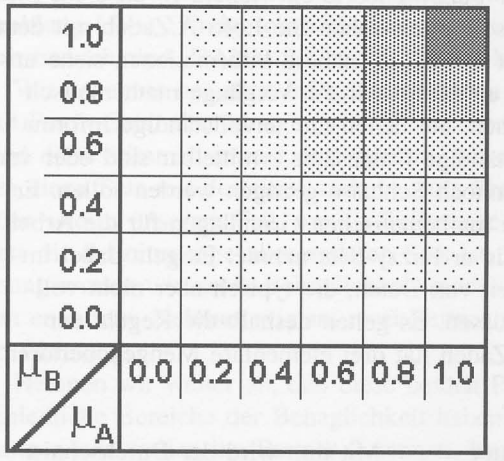


Komplement

1.0	0	.2	.4	.6	.8	1
0.8	0	.2	.4	.6	.8	.8
0.6	0	.2	.4	.6	.6	.6
0.4	0	.2	.4	.4	.4	.4
0.2	0	.2	.2	.2	.2	.2
0.0	0	0	0	0	0	0
μ_B / μ_A	0.0	0.2	0.4	0.6	0.8	1.0

Minimum
UND

Die Fuzzy-UND-Matrix



Die Fuzzy-ODER-Matrixgrafik

1984 wurden von B.Werners für die allgemeine Lösung dieser Aufgabe zusätzlich die Operatoren Fuzzy-UND sowie Fuzzy-ODER vorgeschlagen. Diese beiden Operatoren kombinieren den Minimum- beziehungsweise Maximumoperator mit dem arithmetischen Mittel der beiden Fuzzy-Sets.

Die Definitionen lauten:

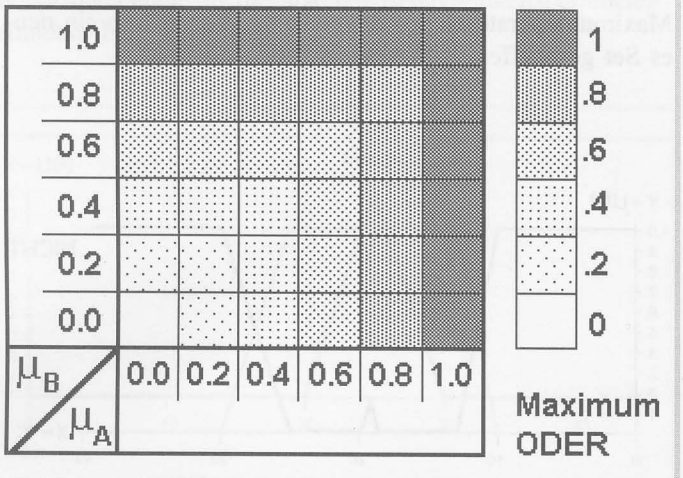
$$\mu(\mu_A(x), \mu_B(x)) = \gamma \min(\mu_A(x), \mu_B(x)) + 1/2(1-\gamma)(\mu_A(x), \mu_B(x))$$
$$\gamma \in [0, 1]$$

Fuzzy-UND

$$\mu(\mu_A(x), \mu_B(x)) = \gamma \max(\mu_A(x), \mu_B(x)) + 1/2(1-\gamma)(\mu_A(x), \mu_B(x))$$
$$\gamma \in [0, 1]$$

Fuzzy-ODER

Für die beiden Fuzzy-Operatoren sind die Minimum- beziehungsweise Maximumoperatoren der Grenzwert, der sich ergibt, wenn Gamma = 1 gesetzt wird. Dann ergibt die aus zwei Teilen bestehende Funktion im zweiten Teil den Wert Null, nur der erste Teil bleibt und entspricht der Definition des Minimum- beziehungsweise Maximumoperators. Wird Gamma = 0 gesetzt, dann verschwindet der erste Teil der Funktion und das arithmetische Mittel beider Fuzzy-Sets bleibt als Ergebnis.



Die Fuzzy-UND-Matrixgrafik

Die Matrixdarstellung

Die Darstellung der Fuzzy-Set-Operatoren erfolgte bisher grafisch mit Linien. Man kann natürlich bei den sich ergebenden zweidimensionalen Feldern auch die Darstellung in Form einer Matrix, einer Tabelle, wählen. Die Matrix für den Maximum-(UND-)Operator zeigt unsere Tabelle "Die Fuzzy-UND-Matrix".

Wenn man jetzt zur besseren Übersicht Zugehörigkeitswerte in der Matrix grafisch durch unterschiedliche Rasterung der Fläche darstellt, so ergeben sich für ein mit dem Minimum-Operator oder ein mit dem Maximum-Operator gebildetes neues Fuzzy-Set die in "Fuzzy-UND-Matrixgrafik" oder "Fuzzy-ODER-Matrixgrafik" gezeigten Darstellungen. Diese Darstellungsart kann man natürlich auch auf alle anderen Operatoren anwenden, bei denen aus zwei Fuzzy-Sets ein neues gebildet wird. Dem aufmerksamen Leser wird aufgefallen sein, daß bisher maximal zwei Fuzzy-Sets gemeinsam in die Betrachtungen einbezogen sind. Um das Thema auch mathematischen Laien weiter verständlich zu halten, soll dies auch im folgenden so bleiben.

Aber es muß hier doch ganz kurz darauf hingewiesen werden, daß eine wesentliche Voraussetzung der praktischen Verwertbarkeit der Fuzzy-Set-Theorie die Möglichkeit ist, mehr als zwei Fuzzy-Sets in die Berechnungen einzubeziehen. Im Wege der Rekursion ist dies bei den meisten Operatoren wie zum Beispiel UND sowie ODER möglich, bei Fuzzy-UND sowie Fuzzy-Oder dagegen nicht.

Wenn Sie sich stärker für die Thematik interessieren, so sollten Sie Ihr Wissen mit den am Ende dieses Abschnitts beschriebenen Büchern von T.Tilli vertiefen. Am Ende dieses nun doch mit etwas mehr Mathematik gefüllten Abschnitts sollen noch einige sogenannte Modifikatoren kurz dargestellt werden. Wir hatten in unserem Zwei-Partner-Beispiel festgestellt, daß die Gemeinsamkeiten in bezug auf das Behaglichkeitsgefühl unter einem Zugehörigkeitswert von 1.0 liegen. Man kann das dort ermittelte neue Fuzzy-Set "normalisieren". Dies bedeutet, daß man das neue Set so umrechnet, daß es den Zugehörigkeitsbereich wieder bis zum Maximalwert von 1.0, also 100prozentig zugehörig, ausfüllt.

Konzentration

Bei der Konzentration werden die Zugehörigkeitswerte mit sich selbst multipliziert. Dies ergibt eine stärkere Konzentration des Sets, es wird "schärfer". Beispiel: 0.9 ergäbe sich 0.9*0.9 = 0.81, 0.82 = 0.64, 0.72 = 0.49 ...

$$\mathbf{Con}(\mu(x)) = (\mu(x))^2$$

Normalisierung

Dazu müssen alle Zugehörigkeitswerte durch den größten Zugehörigkeitswert dividiert werden. In unserem Beispiel würde sich so für den Zugehörigkeitswert 0.9 (dividiert durch 0.9) der neue Zugehörigkeitswert 1.0 ergeben.

Dilation

Der Dilationsoperator ist die Umkehrung des Konzentrations-Operators. Bei seiner Anwendung wird das Set "unschärfer". Hier wird - statt quadriert - die Quadratwurzel gezogen.

$$\mathbf{Dil}(\mu(x)) = (\mu(x))^{1/2}$$

Die unscharfen Zahlen

Nachdem wir einen (kleinen) Teil der zur "Fuzzifizierung" einsetzbaren Operatoren besprochen haben, soll hier noch kurz auf den Bereich der unscharfen Zahlen eingegangen werden. Wir haben am Anfang unserer Betrachtungen ein Fuzzy-Set definiert, bei dem nur ein einziger Wert die Zugehörigkeit 100 Prozent hat. Die Abbildung dieses Sets ergibt ein Dreieck (siehe Bild: Das Fuzzy-Dreieck). In gleicher Weise lassen sich Zahlen betrachten. Auch bei ihnen kann man vor und nach dem exakten (100-Prozent-) Wert Zugehörigkeitsbereiche definieren. Auch diese Mengen kann man mit dem Konzentrationsoperator "schärfer" und mit dem Dilationsoperator "unschärfer" machen, wie man überhaupt alle dargestellten Operatoren anwenden kann. Eine unscharfe Zahl ist insofern nur ein Spezialfall eines Fuzzy-Sets. Wesentlich wichtiger aber ist, daß man auf unscharfe Zahlen algebraische Operatoren für Addition, Subtraktion, Multiplikation, Division, Kehrwertbildung, Negation (Vorzeichenumkehr) und ähnliches anwenden kann. Man kann mit unscharfen Zahlen rechnen. Es gibt sogar eine darauf aufbauende Differential- und Intergralrechnung.

Unscharfe Relationen

Für die Entwicklung unscharfer Regler und Expertensysteme sind unscharfe Relationen von großer Bedeutung. Eine unscharfe Relation ist beispielsweise gegeben, wenn wir die Behaglichkeits-Variable nicht auf scharf definierte Werte auf der Temperaturskala, sondern auf linguistisch definierte Werte wie kalt, warm und heiß beziehen. In der Matrix "Unscharfe Behaglichkeitsrelationen" sind die Zusammenhänge dargestellt. Unscharfe Relationen sind, wie die Matrix zeigt, Fuzzy-Sets in Produkträumen (X * Y-Raum). Das bedeutet, daß auch alle auf "normale" eindimensionale (X-Raum) Fuzzy-Sets anwendbaren Operatoren darauf angewendet werden können.

Anhand der beiden Operatoren zur Ermittlung von Minimum und Maximum sieht man, daß sich Operatoren für Produkträume nur in bezug auf den zweidimensionalen Produktraum von den weiter oben dargestellten eindimensionalen Operatoren unterscheiden.

Minimum, Durchschnitt, UND

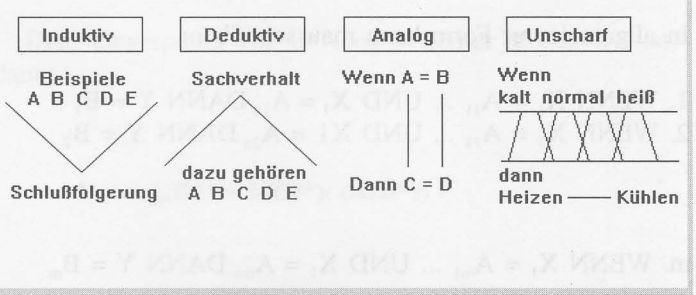
$$\mu_{A \cap B}(x,y) = \min \{\mu_A(x,y), \mu_B(x,y) \mid (x,y) \in X*Y\}$$

Maximum, Vereinigung, ODER

$$\mu_{A \cup B}(x,y) = \max \{\mu_A(x,y), \mu_B(x,y) \mid (x,y) \in X*Y\}$$

X \ Y	Mir ist kalt	Es ist angenehm	Mir ist heiß
Kalt	1.0	0.5	0.0
Warm	0.5	1.0	0.5
Heiß	0.0	0.5	1.0

Unscharfe Behaglichkeitsrelationen



Die Arten des Schließens

Die praktische Anwendung der Fuzzy-Logic

In den vorhergehenden Zeilen dieses Abschnitts haben wir versucht, die Grundlagen der Fuzzy-Set-Theorie ohne allzugroße Befrachtung mit mathematischen Definitionen darzustellen. Dies ist natürlich immer eine Gradwanderung zwischen Verständlichkeit und Exaktheit. Wir hoffen, daß es uns gelungen ist, dem Leser, der sich neu mit Fuzzy Logic beschäftigt, ein Gefühl für die Thematik zu vermitteln, ohne den Leser zu verärgern, der sich schon auskennt. Wenn uns das gelungen ist, dann haben wir jetzt gemeinsam den Punkt erreicht, wo wir uns fragen müssen, wozu das Ganze dient. Es kann ja wohl nicht Ziel der ganzen Erläuterungen gewesen sein, festzustellen, daß zwei Partner miteinander Probleme haben werden. Wozu also Fuzzy Logic?

Vielleicht erinnern Sie sich an unsere Erläuterungen ganz am Anfang dieses Abschnitts. Wir wiesen darauf hin, daß der Oberbegriff für das hier behandelte Thema eigentlich nicht Fuzzy Logic, sondern Fuzzy-Set-Teorie lauten müßte. Die Fuzzy Logic ist der Teilbereich dieses Gesamtkomplexes, der sich mit dem unscharfen Schließen beschäftigt. Und darüber haben wir bisher gar nicht gesprochen.

Ein Beispiel dafür ist die Regel:

WENN Temperatur = Kalt DANN Heizen

wobei die Variable Temperatur zum Beispiel die Werte kalt, normal oder heiß annehmen kann und die zugeordneten Aktionen "Heizen", "Ofen und Klimaanlage aus" sowie "Kühlen" sein könnten. Wie an diesem Beispiel zu ersehen ist, ist unscharfes Schließen eine typisch menschliche Art des Schließens. In unserer Abbildung "Die Arten des Schließens" haben wir drei weitere Arten des Schließens und das unscharfe Schließen in eine grafische Darstellung gebracht. Wir haben soeben eine Regel aufgestellt, die gilt, wenn der Fakt Temperatur = Kalt zutreffend ist. Für die drei unterstellten linguistischen Werte, die die Temperatur annehmen kann, gelten die Regeln:

WENN Temperatur = Kalt DANN Ofen an, Klimaanlage aus: Heizen
WENN Temperatur = Normal DANN Ofen aus, Klimaanlage aus: Keines von beiden
WENN Temperatur = Warm DANN Ofen aus, Klimaanlage an: Kühlen

In allgemeinerer Form kann man schreiben:

1. WENN $X_1 = A_{11}$ ... UND $X_1 = A_{1n}$ DANN $Y = B_1$
2. WENN $X_1 = A_{21}$ ... UND X1 = A_{2n} DANN $Y = B_2$

.
.
.

m. WENN $X_1 = A_{m1}$ ... UND $X_1 = A_{mn}$ DANN $Y = B_m$

Alle Regeln zur Regelung eines bestimmten Prozesses zusammen bezeichnet man als die Regelbasis.

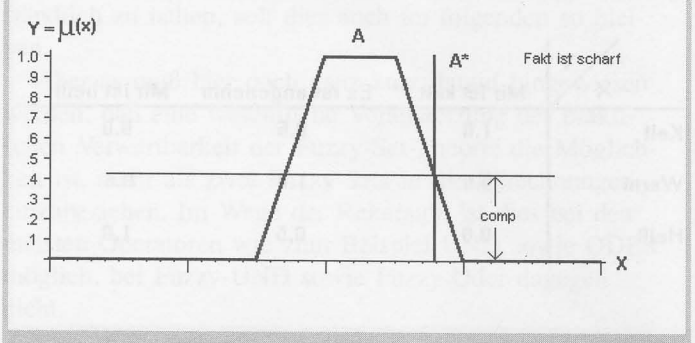
Die Formulierungen WENN X = A ... DANN Y = B sind die Regeln, X = A sind die Begriffe, Fakten sind die tatsächlichen Gegebenheiten, die wir im folgenden, da sie ja den Begriffen entsprechen, mit einem zusätzlichen Stern also A1* ... An* kennzeichnen. Aufgrund der Übereinstimmung der Fakten mit den Begriffen der Regel ergeben sich die Schlußfolgerungen Y = B. Wir arbeiten hier mit Regeln und Fakten, aus denen sich Schlüsse im Sinne von Schlußfolgerungen ergeben. Alle drei Bereiche können im Sinne der Definition scharf sein oder unscharf. Man kann in Regeln und Fakten oder jeweils nur einem von ihnen Fuzzy-Sets oder scharfe Werte verwenden.

Zusätzlich können die Schlußfolgerungen in exakten Anweisungen (scharfen Werten) oder in Fuzzy-Sets mit entsprechender Modifikation erfolgen. Die Aufgabe, aus unscharfen Regeln und Fakten Schlüsse zu ziehen, nennt man unscharfes Schließen. Die letzte Form des unscharfen Schließens, bei der sowohl die Fakten als auch die Schlußfolgerungen unscharf sind, bezeichnet man auch als plausibles Schließen. Mit Hilfe der Fuzzy Logic im engeren Sinn lassen sich Regelbasen erstellen, die industrielle Prozesse regeln können. Bei diesen Prozessen liegen sehr oft unscharfe oder sich schnell ändernde, dynamische Prozeßdaten vor. In beiden Fällen ist die Erstellung eines mathematischen Modells nur mit Hilfe der Regeln der Fuzzy-Set-Theorie überhaupt oder zu vertretbaren Kosten realisierbar. Die Regeln können in Hardware umgesetzt oder mit Software verarbeitet werden. Auf diese Möglichkeit gehen wir unten noch näher ein.

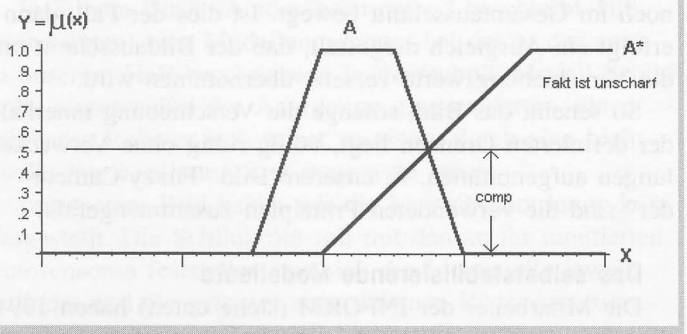
Die Kompatibilität von Fuzzy-Sets

Als nächste Überlegung kommt jetzt die Frage nach dem Grad der Übereinstimmung des jeweils aktuellen Fakt mit Einzelbegriffen in einer Regel. Diese Übereinstimmung bezeichnet man als Kompatibilität. Zu wieviel Prozent stimmt der Fakt mit dem Begriff in der Regel überein? Die Einzelkompatibilität der Fakten, in der Regel m, ergibt sich aus:

$am_1 = op(A_1^* , Am_1)$



Kompatibilität mit scharfenm Fakt



Kompatibilität mit unscharfem Fakt

$am_2 = op(A_2^* , Am_2)$

.

.

.

$a_{mn} = OP(A_n^* , A_{mn})$

Kompatibilität mit scharfem Fakt

Verbal formuliert bedeutet dies: Der Grad der Übereinstimmung amn ergibt sich aus dem Prozentwert, zu dem der Fakt An* mit dem Regelbegriff Amn übereinstimmt. Der Operator OP in der eben dargestellten Funktion ist ein Operator, der den Grad der Übereinstimmung, die Kompatibilität von Fakten und Begriffen ermittelt.

Unsere beiden Abbildungen "Kompatibilität mit scharfem Fakt" und "Kompatibilität mit unscharfem Fakt" zeigen die grafische Aufbereitung. Wenn Sie sich an unser Beispiel mit der Behaglichkeit der zwei Partner erinnern, sehen Sie, daß wir dort einen Übereinstimmungswert ermittelt und festgestellt haben, daß die Übereinstimmung maximal 0.9 also 90prozentig war. Analog ermitteln wir hier die Kompatibilität von Fakten zur Regel. Auch hier ist der gefundene Wert für einen unscharfen Fakt gleich dem Maximum gemeinsamer Zugehörigkeit, für einen scharfen Fakt gleich dem X-Wert bei A* = A. Die Kompatibilität eines Fakts mit einer Regel kann also mit dem Minimumoperator (UND) ermittelt werden.

Nehmen wir an, eine Regel enthält mehrere Fakten und wir haben die Kompatibilität der Fakten mit der Regel ermittelt. Wir haben also zum Beispiel festgestellt, die Übereinstimmung bei Fakt 1 ist 0.9, die bei Fakt 2 ergibt 1.0 ... Dann können wir die Gesamtkompatibilität aller Fakten mit der Regel mit dem für die Situation angemessenen Mengenoperator errechnen. Wenn alle Begriffe der Regel mit UND verknüpft sind, könnte das also der UND-Operator (Minimumoperator), bei Verknüpfung mit ODER der ODER-Operator (Maximumoperator) sein.

Aufgrund der Gesamtkompatibilität am aller relevanten Fakten mit der entsprechenden Regel kann man jetzt für jede Regel einen Schluß (Inferenz) Bm* ziehen, der aufgrund der Übereinstimmung der Fakten mit den Begriffen der Regel (Bm) ein unscharfer Schluß ist, wenn die Gesamtkompatibilität kleiner als 1 ist. Es ist

$B_m^* = INFERENZ(a_m, B_m)$

Aus den einzelnen Schlüssen läßt sich dann auch noch der Gesamtschluß

$B^* = OP(B_1, ..., B_m)$

ziehen, wobei der Operator OP wieder ein UND-Operator oder ein ODER-Operator ist. Der Vollständigkeit halber sei gesagt, daß der Operator OP auch ein parametrisierter Operator sein könnte, ein Typus von Operator, auf den wir hier nicht eingehen.

Eine letzte Überlegung soll diesen gewichtigen und wichtigen Abschnitt unserer Annäherung an Fuzzy Logic abschließen. Wir sind bisher immer noch davon ausgegangen, daß unsere Regeln scharf sind. Für unscharfe Regeln ist am Ende noch eine Operation erforderlich, die Berücksichtigung des Sicherheitsfaktors der Regeln (= c von englisch: certainty-factor). Auch dieser kann einen Wert zwischen 0 und 1 annehmen. Ein Wert von cm = 0 sagt, die Regel m ist absolut unsicher, sie darf zum Gesamtergebnis nicht beitragen, eine Regel mit dem Sicherheitsfaktor cm = 1 dagegen voll. Alle mit ihrem Sicherheitswert dazwischen liegenden Regeln tragen entsprechend diesem Grad zum Gesamtergebnis bei. Das neue Kompatibilitätsmaß ist dann:

$a_m^* = MIN(a_m, c_m)$

Defuzzifizierung

Das Ergebnis aller vorangehenden Überlegungen ist wieder ein unscharfer Wert. Nun kann man einen technischen Prozeß in der Regel nicht mit diesen unscharfen Werten steuern. Er braucht vielmehr exakte Werte. Diese lassen sich aus den bisherigen Werten ermitteln.

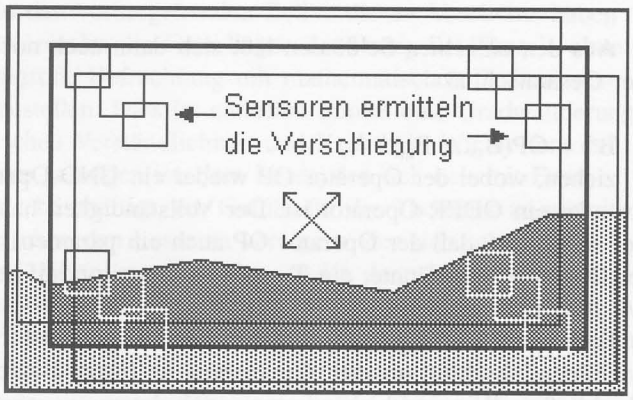
Das Wort "ermitteln" enthält schon den Schlüssel für die jetzt noch erforderlichen Operationen, es muß ein Mittelwert ermittelt werden.

Die wohl häufigste Vorgehensweise ist die Bildung des Schwerpunktes b* aus dem Gesamtschluß B*, der ja seinerseits eine gewichtete Menge darstellt. Dazu berechnet man die Gesamtfläche A und das Moment von B* wie folgt:

$$A(B^*) = \Sigma\ \mu_m(B^*)$$
$$M(B^*) = \Sigma\ x_m, \mu_m(B^*)$$

Der Schwerpunkt von B* (cog = Center of Gravity) ist dann:

$$b^* = cog(B^*) = M(B^*)/\ (A(B^*))$$



Fuzzy-Camcorder

Fuzzy Logic in der Anwendung

Es gibt inzwischen eine Reihe von realisierten oder in der Entwicklung befindlichen Anwendungen der Fuzzy Logic. Es sind dies Anwendungen im Bereich der Expertensysteme und neuronalen Netze, in der Prozeßregelung, der Fahrzeugtechnik, für Konsumartikel und weitere Anwendungsbereiche. Wir wollen hier nur exemplarisch auf drei Anwendungen näher eingehen.

Verwacklungsfreier Camcorder

Wir haben am Anfang dieses Abschnitts schon darauf hingewiesen, daß japanische Unternehmen vor allem im Bereich der Konsumentenelektronik Fuzzy-Logic-Chips einsetzen.

Wie könnte zum Beispiel die Fuzzy Logik beim Camcorder verwendet werden? Um Verwacklungen durch den Benutzer nicht auf das Videoband zu übertragen, muß zunächst ein größerer als der im Sucher angezeigte Bereich in einen Zwischenspeicher genommen werden.

Wir wollen diesen Bereich als Gesamtbereich bezeichnen. Verschiebt sich jetzt der effektiv im Sucher sichtbare Bildausschnitt dann wird dies durch Sensorblocks festgestellt.

Als nächstes wird geprüft, ob sich die Verschiebung noch im Gesamtausschnitt bewegt. Ist dies der Fall, dann erfolgt ein Ausgleich dergestalt, daß der Bildausschnitt um die Verschiebungswerte versetzt übernommen wird.

So scheint das Bild, solange die Verschiebung innerhalb der definierten Grenzen liegt, völlig ruhig ohne Verwackelungen aufgenommen. In unserem Bild "Fuzzy-Camcorder" sind die verwendeten Prinzipien zusammengefaßt.

Das selbststabilisierende Modellauto

Die Mitarbeiter der INFORM (siehe unten) haben 1991, damals noch an der RWTH in Aachen am Lehrstuhl von Professor Dr. Zimmerman, ein Modellauto vorgestellt, das sich mithilfe von Fuzzy Logic in Extremsituationen selbst stabilisiert. Es wurden zum Beispiel im Experiment Vollbremsungen bei 50 km/h durchgeführt, die das Auto zum Schleudern und Rutschen brachten.

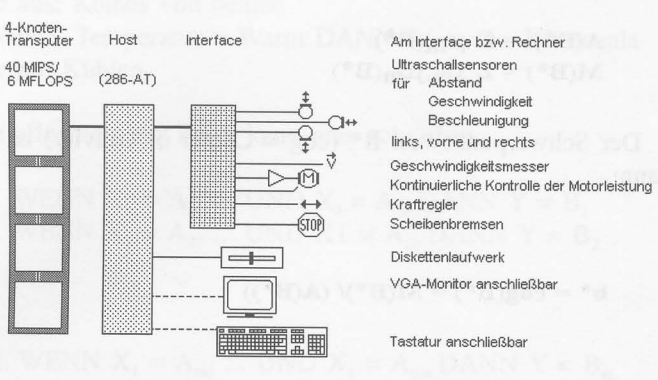
Dieses mit den herkömmlichen Methoden der Mathematik überhaupt nicht oder doch zumindest nur schwer faßbare und formulierbare Verhalten wurde mit Hilfe von Fuzzy-Sets "unscharf" berechnet und so in Steuersignale umgesetzt, so daß sich das Auto auch in physikalischen Grenzbereichen stabilisierte.

Die Sensorkontrolle und die Motorsteuerung erfolgte mit Hilfe eines 286er-Rechners. Das Steuerverhalten selbst wurde von vier in einem Knoten verbundenen Transputern berechnet. Der Zyklus für den Regelalgorithmus wurde auf 10 Mikrosekunden festgelegt.

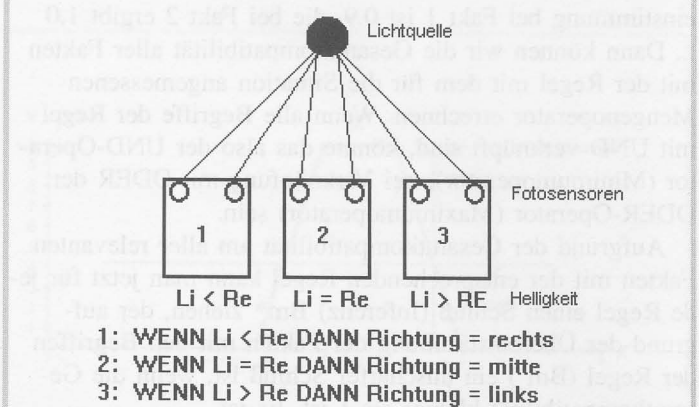
In unserem Bild "Das selbststabilisierende Auto" haben wir das Zusammenwirken dargestellt.

Auf der Diskette zum Heft finden Sie ein Programm unter dem Namen "FUZZYLOG.EXE". Dies ist das von uns umbenannte Programm "AUTO.EXE", das uns von der INFORM freundlicherweise zur Verfügung gestellt wurde. Bitte beachten Sie, daß dieses Programm in einer älteren Version von fuzzyTECH, dem von der INFORM vertriebenen Entwicklungssystem, geschrieben ist.

Weitere Informationen zum Programm finden Sie zum Ende dieses Heftes in dem Abschnitt "Die Programme auf der Diskette".



Das selbststabilisierende Auto



Die fischertechnik-Schildkröte als Versuchsobjekt

Die fischertechnik-Schildkröte zur Zielverfolgung

In seinem Buch "Automatisierung ..." beschreibt Tilli (siehe unten) eine Modellanordnung, bei der er das auch in unserem Heft beschriebene fischertechnik-Modell Schildkröte einsetzt. Bei der Anordnung ging es darum, einen fahrbaren Roboter sich selbst zu einem durch eine Lichtquelle repräsentierten Ziel steuern zu lassen.

In unserem Bild haben wir die Versuchsanordnung kurz dargestellt. Die Schildkröte soll mit den an ihr montierten Fotosensoren feststellen, wo sich die Lichtquelle jeweils befindet und die getrennt angetriebenen Räder so ansteuern, daß sie sich in Richtung Lichtquelle bewegt.

An der Frontseite der Turtle hatte Tilli zwei einfache Fotowiderstände als Licht-Sensoren montiert. Es ging nun darum, das von diesen Widerständen aufgenommene Licht in digitale Werte umzusetzen. Die Fotowiderstände wurden dazu mit den Analogeingängen des fischertechnik-Interface verbunden.

Mit Hilfe der Fuzzy-Logic erfolgte jetzt eine Optimierung der Art, daß bei gleich großen Werten die Turtle geradeaus, bei unterschiedlichen Werten entsprechend nach links oder nach rechts zu steuern war. In dem von Tilli zu diesem Zweck entwickelten Fuzzy-Regler wurden die beiden linguistischen Variablen Sensor_links (SL) und Sensor_Rechts (SR) als Inputvariablen und die linguistische Variable Richtung (RI) als Ausgangsvariable festgelegt. Die aus neun Regeln bestehende Regelbasis findet man wie den Quelltext bei Tilli auf der zu dem Buch gehörenden Diskette. Schon aus diesem Grund kann das Buch dem interessierten fischertechnik-Fan empfohlen werden.

Wie geht es weiter mit Fuzzy Logic?

Es ist erstaunlich, daß in Deutschland und Europa wieder einmal so spät und so zögerlich an die Sache herangegangen wurde, daß Professor Dr. Zimmerman von der RWTH Aachen, der sich mit dem Thema seit etwa 20 Jahren intensiv beschäftigt, sich manchmal "vorkam wie der Rufer in der Wüste". Drei Sätze aus einem am 13. Oktober 1992 im Kölner Stadtanzeiger veröffentlichten Interview mit Hartmut Weule aus dem Daimler-Konzern zeichnen das allgemeine Bild der Situation:

"Es besteht die berechtigte Sorge, daß unsere Grundlagenforschung zum Nulltarif nach Japan geht, und unsere Spitzeninstitute - wie schon in Amerika - für Japaner arbeiten. Wir Deutschen sind ideenreicher. Andererseits sind die Japaner im Umsetzen schneller"

Wichtig ist, daß insbesondere japanische Unternehmen mit einer Vielzahl von erteilten und noch mehr angemeldeten Patenten zumindest auf dem Sektor Fuzzy Logic wieder einmal die Nase vorn haben. Aber inzwischen scheint sich die Entwicklung in Europa und in Deutschland zu forcieren.

Bei der INTERKAMA, dem "innovationsmarkt messen und automatisieren" der im Dreijahresturnus im Düsseldorfer Messegelände stattfindet, wurde 1992 mit einem speziellen Bereich auf dem Messestand der Klöckner Moeller-Unternehmensgruppe erstmals die Thematik Fuzzy Logic auf der Messe dargestellt. Es ist zu hoffen, daß die Idee mit der Stiftung ELITE (siehe unten) Erfolg hat, die sich konzentriert um die Verbreitung des Fuzzy-Logic-Gedankengutes auch und vor allem in mittelständischen Unternehmen bemüht.

Entwicklungswerkzeuge für Fuzzy-Systeme

Man kann die Programme für die Fuzzy-Systeme in jeder beliebigen Programmiersprache erstellen. Sinnvoll ist die Verwendung von Hochsprachen zur Erleichterung der Arbeit. Inzwischen gibt es auch schon vollständige Entwicklungsumgebungen, die die Arbeit erleichtern. Wir wollen hier nur auf eines dieser Werkzeuge eingehen.

Weitere Systeme stellt Tilli in "Automatisierung..." (siehe unten) kurz vor. An der RWTH Aachen und von der INFORM wurde ein interaktives Entwicklungssystem für Fuzzy-Prozesse neu- und fortentwickelt, das unter dem Namen fuzzyTECH vertrieben wird.

Mit diesem Entwicklungswerkzeug wurde auch die Software für das selbststabilisierende Auto (siehe oben) erstellt. Dabei ist man, und das ist exemplarisch für die Einsatzmöglichkeiten von fuzzyTECH, wie folgt vorgegangen:

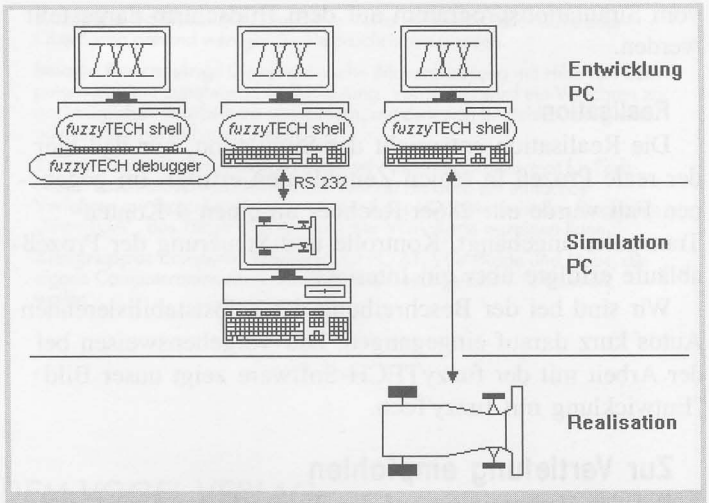
Modellentwicklung

Ein theoretisches Modell wurde direkt auf dem PC unter der fuzzyTECH-Shell entwickelt. Mit dem integrierten fuzzyTECH-Debugger wurden die Inferenz-Vorgänge grafisch auf dem Bildschirm dargestellt. Dabei wurden alle im Modell verwendeten Symbole wie linguistische Variablen und Regeln auf dem Bildschirm dargestellt.

Simulation

Hierzu werden zwei Rechner benötigt. Diese sind durch die serielle Schnittstelle miteinander verbunden. Auf dem Master ist die fuzzyTECH-Software installiert. Auf dem anderen Rechner befindet sich die Simulations-Software.

Zwischen beiden Systemen wird eine Verbindung dadurch hergestellt, daß vom Master aus der Fuzzy-Kontrol-



Entwicklung mit fuzzyTech

Der Begründer der Fuzzy-Theorie, Lofti A. Zadeh

ler auf dem anderen Rechner installiert wird. Der Kontroller kann vom Master aus modifiziert werden. Der Kontroller ermöglicht sodann die Überprüfung von Abläufen, die vom Simulationsprogramm auf dem Bildschirm dargestellt werden.

Realisation

Die Realisation entspricht der Simulation, nur daß hier der reale Prozeß in realen Zeitabläufen erfolgt. Im gegebenen Fall wurde ein 286er-Rechner an einen 4-Konten-Transputer angehängt. Kontrolle und Steuerung der Prozeßabläufe erfolgte über ein Interface.

Wir sind bei der Beschreibung des selbststabilisierenden Autos kurz darauf eingegangen. Die Vorgehensweisen bei der Arbeit mit der fuzzyTECH-Software zeigt unser Bild "Entwicklung mit fuzzyTech.

Zur Vertiefung empfohlen

Bei der Vorbereitung auf das Thema Fuzzy Logic sind wir auf zwei Bücher gestoßen, die dem Leser empfohlen werden können, der an einer Vertiefung der Erkenntnisse interessiert ist: Die beiden Bücher sind:

Thomas Tilli: "Fuzzy-Logic" Grundlagen, Anwendungen, Hard- und Software

In diesem Buch vermittelt Tilli die theoretischen Grundlagen der Fuzzy-Set-Theorie. Sehr ausführlich beschrieben wird auch die Implementierung aller dargestellten Operationen in ein
Turbo-Pascal-Projekt.
Buch mit Diskette (Quelltexte der wichtigstenProgrammbeispiele),
2. Auflage, Franzis Verlag, München 1992

Thomas Tilli: "Automatisierung mit Fuzzy-Logic" Fuzzy-Hardware und Entwicklungstools im praktischen Einsatz. Hier wird vertieft auf die im ersten Buch nur angerissene Thematik der praktischen Nutzung der Fuzzy-Set-Theorie eingegangen.
Buch mit Diskette (Demoprogramm einer Fuzzy-Shell und Quelltexte),
Franzis Verlag, München 1992

Beide Bücher richten sich vor allem an den mathematisch schon etwas versierteren Leser, der am Programmieren in Turbo Pascal Freude hat. Der Ingenieur Thomas Tilli verdient Dank dafür, daß er als einer der ersten sich bemüht hat, die nicht leicht darzustellende Thematik auch für den interessierten Laien aufzubereiten. Ein Dank des Autors gilt auch dem Institut für Operations Research und Management GmbH (INFORM) in Aachen. Bei einem ersten Treffen auf der INTERKAMA 1992 in Düsseldorf hatte der Leiter des Geschäftsbereichs Fuzzy Technologien, Dipl.Ing Constantin von Altrock spontan seine informative Unterstützung bei der Bearbeitung des Themas zugesagt und danach auch eingehalten. Von INFORM wurde die Fuzzy-Entwicklungsoberfläche fuzzyTech entwickelt, die mit der Versionsnummer 3.0 auf dem Stand der Firma Klöckner-Moeller gezeigt wurde. Dieses System läuft unter Windows und ermöglicht die hardwareunabhängige Entwicklung von Fuzzy-Lösungen. Mit Hilfe der fuzzyTech-Codegeneratoren kann fast jede Hardware optimiert unterstützt werden. Das Unternehmen INFORM wurde vor 22 Jahren von Professor Dr. Zimmermann von der RWTH Aachen gegründet und beschäftigt sich seit mehr als 12 Jahren mit dem Einsatz von Fuzzy Logic. Professor Dr. Zimmermann ist seit vielen Jahren einer der anerkanntesten Forscher auf dem Gebiet Fuzzy
Logic. Er gehört auch zum wissenschaftlichen Beirat der europäischen, gemeinützigen Stiftung ELITE (European Laboratory for intelligent Techniques Engineering), die sich zum Ziel gesetzt hat, die Fuzzy-Logic-Thematik insbesondere mittelständischen Unternehmen nahezubringen. Leser, die sich über die Stiftung informieren wollen, sollten sich an die Firma Klöckner-Moeller in Bonn wenden, die schon bei der Gründung der Stiftung mitgewirkt hat.

Zum Schluß

In diesem Abschnitt haben wir den Versuch unternommen, die Grundlagen der Fuzzy Logic vereinfacht darzustellen. Formeln ließen sich von der Materie her leider nicht vermeiden. Wir hoffen dennoch, daß unsere Leser ohne Vorkenntnisse sich in die Thematik eindenken konnten. Für uns ist die Beschäftigung mit dem Thema zumindest Anlaß, uns auch künftig mit dieser wahrlich faszinierenden Logic weiterzubeschäftigen. Wenn auch Sie daran interessiert sind, tiefer in diese Materie einzusteigen, sollten Sie uns Ihr Interesse mitteilen.

Letzte Informationen

Die schon erwähnte INFORM in Aachen führt Workshops und Seminare, zum Beispiel über die Themen Fuzzy-Entwurfsmethodik und Industrielle Anwendung der Fuzzy Logic Control, durch.

Bei Interesse sollten Sie die nächsten Termine direkt bei dem Institut erfragen. Von besonderer Bedeutung für die gesamte Thematik wird das 3. Aachener Fuzzy-Symposium sein, das in Frankfurt stattfinden wird.

Nachdem bereits während des 1. Aachener Fuzzy-Symposiums im Juli 1991 die Erfinder und Väter der Fuzzy-Technologien die Grundlagen der Fuzzy Logic dargestellt haben, stand das 2. Symposium im März 1992 bereits im Zeichen der praktischen Anwendungen.

Im 3. Aachener Fuzzy-Symposium, das wie die vorhergehenden von INFORM ausgerichtet wird, werden nun erstmals nur die Anwender das Wort haben. Die Veranstaltung findet am 10. Mai 1993 im Airport-Conference-Center Frankfurt statt.

Zwölf Referenten aus deutschen Top-Unternehmen berichten anhand ihrer Fuzzy-Applikationen über ihre Erfahrungen von der Entwicklung über die Implementation bis zur Vermarktung von Fuzzy-Anwendungen. Breiten Raum wird die Diskussion mit den Referenten einnehmen.

Wenn Sie Grundkenntnisse über Fuzzy-Logic haben, sollten Sie sich bei INFORM baldmöglichst anmelden. Hier aus diesem Grund nochmals die vollständige Anschrift:

INFORM GmbH
Geschäftsbereich Fuzzy Technologien
Pascalstr. 23
5100 Aachen

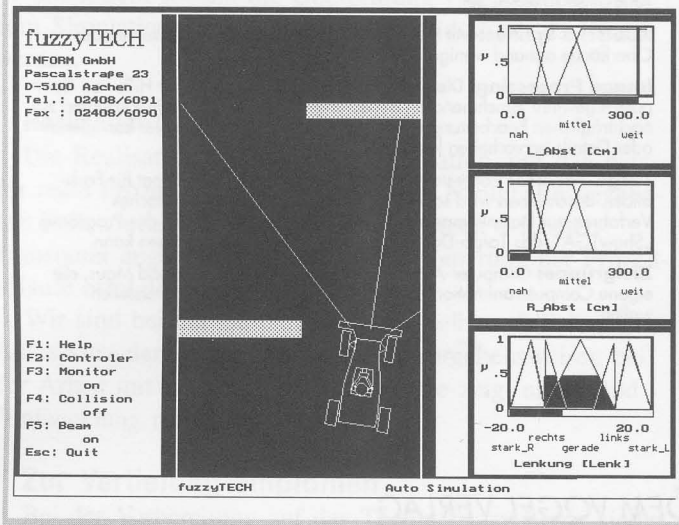
Tel.: 02408-9456-0
Fax: 02408-6090

Die Demoversion von fuzzyTECH

Auf der Diskette finden Sie komprimiert Programm und Dateien einer fuzzyTECH Simulation. Es geht dabei um eine etwas reduzierte Version der Originalprogramme. Bitte beachten Sie, daß die in der Demo verwendete fuzzyTECH-Version eine ältere ist. Informationen über den neuesten Stand dieses Entwicklungssystems sollten Sie sich bei der INFORM, Aachen, einholen.

Als Lehr- und Forschungsobjekt einer komplexen, dynamischen Aufgabenstellung der Regelungstechnik wurde am Lehrstuhl für Operations Research der Rheinisch-Westfälischen Technischen Hochschule (RWTH) Aachen das Modell eines Hochleistungsfahrzeuges aufgebaut. Das Originalfahrzeug verfügt über einen Elektro-Mittelmotor mit 770 Watt (1 PS) Leistung und Scheibenbremsung. Alle Räder sind einzeln aufgehängt und mit Federbeinen und Stoßdämpfern ausgerüstet. Der Antrieb erfolgt über ein sperrbares Differential. Insgesamt ist das Auto etwa 55 Zentimeter lang und wiegt etwa 3,5 Kilogramm.

Das Fahrzeug erreicht Spitzengeschwindigkeiten bis zu 80 Kilometer pro Stunde, die Schleuder- und Rutschexperimente werden bei Geschwindigkeiten zwischen 30 und 50 Kilometern pro Stunde durchgeführt. Die Steuerung eines solchen Fahrzeugs im Grenzbereich in Echtzeit stellt für die herkömmliche Regelungstechnik eine schwierige Aufgabe dar. Ziel der Forschungsarbeit war es, die Leistungsfähigkeit der Fuzzy Logic an dieser Problemstellung darzustellen.



Der fuzzyTECH-Bildschirm des Demogramms

Um eine Echtzeitsteuerung bei diesen Geschwindigkeiten zuverlässig durchführen zu können, ist eine Taktzeit des Reglers von zehn Millisekunden notwendig. Da in dieser kurzen Zeit keine echte Bilddatenauswertung möglich ist, wurde eine sehr primitive, aber schnelle Sensorik verwendet. Die gesamte Sensorik besteht aus drei Ultraschallelementen, die jeweils nach vorne, rechts und links gerichtet und fixiert sind, sowie Infrarotsensoren in allen Rädern zur Messung der Radgeschwindigkeiten.

Da die Ultraschallsensoren nur die relative Entfernung des Autos zu Hindernissen (und deren erste und zweite Ableitung) in einem festen Winkel zur Fahrtrichtung erfassen, stellt sich die Aufgabe, aus dieser geringen Information den aktuellen "Zustand" des Fahrzeugs in seiner Umgebung herzuleiten. Genau dieses "herleiten" erfordert ein "ins-Verhältnis-setzen" der sensorischen Daten, was mit Hilfe der Fuzzy Logic erfolgt. Aus dieser umgangssprachlichen Beschreibung des aktuellen Zustands (Beispiel: "Geschwindigkeit ziemlich hoch, Auto schiebt in enger Linkskurve stark über die Vorderräder") erfolgt die Ableitung der erforderlichen Reaktion, um das Auto wieder zu stabilisieren (Beispiel: "Motor stark drosseln, Lenkeinschlag beibehalten") wiederum mit Hilfe der Fuzzy Logic.

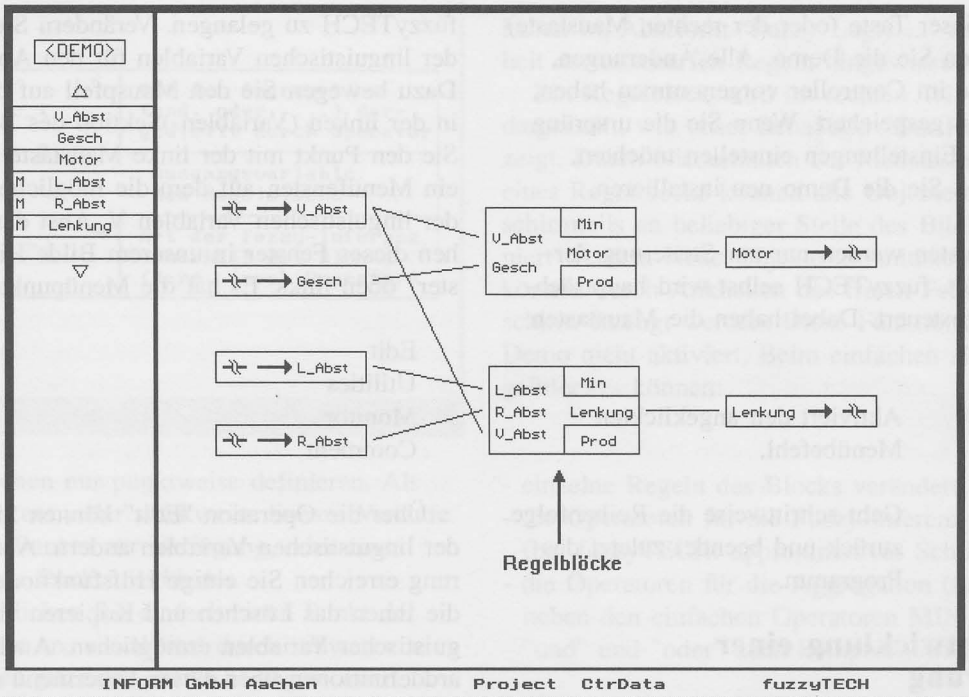
Simulation des Autos

Zur on-line-Entwicklung von Fuzzy-Systemen benötigt man entweder die Verbindung des Entwicklungssystems mit dem zu regelnden Prozeß selbst oder eine Prozeßsimulation. Um Ihnen einmal ein "Gefühl" dafür zu geben, wie eine Fuzzy-Steuerung mit fuzzyTECH entwickelt wird, ist in unserem Beispielprogramm die Simulation des Fuzzy-Autos mit dem on-line-Modul von fuzzyTECH gekoppelt. Eine erste, einfache Regelstrategie, so wie man sie in wenigen Minuten selbst formulieren könnte, ist für einen schnellen Start bereits eingestellt.

Da diese Simulation nur als Demonstration konzipiert ist, wurden die linguistischen Variablen, die Attribute und die Regelblöcke in ihrer Struktur festgelegt. Die Fuzzy-Steuerung des realen Fuzzy-Autos ist wesentlich umfangreicher und berücksichtigt auch noch weitere Sensordaten. Sie haben dennoch in dieser Demo die Möglichkeit, die Formen der Zugehörigkeitsfunktionen der Attribute und die Einsichtigkeitsgrade der Regeln zu verändern.

Der Bildschirm wird von der Simulation und von fuzzyTECH benötigt. Daher muß zwischen beiden hin- und hergeschaltet werden. Aufgrund des on-line-Konzeptes ist nach einer Veränderung der Regelstrategie mit fuzzyTECH kein Recompilieren notwendig - das Auto könnte also auch während der Entwicklung weiterfahren.

Das Regelziel ist ganz einfach: "Fahr so schnell es geht, ohne an den zufällig autauchenden Hindernissen anzustoßen". Da das Auto weder seine eigene Position auf dem Weg "kennt", noch die Richtung, in die es fährt, muß



Der fuzzyTECH-Bildschirm

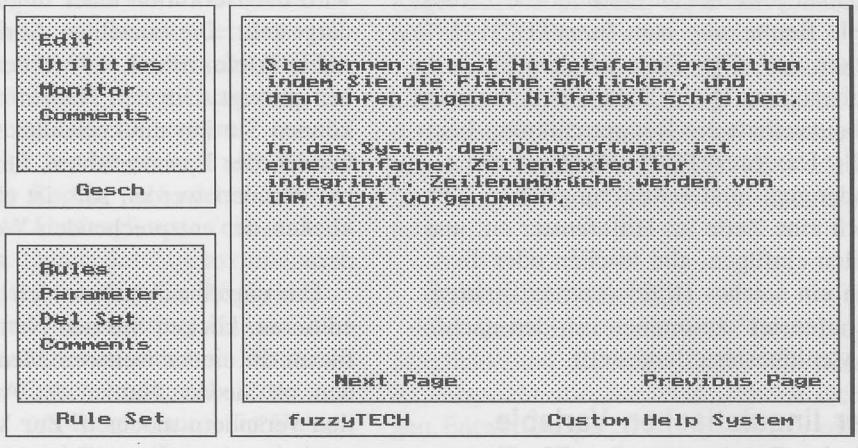
die Entscheidung über Gas und Bremse sowie über die neue Lenkstellung nur aufgrund der Abstandsinformationen und der Geschwindigkeit des Autos erfolgen. Da in dieser Demo-Simulation weder Rutschen noch Schleudern simuliert wird, existiert statt der Geschwindigkeitsmessung aller vier Räder nur eine Gesamtgeschwindigkeitsmessung.

Mit den Funktionstasten können Sie auch im Demo-Programm einige Eigenschaften der Simulation aus- und einschalten:

<F1> Aufruf des fuzzyTECH Hilfe-Systems.
<F2> Umschalten des Bildschirms zur fuzzyTECH-Demoversion.
<F3> Ausschalten des Monitors.

Monitore (Prozeßmonitore) visualisieren linguistische Variablen in Echtzeit. Sie sind Bestandteil des fuzzyTECH-On-Line-Werkzeugs, werden aber im Bild der Simulation angezeigt. Da in der Demo die Monitore synchron mit dem Prozeß gekoppelt sind, verlangsamt ihre Ausgabe auf dem Bildschirm die Fahrdarstellung des Autos. Sie können sowohl einzeln mit fuzzyTECH ein- und ausgeblendet werden, als auch alle zusammen direkt aus der Simulation heraus mit dieser Funktionstaste.

<F4> Optionale Kollisionserkennung. Überprüft den Fahrstil Ihrer Wissensbasis.
<F5> Anzeige der Abstandsmessung. Hier sehen Sie, wie sich das Auto an seiner Umgebung orientiert.



Hilfe- und Menüfenster

<ESC> Mit dieser Taste (oder der rechten Maustaste) beenden Sie die Demo. Alle Änderungen, die Sie im Controller vorgenommen haben, bleiben gespeichert. Wenn Sie die ursprünglichen Einstellungen einstellen möchten, müssen Sie die Demo neu installieren.

Diese Funktionstasten werden nur zur Steuerung der Simulation verwendet. fuzzyTECH selbst wird hauptsächlich mit der Maus gesteuert. Dabei haben die Maustasten folgende Funktionen:

Linke Maustaste: Aktiviert den angeklickten Menübefehl.

Rechte Maustaste: Geht schrittweise die Reihenfolge zurück und beendet zuletzt das Programm.

"On-line"-Entwicklung einer Fuzzy-Steuerung

Durch Drücken der <F2>-Taste schalten Sie den Bildschirm auf fuzzyTECH um. Das Hauptfenster ist in zwei Bereiche unterteilt: Links befindet sich die Liste aller Variablen im System, rechts wird die Struktur des Fuzzy-Systems angezeigt und bearbeitet.

Damit dieses Demoprogramm mit praktischem Beispiel einer Entwicklung von Fuzzy-Regelungen nicht eine komplette Einarbeitung in fuzzyTECH erfordert und auf eine Diskette paßt, ist nur der Teil der Funktionen aktiviert, die Sie unbedingt benötigen, um leichte Änderungen am System selbst vornehmen zu können.

Die grundsätzliche Bedienung von fuzzyTECH ist schnell erklärt:

Durch einmaliges Anklicken mit der linken Maustaste werden die auf dem Bildschirm dargestellten Objekte (linguistische Variablen, Schnittstellen, Regelblöcke ...) aktiviert. Bei den meisten von fuzzyTECH dargestellten Objekten wird beim Aktivieren ein Menüfenster angezeigt, auf dem die möglichen Operationen für dieses Objekt angezeigt werden. Durch Drücken der rechten Maustaste oder <ESC> gelangt man jeweils eine Menüebene zurück.

Noch eine generelle Bemerkung zum fuzzyTECH-Hilfe-System. Durch Drücken der <F1>-Taste erhalten Sie in jeder Situation kontextbezogene Hilfe. Daher wird Ihnen im folgenden Text nur beispielhaft die Benutzerführung beim Entwickeln vorgestellt. Das fuzzyTECH-Hilfesystem ist von Ihnen frei veränderungs- und erweiterungsfähig. Klikken Sie hierzu einfach eine Zeile im Hilfefenster an, und Sie können neue Zeilen einfügen, alte löschen oder Text verändern. Wir haben ein solches Hilfefenster in unserem Bild "Hilfe- und Menüfenster" zusammen mit den nachfolgend beschriebenen Menüfenstern dargestellt.

Verändern einer linguistischen Variable

Machen Sie die Probe: Drücken Sie die <F2>-Taste von der Auto-Simulation aus, um in die Demoversion von fuzzyTECH zu gelangen. Verändern Sie nun die Definition der linguistischen Variablen für den Abstand nach vorne. Dazu bewegen Sie den Mauspfeil auf den Kasten V_Abst in der linken (Variablen-)Sektion des Bildschirms. Klicken Sie den Punkt mit der linke Maustaste an, dann erscheint ein Menüfenster, auf dem die möglichen Operationen mit der linguistischen Variablen V_Abst dargestellt sind. Sie sehen dieses Fenster in unserem Bild "Hilfe- und Menüfenster" oben links. Es hat die Menüpunkte:

Edit
Utilities
Monitor
Comment

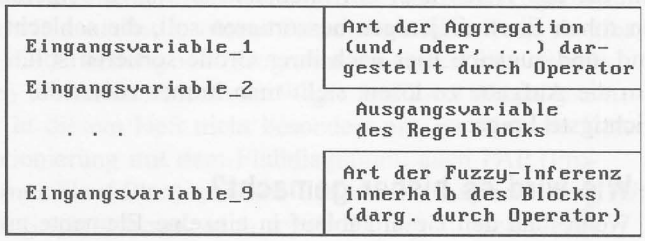
Über die Operation "Edit" können Sie die Definitionen der linguistischen Variablen ändern. Als Arbeitserleichterung erreichen Sie einige Hilfsfunktionen über "Utilities", die Ihnen das Löschen und Kopieren von Definitionen linguistischer Variablen ermöglichen. Auch sind Standarddefinitionen über dieses Untermenü aufzurufen, und es können die Namen der Variablen verändert werden. Über "Monitor" können Variablen für den Echtzeit-Prozeßmonitor ausgewählt werden. Diese Funktion ist in der Demoversion aktiv. Durch Anklicken mit der linken Maustaste gelangen Sie in ein Menü, in dem Sie die anzuzeigende Position in der Autosimulation auswählen können. Dieses Menü weist dann folgende Punkte auf:

Top
Center
Bottom
Off

Über "Comment" gelangen Sie in einen Texteditor, mit dem Sie alle Objekte Ihres Fuzzy-Systems beschreiben können. In der Compiler-(Voll-)Version können diese Kommentare (als Option) mit in den erzeugten Code hineingebunden werden. Die Bedienung des Kommentareditors entspricht der des fuzzyTECH-Hilfesystems.

Wenn Sie jetzt einmal die "Edit"-Funktion anklicken, wird die Definition einer linguistischen Variable grafisch dargestellt. Die einzelnen Attribute einer linguistischen Variable (klein, mittel, ...) werden in unterschiedlichen Farben gezeigt. Um den Namen der Basisvariablen zu ändern, können Sie ihn direkt anklicken und verändern. Der Wertebereich der Basisvariablen, für den die linguistische Variable definiert werden soll, ist ebenfalls durch direktes Anklicken der entsprechenden Werte in der Grafik zu verändern.

Um einzelne Attribute der linguistischen Variablen zu editieren, klicken Sie den Namen der Variablen in der Grafik an. In einem weiteren Menü können Sie dann entscheiden, ob Sie den Namen des Attributes oder dessen Definition verändern möchten. Zur Veränderung der Definition erscheint ein weiteres Fenster, in dem nur das selektierte Attribut dargestellt ist. In der Demoversion können Sie



Der Regelblock

Zugehörigkeitsfunktionen nur punktweise definieren. Als Kurvenformen stehen entweder stückweise lineare Verläufe durch die definierten Punkte zur Verfügung, oder eine S-Kurve, die definierte Punkte verbindet.

Zur Veränderung klicken Sie einfach einen Punkt der Zugehörigkeitsfunktion an, visualisiert durch schwarze Rechtecke. Sie können jetzt den Punkt bewegen, allerdings nur soweit, daß die Funktion eindeutig bleibt. Durch erneutes Klicken wird die Maus wieder freigegeben, und der Punkt ist neu definiert. Auch Löschen und Einfügen von Punkten ist möglich: Zum Löschen klicken Sie den Punkt mit der linken Maustaste an, und drücken die <Del>/<Entf>-Taste. Um einen neuen Punkt einzufügen, klicken Sie einfach die Position, an der der Punkt erscheinen soll, mit der Maus an.

Veränderung von Regelmengen

Im Zusammenhang mit der Diskussion um die Programmierung von wissensverarbeitenden Systemen mit Expertensystem-Shells wurde argumentiert, eine Menge von "wenn-dann"-Regeln sei leichter zu übersehen als ein prozedurales Programm. Allerdings verliert sich die Anschaulichkeit von Regelmengen ab einer gewissen Anzahl von Regeln recht schnell. Daher sind in fuzzyTECH Regeln in Blöcken strukturiert und die Gesamtheit der Regeln innerhalb eines Blocks kann zur besseren Übersicht grafisch dargestellt werden.

Auch die Beschränkung anderer Fuzzy-Werkzeuge darauf, eine Regel entweder nur voll zuzulassen oder gar nicht (Sie wird entweder hingeschrieben oder nicht), ist in fuzzyTECH aufgehoben. Es wäre eine Invertierung des Fuzzy-Gedankens, Zugehörigkeiten von Werten zu bestimmten Klassen von "völlig" bis "überhaupt nicht" zuzulassen, und dann für einzelne Regeln zu fordern, daß sie entweder "immer und in vollem Umfang" oder "nie und in keinster Weise" gelten müssen. Durch das "approximative (näherungsweise) Schließen" hat man nun die Möglichkeit, auch Regeln mit einer Gültigkeit "zu einem gewissen Grade" zuzulassen. Analog zu unserem menschlichen Denken wird dann beim Ableiten einer Entscheidung neben der Gültigkeit der Vorbedingungen von Regeln auch die Gültigkeit der Regel selbst in dem betrachteten Zusammenhang durch die Fuzzy-Inferenz "in ein Verhältnis" gesetzt. Wir haben im Abschnitt "Fuzzy Logic ..." auf diese Besonderheit der unscharfen Regeln hingewiesen.

Ein Regelblock wird im rechten Teil des Bildschirms dargestellt, wie unser Schaubild "Der Regelblock" es zeigt. Durch Niederhalten der Maustaste beim Anklicken eines Regelblocks können alle Objekte des rechten Bildschirmteils an beliebiger Stelle des Bildschirms positioniert werden. Neue Objekte (Schnittstellen, Module ...) können durch Anklicken des freien Feldes aus dem Bildschirm erzeugt werden. Diese Funktionen sind nicht in der Demo nicht aktiviert. Beim einfachen Anklicken eines Regelblockes können:

- einzelne Regeln des Blocks verändert werden,
- die Operatoren für die Fuzzy-Inferenz (bei fuzzyTECH: approximatives Schließen) und
- die Operatoren für die Aggregation (fuzzyTECH bietet neben den einfachen Operatoren MIN und MAX für "und" und "oder" auch kompensatorischen Operatoren).

Auch lassen sich über dieses Menü Regelblöcke (Rule Sets) löschen und kommentieren. Als Operatoren stehen in der Demo "min-max", "min-av" und "gamma" zur Verfügung. Alle diese Operatoren sind parametrisiert, das heißt, stufenlos über einen Rollbalken einstellbar.

Diesen Rollbalken können Sie mit der Maus einstellen, durch Bewegen der Maus von rechts nach links können Sie auch die Genauigkeit Ihrer Einstellung variieren. Minimum- und Maximum-Operatoren sind Spezialfälle (Parameter 0 oder 1) des "min-max"-Operators.

Über die Option "Plot" kann das Übertragungsverhalten des jeweiligen Operators auch grafisch dargestellt werden. Um die einzelnen Regeln zu definieren, klicken Sie einen Regelblock einmal an und wählen die Funktion "Rules" (Regeln). Falls in dem betrachteten Regelblock nur zwei Eingangsvariablen miteinander verknüpft sind, werden die einzelnen, sie verknüpfenden Regeln in einer Matrix dargestellt. Ist das entsprechende Feld in der Matrix weiß, so gilt die Regel völlig, ist es schwarz, so gilt die Regel überhaupt nicht. Über den Rollbalken können Sie hier auch Zwischenwerte einstellen.

Sind mehr als zwei Eingangsvariable zu verknüpfen, so kann die Netzflußdarstellung übersichtlicher als die Matrixdarstellung sein. Auch können einzelne Regeln zur Darstellung im Prozeßmonitor ausgewählt werden. Der Ablauf einer Fuzzy-Inferenz läßt sich so lückenlos in Echtzeit verfolgen.

Schlußbemerkung

Das Ihnen hier vorliegende Demoprogramm bietet viele Möglichkeiten der individuellen Beeinflussung durch Sie. Es entspricht aber aus Platzgründen nicht dem vollständigen Entwicklungssystem. Außerdem muß hier nochmals der Hinweis erfolgen, daß es auf einer älteren fuzyTECH-Version basiert.

Ein Roboter wird konstruiert

Wir haben die Prinzipien der Steuerung und Regelung dargelegt. Wir wissen, daß mit Sensoren Informationen beschafft, diese Informationen in SPS oder/und Zentralrechnern verarbeitet werden und die Ergebnisse als Steuersignale an Aktoren gehen. Wie diese Elemente zusammenwirken, das wollen wir in diesem Beitrag behandeln.

Zunächst auch hier wieder eine kleine Begriffsbestimmung: Als Sensoren bezeichnen wir Elemente in einem Steuerungs- oder Regelungsprozeß, die Meßdaten liefern, als Aktoren Elemente, die Aktionen der Steuerung oder Regelung bewirken und ausführen. Ein Beispiel soll die Zusammenhänge kurz erläutern.

Es ist ein Beispiel aus den Anfangszeiten des technischen Zeitalters. Schon schnell nach der Erfindung der (Zweizylinder-)Dampfmaschine stellte James Watt fest, daß es problematisch ist, die Geschwindigkeiten des Antriebs durch mehr oder weniger Kohle regeln zu wollen. Gegen Überdruck erfand man bald ein entsprechendes Ventil, das durch ein Gewicht schloß.

Zur Regelung der abgegebenen Leistung entstand der Gedanke, den Dampfdruck im Kessel an die Geschwindigkeit der Treibräder zu koppeln. Steigt die Geschwindigkeit, so muß der Dampfdruck reduziert, fällt die Geschwindigkeit, muß er erhöht werden.

Es entstand das Prinzip des Fliehkraftreglers. Bei ihm drehen sich zwei Pendelarme mit Gewichten mit einer durch die Dampfmaschine angetriebenen Welle. Je höher die Umdrehungsgeschwindigkeit ist, umso wirksamer greift die Fliehkraft auf die Pendelarme zu und drückt diese nach außen.

Eine am unteren Teil des Systems angebrachte Muffe wird entsprechend nach oben gezogen und öffnet eine Drosselklappe (ein Ventil), über die der Dampf entweichen kann. Läßt der Dampfdruck nach, so reduziert sich die Geschwindigkeit und die Drosselklappe schließt sich.Hier ist das Pendelarmsystem der (analog arbeitende) Meßfühler, die Drosselklappe der Aktor (siehe Bild: Fliehkraftregler - Sensor und Aktor). Aktoren sind Schalter und Ventile, die die eigentlichen Aktionen ausführen. Wir wollen am Beispiel einer Maschine zum Aufnehmen von rohen Eiern die Zusammenarbeit von Sensoren, Reglern und Aktoren beschreiben.

Stellen Sie sich vor, Sie sollen eine Maschine konstruieren, die aus vielen über ein Fließband an ihr vorbeigeführten rohen Eiern diejenigen aussortieren soll, die schlecht sind, und zugleich Eier nach ihrer Größe sortieren soll. Um die Aufgabe zu lösen, stellt man immer zuerst die wichtigste Frage:

Wie wird es bisher gemacht?

Wenn man den Gesamtablauf in einzelne Elemente auflöst, kommt man etwa zu folgendem Ergebnis: Die Eier wandern auf einem Fließband zum Aussortieren. Dort werden sie durchleuchtet. Schlechte Eier lassen das Licht schlechter durch wie gute Eier.

Dies ist das Kriterium. Die schlechten Eier werden mit der Hand aufgenommen und in den Abfalleimer geworfen. Die guten werden an Schablonen auf ihre Größe geprüft und entsprechend sortiert. Wenn man weiß, wie der bisherige Ablauf war, kommt der nächste Schritt.

Konzeptentwicklung

Aus der Kenntnis der bisherigen Abläufe und der verfügbaren Bauelemente entsteht das Konzept der Problemlösung. In unserem Fall muß zunächst die wichtigste Frage geklärt werden, ob es klar definierbare Unterschiede der Lichtdurchlässigkeit zwischen faulen und frischen Eiern gibt. Es könnte auch sein, daß faule Eier über die Schale das Fäulnisgas Schwefelwasserstoff abgeben.

Gibt es eine Möglichkeit, mehrere Eier zugleich zu prüfen? Gehen wir einmal davon aus, diese Fragen seien geklärt, dann muß der Entwickler recherchieren, ob es Sensoren mit der erforderlichen Genauigkeit gibt. Auch das sei gegeben.

Dann folgt als nächster Schritt die Klärung der Frage, wie die faulen Eier vom Fließband entfernt werden. Nehmen wir an, der Entwickler hat die Idee, mit Saugnäpfen die Eier vom Band aufzunehmen und über dem Abfalleimer wieder fallenzulassen. Wir haben in unserem Bild "Das Konzept" das Rohkonzept für unsere Problemlösung skizziert. Daraus sieht man, daß Sensoren und die Aktoren (Saugnäpfe) in Querrichtung zum Band und in vertikaler Richtung beweglich sein müssen.

Dies bringt uns zu der wichtigen Frage der Freiheitsgrade von Robotern, auf die wir im folgenden Unterabschnitt eingehen werden. Anhand einer Modellskizze, wie sie in unserem Bild dargestellt ist (deren Details man zunächst noch gar nicht zu sehr ausführen muß) ist der folgende Schritt wesentlich vereinfacht.

Im Konzept zeigt sich, daß folgende Motoren für Aktionen (dargestellt durch Richtungspfeile) erforderlich werden: Ein Motor für die Bandbewegung, der zweite für die links-/rechts-Bewegung des S und ein dritter für die Auf-/Ab-Bewegung des S. Als Sensoren haben wir neun Lichtsensoren, als Aktoren neun Saugnäpfe, deren Saugwirkung durch einen weiteren Motor verbunden mit einer kleinen Vakuumpumpe erzeugt wird. Vier Motoren und neun Saugnäpfe, das sind die Aktoren, neun Lichtsensoren die Sensoren unseres Konzepts.

Programmentwurf

Für die Abläufe entwerfen wir ein kleines Programm, das wir zunächst in einem Pseudocode schreiben. Es gibt für die Erstellung von Programmentwürfen im wesentlichen drei unterschiedliche Wege. Auf zwei davon gehen wir in diesem Heft nicht besonders ein, nämlich die Konzeptionierung mit dem Flußdiagramm, auch PAP (Programmablauf-Plan) genannt, und die Konzeptionierung mit dem Struktogramm, nach seinen Erfindern auch Nassi-Shneiderman-Diagramm bezeichnet. Beide verwenden Symbole für die Darstellung der Abläufe in einem Programm. Wir verwenden hier die dritte Vorgehensweise, die Programmierung in Pseudocode, den man dann anschließend in den entsprechenden Programmcode umsetzen kann. Im Abschnitt "Wie man Computer programmiert" sind wir darauf näher eingegangen.

Damit wir keinen Fehler machen, gehen wir beim Programmentwurf Schritt für Schritt vor. Wir entwerfen zunächst den Programmrahmen, der beispielsweise die allgemeine Steuerung des Transportbandes enthält. Danach formulieren wir die einzelnen Prozeduren, die immer dann angesprungen werden, wenn ein Sensor, ein Schalter einen bestimmten Wert liefert. Wir fangen also mit der obersten Ebene des Programms an und gehen dann immer mehr ins Detail. Diese Vorgehensweise wird auch Top-Down-Methode (deutsch: Von Oben nach Unten) bezeichnet. Als erstes wollen wir das Hauptprogramm erstellen. Es handelt sich dabei im wesentlichen um eine Schleife, die solange ausgeführt wird, bis der Hauptschalter alles abschaltet. Davor aber wollen wir zwei Konstanten festlegen, die die Werte Wahr (-1) und Falsch (0) erhalten.

```
KONSTANTE Wahr = -1
KONSTANTE Falsch =  0
Bandlaufschalter_Schalten Übergebe Hauptschalter
WIEDERHOLE_SOLANGE Hauptschalter = Wahr
        Band_eine_Einheit_vorschieben
        Prüfen_Und_Aussortieren
        Bandlaufschalter_Schalten (Hauptschalter)
ENDE_WIEDERHOLE
```

Die Prozedur für den Vorschub des Bandes ist einfach. In ihr wird nur die Anweisung formuliert, die dem entsprechenden Motor für eine bestimmte Umdrehungszahl in einer bestimmten Richtung Strom zuführt. Die Übergabe Bandmotor_Drehen_Links erfolgt über die entsprechende Schnittstelle.

Die Details der Schnittstelle setzen wir als gelöst voraus. Die Argumente bedeuten: 1 ist der Motor 1, Links ist die Laufrichtung und 5 ist die Anzahl der Umdrehungen.

```
PROZEDUR Band_Eine_Einheit_Vorschieben
        Motor_Drehen (1, Links, 5)
ENDE_PROZEDUR
```

Auch die Prozedur Bandlaufschalter_Schalten gibt keine großen Probleme. Der Bandlauf soll über den Hauptschalter Ein oder Aus geschaltet werden. Bei dieser Prozedur wird aber im Gegensatz zur vorhergehenden ein Wert (Hauptschalter) übergeben. Diese Übergabe erfolgt in beiden Richtungen. Die Prozedur "erfährt" also aus dem Hauptprogramm, welche Position der Hauptschalter hat und gibt ihrerseits diesen Wert wieder an das Hauptprogramm zurück.

```
PROZEDUR Bandlaufschalter_Schalten Übergebe
                Hauptschalter
    WENN Hauptschalter = Falsch DANN
        Hauptschalter = Wahr
    SONST
        Hauptschalter = Falsch
    ENDE_WENN
ENDE_WIEDERHOLE
```

Die wichtigste und umfangreichste Prozedur des Programms ist Prüfen_Und_Aussortieren. Sie greift sinnvollerweise für die immer wiederkehrenden Vorgänge auf weitere Prozeduren zu. Gehen wir vor wie beim Hauptprogramm und entwerfen wir wieder zunächst den Rahmen.

```
PROZEDUR Prüfen_Und_Aussortieren
    FÜR EinNr = 1 BIS EinNr = 9
        WENN Prüfen (Ei-Nr) = Falsch DANN
            Aussortieren (EiNr)
        ENDE_WENN
    ENDE_FÜR
ENDE_PROZEDUR
```

In dieser Prozedur haben wir vier neue Prozeduren (Funktion Prüfen (Ei-Nr), Prozedur Aussortieren (Ei-Nr), Funktion AbfallEimer_Voll, Prozedur Ablauf_Eimer_Leeren) aufgerufen, die wir jetzt zu programmieren haben. Die erste dieser Prozeduren ist die FUNKTION Prüfen (Ei-Nr). Sie soll den Wert Wahr zurückgeben, wenn das Ei (Ei-Nr) in Ordnung und den Wert Falsch, wenn es nicht in Ordnung ist. Damit diese Funktion arbeiten kann, muß der Sensorwert für das betreffende Ei an das Programm übergeben und in der Funktion dann mit dem Grenzwert verglichen werden. Die Übergabe des Sensorwertes erfolgt über die Schnittstelle. Ist der Sensorwert größer als der Grenzwert, dann ist die Aussage Prüfen logisch = Wahr und der zurückgegebene Wert auch. Im anderen Fall ist der Rückgabewert logisch = Falsch.

```
FUNKTION Prüfen (Ei-Nr )
    Schlitten_Auf_Ab (Prüfposition)
    Prüfen = (Sensorwert  Ei-Nr > GrenzWert)
    Schlitten_Auf_Ab (Normalposition)
ENDE_FUNKTION
```

In der mittleren Zeile dieser Prozedur wird der Wert für Prüfen daraus ermittelt, ob die Bedingung (Sensorwert Ei-Nr GrenzWert) stimmt. Entsprechend ist das Ergebnis und damit der Wert von Prüfen Wahr oder Falsch. Die in der Funktion zusätzlich erforderliche Aktion der Schlittenpositionierung Schlitten_Auf_Ab gibt die Motornummer 2, die Drehrichtung und Anzahl der Umdrehungen an die Prozedur Bandmotor_Drehen weiter, die dann ihrerseits die Schnittstelle ansteuert.

Die Prozedur Aussortieren in Prüfen_Und_Aussortieren beinhaltet wieder mehrere Vorgänge, nämlich das Aufnehmen des schlechten Eis, das Fahren des Schlittens über den Abfallbehälter, dort das Fallenlassen des Eis und das Zurückfahren in die Ausgangsposition.

```
PROZEDUR Aussortieren (EiNr)
    Ei_Aufnehmen (EiNr, Wahr)
    Schlitten_Fahren (Abfalleimer)
    Ei_Aufnehmen (EiNr, Falsch)
    Schlitten_Fahren (Normalposition)
ENDE_PROZEDUR
```

Es ging uns hier darum, das Prinzip der Programmentwicklung darzustellen. Deshalb wollen wir auf die weitere Detaillierung des Programms verzichten. Wie Sie aber schon sehen konnten, führt so ein einfacher Vorgang wie das Aussortieren fauler Eier zu sehr komplexen Programmen, wenn es um die Umsetzung in ein Steuer- und Regelsystem geht. Vieleicht übernehmen Sie selbst die weitere Formulierung. Nach der Formulierung in Pseudocode kann dann die Umsetzung des Programms in die entsprechende Programmiersprache erfolgen. Wer sich sehr gut in der Programmiersprache auskennt, kann natürlich direkt in dieser Sprache programmieren.

Modellbau

Im Modell werden zum ersten Mal die Konzeptideen und die Wirkung des Programms überprüft. Durch Versuche mit dem Modell stellt man fest, ob sich alles realisieren läßt. Entsprechend erfolgen Korrekturen am Konzept und Modell. Nehmen wir an, das gleichmäßig laufende Fließband ergebe immer dann ein Problem, wenn der Abfalleimer geleert wird, weil die Anlage angehalten werden muß. Die erste Idee ist wohl, die Entsorgung kontinuierlich abzuwickeln, also Ablaufwannen zu schaffen, die ihren Inhalt an einen großen Sammelbehälter abgeben. Wir wollen in unserem Beispiel annehmen, daß es gegen diese Lösung nicht aufhebbare Hindernisse gibt.

Dann muß das Band in dem Augenblick der Entleerung des Abfallbehälters gestoppt werden. Dazu muß ein Meßfühler im Abfalleimer permanent die Füllhöhe oder einmal die maximale Füllhöhe ermitteln.

Ist die Füllhöhe erreicht, sind alle Aktionen von Band (= B) und Lichtsensor/Saugnapf-Einheit (= Schlitten S) einzustellen, der Abfalleimer muß entleert werden und B und S sind wieder zu starten. Aus dieser Forderung ergibt sich eine Programmergänzung, die direkt in das Programm eingefügt die Prozedur Prüfen_Und_Aussortieren wie folgt ändert:

```
PROZEDUR Prüfen_Und_Aussortieren
    FÜR EinNr = 1 BIS EinNr = 9
        WENN Prüfen (Ei-Nr) = Falsch DANN
            Aussortieren (EiNr)
        ENDE_WENN
    ENDE_FÜR

    WENN AbfallEimer_Voll = Wahr DANN
        Bandlaufschalter_Schalten (Bandschalter)
        Ablauf_Eimer_Leeren
        Bandlaufschalter_Schalten (Bandschalter)
    ENDE_WENN
ENDE_PROZEDUR
```

Hier könnte sich auch die Erkenntnis durchsetzen, daß der einzelne Transport der Eier zu zeitaufwendig ist. Eine weitere Programmergänzung müßte also dazu dienen, die Saugnäpfe, unter denen faule Eier liegen, gleichzeitig zu aktivieren und über dem Abfalleimer zu deaktivieren.

Technische Umsetzung

Dies ist die nach außen sichtbare Arbeit an dem Projekt. Hier geht es zunächst darum, die richtigen Sensoren und Aktoren zur Realisierung zu finden. Aus den Gerätewerten können sich für die Schnittstellengestaltung noch zusätzliche Änderungen ergeben. Die Anlage ist dann technisch zu erstellen und nach entsprechenden Test- und Justierungsläufen in Betrieb zu nehmen.

Verbesserung

Nachdem die Anlage installiert und in Betrieb genommen wurde, ergeben sich aus der Praxis heraus sehr häufig Änderungs- und Verbesserungsvorschläge. Diese dienen dazu, die konkrete Anlage oder Maschine den Praxisanforderungen anzupassen und Ideen und Anregungen für künftige Entwicklungen zu gewinnen.

Unser Beispiel hat gezeigt, wie komplex ein so einfach erscheinender Vorgang wie das Aussortieren von faulen Eiern ist.

Übrigens, haben Sie auch schon einmal gesehen, wie bei Kücken die weiblichen von den männlichen Tieren getrennt werden. Das Geschick der Arbeiterinnen und Arbeiter ist in absehbarer Zeit wohl nicht durch einen Roboter zu ersetzen.

Die Freiheiten eines Roboters

Mit dem Begriff Freiheiten meinen wir in diesem Zusammenhang die Bewegungsfreiheiten. In unserem Beispiel haben wir gesehen, daß das Band eine Bewegung in einer Richtung, also eindimensional, auszuführen hat. Die Bewegung des Schlittens S mit den Sensoren ist eine Vor_und_Zurück- sowie eine Auf_und_Ab-, eine zweidimensionale Bewegung.

Diese Bewegungen sind genau durch die zurückgelegten Wege zu definieren, also auch einfach in die Anzahl der Umdrehungen des Motors umzusetzen. Schwieriger wird dies bei Aktionen die dreidimensional erfolgen, bei denen also zu Vor_Zurück sowie Auf_Ab noch die Richtung Links_Rechts dazukommt.

Diesen dreidimensionalen Raum kann man auf unterschiedliche Weise abdecken. Wir haben in unserem Bild "Roboterfreiheiten" die verschiedenen Koordinatensysteme dargestellt. Das am einfachsten zu beherrschende System ist das kartesianische Koordinatensystem. Das System wurde erstmals in der von dem französischen Mathematiker René Descartes (daher der Name) 1637 veröffentlichten "Abhandlung über die Methode" beschrieben.

Bei diesem System stehen alle drei Raumdimensionen zueinander in rechten Winkeln, und die Berechnungen gehen von einem gemeinsamen Nullpunkt aus. Hier kann mit je einem X-, Y- und Z-Wert jeder Punkt im Raum präzise angesteuert werden.

Etwas komplizierter wird es, wenn wir neben unserem Band einen Greifroboter plazieren, der zwar selbst seine Position nicht verändern, den Greifer aber durch Drehbewegungen um seine Achse und Ein- und Ausfahren an beliebige Stellen des von ihm bearbeitbaren Raumes bewegen kann.

In diesem zylindrischen Koordinatensystem ergeben sich die Raumpositionen aus dem Z-Wert der Auf- und Abbewegung sowie einer Positionsbestimmung in der jeweiligen Ebene aus der Entfernung des Greifers vom Drehzentrum des Roboters (R = Reichweite) und dem Winkel zwischen Arm und x-Achse des Greifroboters.

Das dritte System ist das sphärische System. Hier sind sowohl der Winkel zu einer Basislinie in der X/Y-Ebene als auch der zur Z-Achse sowie die Reichweite die Werte, die einen Punkt im bearbeitbaren Raum ergeben.

Will man einen Roboter konstruieren, so muß man zunächst ermitteln, in welchem Koordinatensystem er die ihm zu übertragenden Aufgaben am besten erfüllen kann. Aus diesem System ergibt sich dann der Rechenaufwand, der für eine korrekte Positionierung zu treiben ist.

In unserem Modellfall mit den faulen Eiern wäre das kartesianische System ausreichend für die Positionierung des Schiebers.

Wir werden in einem der folgenden Abschnitte am Beispiel des fischertechnik-Modells "Greifarmroboters" sehen, wie die Berechnungen in einem sphärischen System erfolgen. Das Modell "Plotter" dagegen zeigt die Berechnungen im kartesianischen System.

Profi Sensoric und Profi Computing

Bisher haben wir in diesem Heft die Grundlagen für die Automations- und Robotertechnik dargestellt. In diesem Beitrag behandeln wir die Umsetzung in reale Modelle. Dazu dienen uns die fischertechnik-Konstruktionsbaukästen Profi Sensoric und Profi Computing.

Weihnachten 1965 wurden die ersten 1000 fischertechnik-Konstruktionsbaukästen der "Aktion Sorgenkind" gespendet. Aus einer Idee, mit der der Erfinder des inzwischen weltweit bekannten, oft kopierten und nie erreichten Fischer-S-Dübels, der spätere Professor Dr. h.c. Artur Fischer damals seinen Kunden die Verwendungsmöglichkeiten von Kunststoffen nahebringen wollte, war ein eigenes Produktkonzept geworden. Der Konstruktionsbaukasten, der ein Jahr später in Frankreich die Auszeichnung "Mailleur Jouet 1966" erhielt, hat sich seither zu einer ganzen Produktfamilie entwickelt. Zu dieser Familie gehören folgende Typreihen:

Junior und Junior PlusEinstiegsbaukästen für den ersten Kontakt Master und Master PlusSpezialthemen wie Motorisierung, Special-Trucks ProfiCartech, Sensoric, Computing

In der Reihe Special erhält man zu dem System passende Ergänzungsteile. Alle Typreihen können sowohl von Junioren als auch Senioren für den Einstieg zu oder für die Vertiefung von Kenntnissen der jeweiligen Konstruktionsthematik genutzt werden. In diesem Heft gehen wir nur auf die Konstruktionsbaukästen Profi Sensoric und Profi Computing ein. Wenn Sie sich über das gesamte Programm informieren wollen, wenden Sie sich bitte an die am Ende dieses Abschnitts stehenden Adressen.

Was Sie nicht in den Baukästen finden

Sie finden in beiden Profi-Konstruktionsbaukästen fast alles, was Sie zum Bau und Betrieb der Modelle brauchen. Ein wichtiges Teil fehlt aber in beiden, das Netzteil. Dies ist sicher deshalb vertretbar, weil viele Elektronik-Bastler eigene Trafos mit sechs bis neun Volt-Nennspannung haben oder auf Blockbatterien zurückgreifen wollen.

Netzteil (Power-Supply)

Die in den Profi-Konstruktionsbaukästen beschriebenen Modelle arbeiten mit Gleichstrom mit einer Nennspannung zwischen 6 und 9 Volt. Dafür kann man auch 9-Volt-Blockbatterien einsetzen. Im Sensoric-Konstruktionsbaukasten befindet sich dafür ein Batteriegehäuse mit einem Batterieclip für 9 V Blockbatterien. Wenn Sie die Anschaffung des nachstehend beschriebenen Netzteils scheuen, können Sie sich auch für den Profi-Computing-Baukasten eine solche Halterung besorgen.

Empfehlenswert ist die Anschaffung des fischertechnik-Netzgerätes (Power-Supply), das einen Wechselstrom von 230 V mit 50/60 Hz Frequenz (normale Werte des Haushaltsstromes) in einen Gleichstrom von 8 V umwandelt. Es kann aber auch jede andere Transformatoren-Einheit mit zwischen 6 und 9 Volt liegender Nennspannung verwendet werden. Durch die Umwandlung in Gleichstrom mit niedriger Spannung ist die Arbeit mit den Modellen völlig gefahrlos. Es empfiehlt sich aus Sicherheitsgründen, in regelmäßigem Turnus den Trafo und insbesondere die Zuleitung auf Beschädigungen zu überprüfen.

Werkzeuge

In beiden Baukästen wird ein Schraubendreher mitgeliefert, mit dem man die winzigen Schräubchen für die Klemmen anziehen kann. Empfehlenswert ist zumindest für die ersten Arbeiten eine (Elektroniker-)Abisolierzange. Nicht jeder ist nämlich in der Lage, die sehr dünnen Drähte ohne Beschädigung abzuisolieren. Zum Trennen der Adern nimmt man entweder eine Nagelschere oder (Juniors aufgepaßt) ein scharfes Messer.

Was Sie in den Baukästen finden

Als erster und sicher wichtiger Bestandteil beider Konstruktionsbaukästen finden Sie ein Handbuch, in dem jeweils erprobte Modelle und Programme zu ihrer Steuerung beschrieben werden. Der Aufbau aller Modelle wird in den Handbüchern detailliert beschrieben und durch großformatige Zeichnungen plastisch dargestellt. Wir zeigen, allerdings im wesentlich verkleinerten Format, einige dieser Zeichnungen unten im Abschnitt über den Profi-Computing-Baukasten. Der Zusammenbau wird in einzelnen Schritten durch farbige Kennzeichnung der Elemente und Teile beschrieben, so daß man sich nach kurzer Einübung gut damit zurechtfindet.

Desweiteren finden Sie in den Baukästen alle für den Bau der beschriebenen Modelle nötigen Bauelemente sowie die in der untenstehenden Beschreibung zu den Kästen dargestellten elektronischen und elektrischen Bauteile.

fischertechnik-Profi-Sensoric-Konstruktionsbaukasten

Die Sensorik ist der Teilbereich der Meßtechnik, der die Theorie und Praxis der elektronischen Sensoren behandelt. Wir sind in unserem Abschnitt "Messen, Steuern, Regeln" schon auf das Thema und die wichtigsten Typen von Sensoren eingegangen. Der Konstruktionsbaukasten fischertechnik-Profi-Sensoric beschäftigt sich mit diesem Bereich, das heißt, dem Einsatz von Sensoren. Am Anfang unserer Beschreibung des Baukastens wollen wir deshalb kurz die im Konstruktionsbaukasten befindlichen Sensoren und elektrischen Bauteile darstellen.

Das Flip-Flop

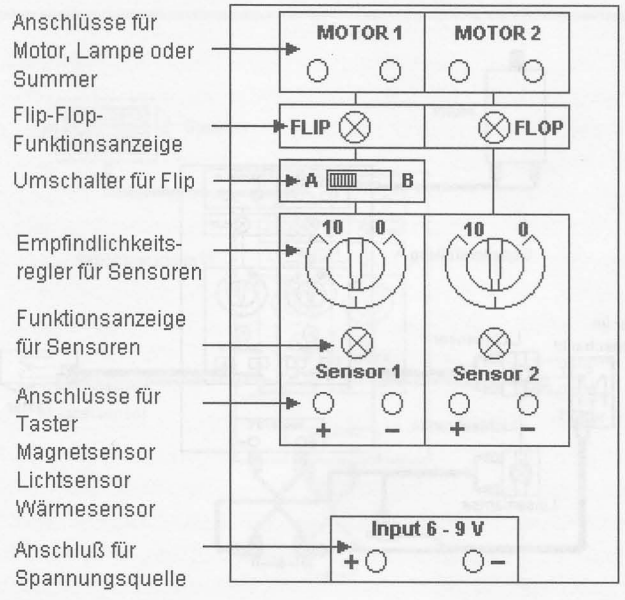
In unserem Bild "Das Flip-Flop in Aktion" sehen Sie eine schematisierte Ansicht des als Flip-Flop bezeichneten Bauteils aus dem fischertechnik-Profi-Sensoric-Baukasten. In ihm sind mehrere elektronische Funktionen für die im Konstruktionsbaukasten vorgesehenen Modelle zusammengefaßt. Das eigentliche Flip-Flop ist ein in dem Bauteil verborgener elektronischer Schalter, der links-rechts also "Flip" und "Flop" den Stromdurchgang zuläßt.

Die von Sensoren über die Anschlüsse Sensor 1 oder Sensor 2 kommenden (schwachen) Signale werden in dem Flip-Flop-Bauteil verstärkt und setzen das Flip-Flop oder setzen es zurück.

Ein Stromstoß setzt es, der nächste setzt es zurück und entsprechend wird der Stromdurchgang links oder rechts geschaltet. Ist beispielsweise ein Motor an einen der beiden Anschlüsse für die Motoren angeschlossen, läuft er bei Anschluß 1 bei gesetztem Flip-Flop (grüne Lampe), bei Anschluß 2 bei rückgesetztem Flip-Flop (rote Lampe). Die Funktionsweise des Flip-Flop zeigen am einfachsten je eine Schaltung mit den zwei Tastern und mit dem Lichtsensor und einem Taster.

Motor und Getriebe

Der Motor ist ein Gleichspannungsmotor, der den zum Antrieb erforderlichen Strom über das Flip-Flop (siehe oben) vom Netzteil oder der Batterie erhält. Da der Motor sehr hochtourig läuft, ist ein Untersetzungsgetriebe im Konstruktionsbaukasten enthalten, durch das die Umdrehungszahl reduziert und gleichzeitig die Kraft des Motors verstärkt wird.



Das Flip-Flop

Summer und Lampen

An einem der Motoranschlüsse des Flip-Flop kann auch der im Konstruktionsbaukasten enthaltene Summer angeschlossen werden. Er gibt dann solange einen Signalton ab, wie die entsprechende Stellung des Flip-Flop geschaltet ist. Der Summer arbeitet mit maximal 12 V Spannung. In ähnlicher Weise können die Lämpchen aus dem Konstruktionsbaukasten an einem Motorausgang angeschlossen werden.

Sensoren und Taster

Im Konstruktionsbaukasten befinden sich folgende drei Sensortypen:

Wärmesensor

Ein NTC-Widerstand dient als Wärmesensor. Was ein NTC-Widerstand ist, haben wir schon im Abschnitt "Messen, Steuern, Regeln" unter "Fachbegriffe, kurz erklärt" dargestellt. Er leitet den Strom besser mit steigenden Temperaturen. Der NTC des Sensoric-Baukastens hat einen Widerstand von 60 kW bei 20 Grad Celsius.

Magnetsensor

Der Reedkontakt wird als Magnetsensor verwendet. Auch dessen Funktion haben wir beschrieben. Der Reedkontakt wird geschlossen, wenn ein Magnetfeld entsprechender Stärke auf ihn einwirkt. Die maximale Spannung des Reedkontaktes aus dem Konstruktionsbaukasten liegt bei 12 V. Im Baukasten befindet sich ein Reed-Kontakt.

Lichtsensor

Fototransistor als Lichtsensor. Bei einem Fototransistor ist die Leitfähigkeit direkt proportional der auf den Sensor fallenden Lichtenergie. Bei dem Fototransistor muß, wie bei jedem Transistor, auf genaue Polung geachtet werden, da er nur in einer Richtung den Strom durchläßt.

Wichtig: Der Fototransistor darf nicht an einen der Motorausgänge angehängt werden, da er zerstört würde.

Taster

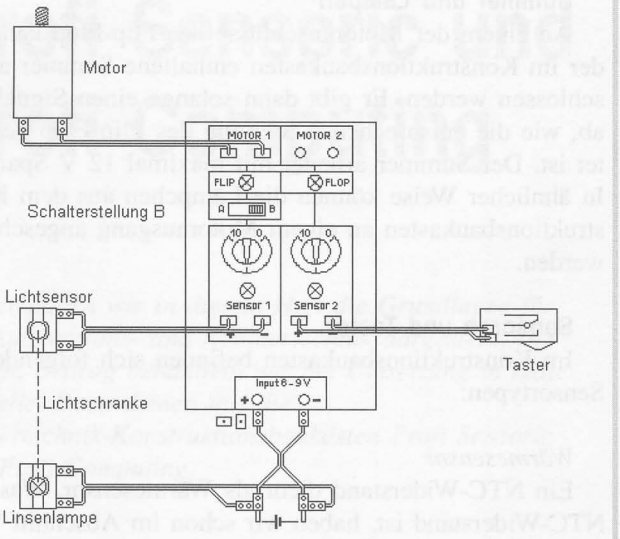
Anstelle der Sensoren kann dem Flip-Flop ein Signal auch durch Taster gegeben werden. Im Konstruktionsbaukasten befinden sich zwei Taster.

Erweiterungen

Man kann natürlich auch eigene Bauteile über das Flip-Flop schalten. Voraussetzung ist nur, daß die Teile im vorgegebenen Spannungsbereich zwischen 6 und 9 V (maximal 12 Volt) arbeiten.

Die Modelle

An den im Handbuch zum Baukasten beschriebenen Modellen wollen wir jetzt zeigen, wie man mit dem fischertechnik Profi-Sensoric-Konstruktionsbaukasten arbeitet und wie man die im Baukasten enthaltenen elektrischen und elektronischen Bauteile einsetzt. Im Handbuch werden folgende Modelle beschrieben:



Verdrahtungsschema Händetrockner

Händetrockner

Der Händetrockner setzt bei Unterbrechung einer Lichtschranke über das Flip-Flop einen Motor in Betrieb, an dem eine Luftschraube befestigt ist. Mit einer Taste wird der Motor dann wieder ausgeschaltet. Arbeitsweise und Funktion der Lichtschranke werden an diesem Beispiel gezeigt. Ein Lichtsensor empfängt im Ruhezustand Licht von einer Linsenstecklampe. Wird diese Lichtschranke unterbrochen, wird der Motor des Ventilators in Gang gesetzt. Die Funktion der Lichtschranke ist hier eine logische NOT-Funktion:

Wenn das Licht NICHT auf den Lichtsensor fällt, läuft der Motor.

In der Praxis verwendet man für Lichtschranken anstelle der Kombination für sichtbares Licht (Lampe, Fotosensor) pulsierendes Infrarotlicht, also in kurzen Blitzen ausgesandtes Infrarotlicht und den entsprechenden Infrarotsensor.

(Klein-)Geldautomat mit Scheckkarte

Der Geldautomat zeigt das Prinzip der Kreditkartenautomaten mit Codekarten. Im Modell wird anstelle der in der Praxis verwendeten Magnetkarten eine gelochte Pappkarte eingesetzt. Diese Karte wird in den Kartenschlitz eingeschoben. In diesem Schlitz befinden sich ein Taster und eine Lichtschranke. Taster und Lichtschranke wirken für die Ausgabe der Münzen zusammen. Erst wenn der Taster getätigt und die Lochmarkierung sich genau an der richtigen Stelle der Codekarte befindet, wird die Münze freigegeben. Das Zusammenwirken von Taster und Lichtschranke bildet eine logische AND-Funktion:

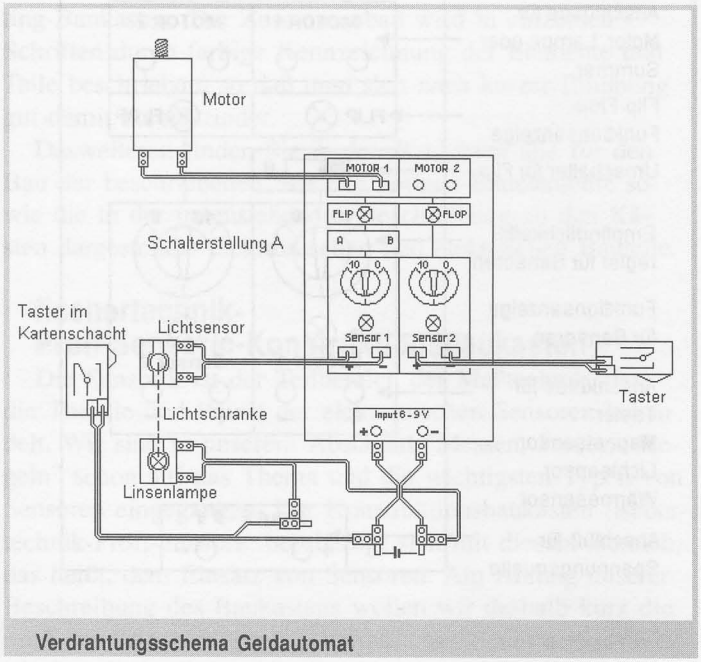
Wenn Lichtschranke frei UND Taster gedrückt, dann Motorstart

In der Praxis arbeitet man heute mit Magnetkarten und, zunehmend, mit Chipkarten. Das Prinzip der Magnetkarten entpricht dem der strichcodierten Barcode-Karten . Auf der Barcode-Karte befinden sich schwarze Balkenmarkierungen (daher der Name: Bar = Balken). Breite und Abstand der jeweiligen Balken werden zur Codierung verwendet. Auf der Magnetkarte sind es Bänder von magnetisierten und nicht magnetisierten Streifen, die jeweils Bit-gesetzt und Bit-nicht-gesetzt bedeuten. Während der Barcode ein Streifen ist, der in Klarschrift oder verschlüsselt die Informationen enthält, befinden sich auf den Magnetstreifen mehrere Spuren, die unterschiedliche Funktionen haben können und jeweils von einem Schreibkopf magnetisiert also beschrieben oder von einem Lesekopf gelesen werden.

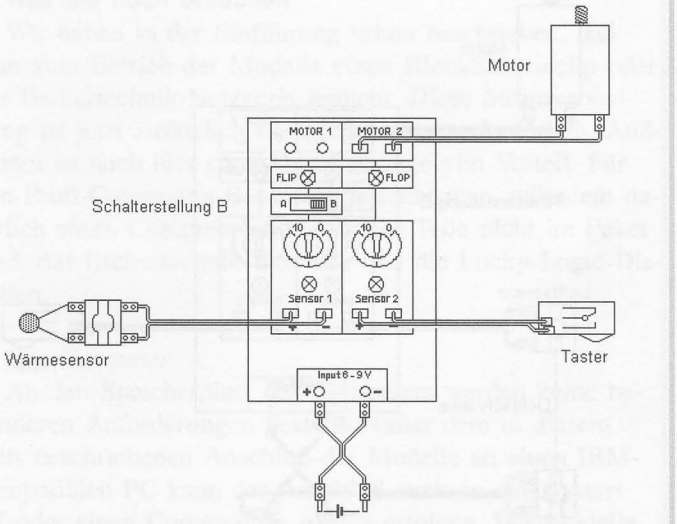
Zur Sicherheit gegen unbeabsichtigtes Löschen und Fälschung kann die Magnetisierung des Magnetstreifens mit niedriger (LoCo = Low-Codet = niedrig codiert) oder mit hoher Feldstärke (HiCo = High-Codet = hoch codiert) erfolgen. Die Chipkarte, bei der sich auf der Karte ein Chip befindet, bietet zusätzliche Sicherheiten gegen Kopieren und Fälschen. Der Chip ist ein Speicherchip, dessen Inhalt im Schreib-/Lesegerät eingelesen und geändert wird. Der Chip bietet eine um vieles höhere Speicherdichte und damit mehr Möglichkeiten der Verschlüsselung. Er wird durch magnetische Felder nicht verändert.

Eierkühlmaschine

An diesem Modell wird die Funktion von NTC-Wärmewiderständen (Beschreibung im Abschnitt "Messen, Steuern, Regekn") gezeigt. Diese haben einen negativen Temperaturkoeffizienten, das heißt, sie leiten den elektrischen Strom bei höherer Temperatur besser. Die Eierkühlmaschine arbeitet mit einem ähnlichen Prinzip wie das Händetrocknermodell. Bei ihr wird anstelle der Lichtschranke ein Wärmesensor (NTC-Widerstand) eingesetzt. Die Wärme des warmen Eis verringert den Widerstand im NTC und setzt das Flip-Flop. Dadurch wird der Ventilatormotor



Verdrahtungsschema Geldautomat



Verdrahtungsschema Eierkühlmaschine

gestartet. Unterschreitet die Temperatur des Eis einen Grenzwert, wird das Flip-Flop zurückgesetzt und der Ventilatormotor gestoppt.

Schatztruhe als Tresor

Dieses Modell zeigt den Einsatz von Magnetsensoren zur Türsicherung. Das Prinzip ist ganz einfach. Wenn der Magnetsensor sich in einem magnetsichen Feld ausreichender Stärke befindet, wird von ihm ein Stromkreis geschlossen. Magnet und Sensor werden so an Deckel und Truhe montiert, daß bei geschlossenem Deckel der Sensor den Stromkreis schließt. Wird der Deckel der Truhe geöffnet, wird der Stromkreis unterbrochen, das Flip-Flop gesetzt und der Summer tritt in Aktion.

Presse oder Stempelmaschine

Auch dieses Modell verwendet die Lichtschranke. Am Modell wird gezeigt, wie Kartenstempler funktionieren. Unterbricht eine Karte den Lichtstrahl der Schranke, wird ein Stempelwerk in Betrieb gesetzt. In ähnlicher Weise arbeiten Sicherheitseinrichtungen an Pressen. Das Modell ist auch deshalb interessant, weil an ihm die Umsetzung von drehender (Motor-)Bewegung in Auf-/Ab-Bewegung gezeigt wird. Das Verdrahtungsschema entspricht dem beim Händetrockner.

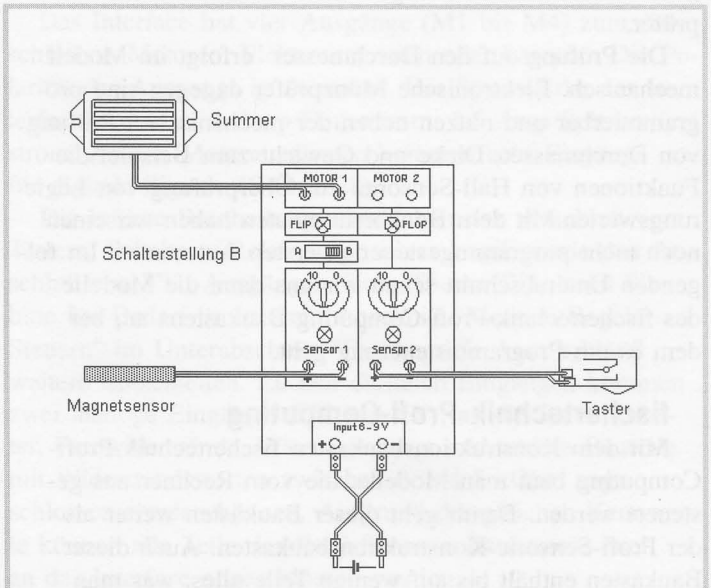
Sortierband

Am Beispiel der Sortierung von magnetisierbarem und nicht magnetisierbarem Metall wird die Funktion des Magnetsensors (Reed-Kontakt) gezeigt. Dabei wird wegen der geringen Empfindlichkeit des Reed-Kontaktes der mitgelieferte Dauermagnet an einem Baustein aus dem Baukasten befestigt. Diesen kann der Senosr jetzt von den anderen Teilen ohne Dauermagnet trennen. Im Modell gilt das logische OR:

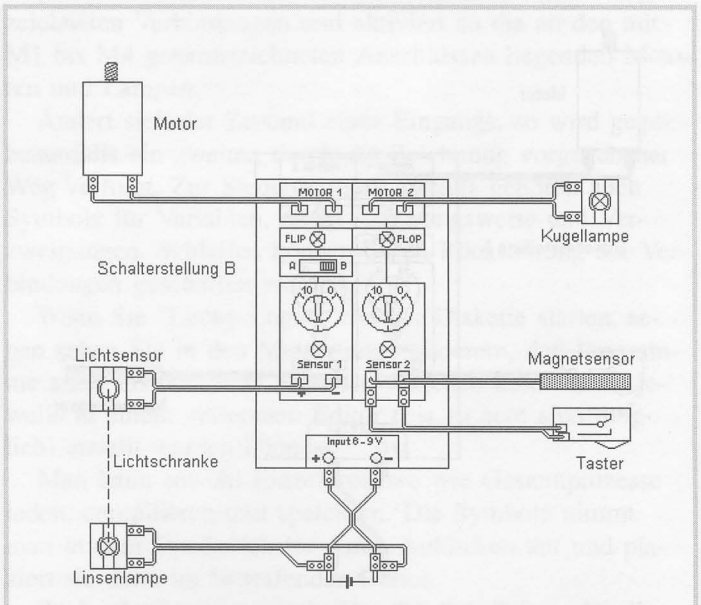
Wenn der Magnetsensor anspricht ODER der Taster gedrückt ist, dann wird der Motor gestoppt.In der Praxis werden empfindlichere Sensoren (Hall-Sensoren, Feldplatten) verwendet, um etwa in einer Recycling-Anlage Eisen von Nichteisen-Metallen zu trennen.

Reaktionsspiel (und Geschicklichkeitsspiel)

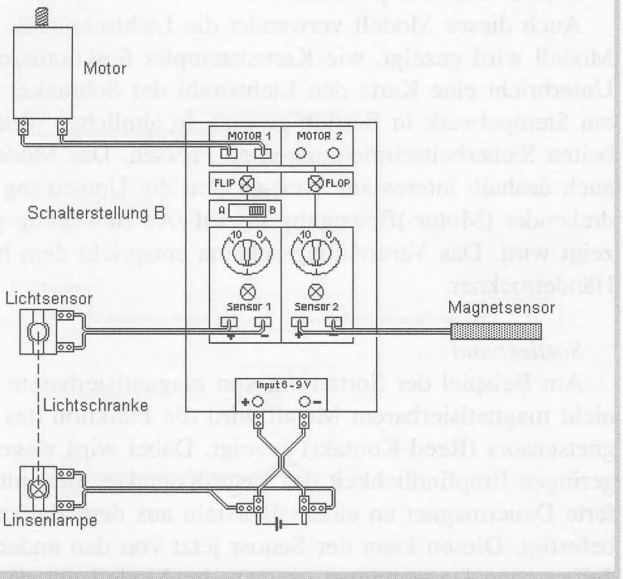
Eine Kombination von Lichtschranke und Magnetfeld ist Grundlage dieses Modells. Ziel des Reaktionsspiels ist es, so schnell zu sein, daß man einen Magneten fassen kann, der nach Unterbrechung der Lichtschranke durch die Hand mit dem Motor schnell nach hinten gezogen wird. Beim Geschicklichkeitsspiel wird der Baustein nach hinten gezogen, wenn er angehoben wird. An diesem Modell



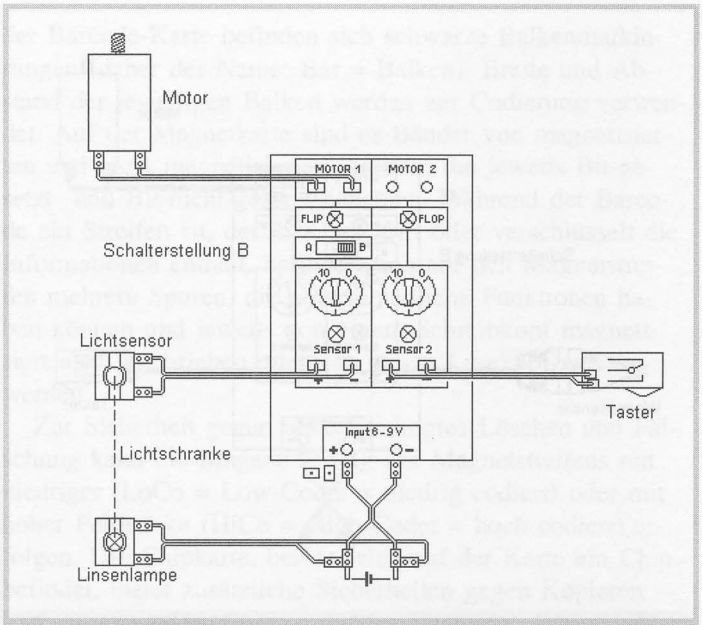
Verdrahtungsschema Schatztruhe



Verdrahtungsschema Sortierband



Verdrahtungsschema Reaktionsspiel



Verdrahtungsschema Bonbonautomat

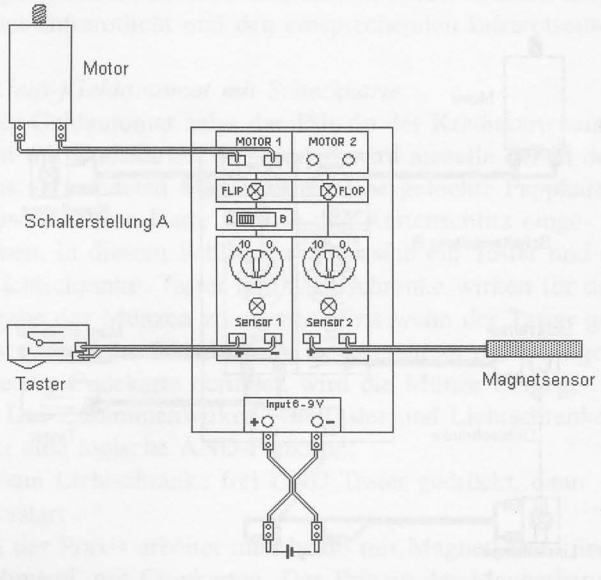
lernt man im wahrsten Sinne des Wortes spielerisch die Wirkung von Lichtschranken kennen.

Dosieranlage

In der Dosieranlage wird das Schüttgut (fischertechnik-Bausteine) über ein Förderband zu einer Auffangschale transportiert, die an einem Waagebalken befestigt ist. An der anderen Seite des Waagebalkens ist ein Dauermagnet montiert. Ein Magnetsensor schließt den Kontakt, wenn der am Waagebalken montierte Magnet in seine unmittelbare Nähe kommt, und das Band wird gestoppt.

Garageneinfahrt mit Schranke

In diesem Modell wird das System der automatischen Schrankenöffnung simuliert. Die Schranke wird geöffnet



Verdrahtungsschema Dosieranlage

und eine Ampelanlage schaltet von Rot auf Grün, wenn ein berechtigtes Fahrzeug in den Kontaktbereich kommt. Dieses Modell reizt dazu, es z.B. mit dem Kartenprüfer beim Scheckkartenautomaten zu erweitern. Für diesen Fall muß man aber weitere Taster haben. Bitte wenden Sie sich in solchen Fällen an die Fischerwerke.

Bonbonautomat

Warenautomaten stehen bei uns überall. Und wie Sie aus dem einführenden Teil wissen, sind Warenautomaten in Form der Gänsekiel- oder Weihwasserspender die ersten Vorläufer unserer heutigen Automation gewesen. Es ist deshalb sehr reizvoll, sich das Prinzip solcher Automaten einmal näher anzusehen. Mit dem beschriebenen Münz-(Durchmesser-)Prüfer hat man einen der Vorläufer der heute in Warenautomaten eingesetzten elektronischen Münzprüfer.

Die Prüfung auf den Durchmesser erfolgt im Modell mechanisch. Elektronische Münzprüfer dagegen sind programmierbar und nutzen neben der mechanischen Prüfung von Durchmesser, Dicke und Gewicht zum Beispiel die Funktionen von Hall-Sensoren zur Überprüfung von Legierungswerten.Mit dem Bonbonautomaten haben wir einen noch nicht programmgesteuerten ersten Automaten. Im folgenden Unterabschnitt sehen wir uns dann die Modelle des fischertechnik-Profi-Computing-Baukastens an, bei dem es um Programmsteuerung geht.

fischertechnik Profi-Computing

Mit dem Konstruktionsbaukasten fischertechnik-Profi-Computing baut man Modelle, die vom Rechner aus gesteuert werden. Damit geht dieser Baukasten weiter als der Profi-Sensoric-Konstruktionsbaukasten. Auch dieser Baukasten enthält bis auf wenige Teile alles, was man zum Bau und Betrieb der Modelle braucht. Einige Teile sind aber zusätzlich erforderlich oder empfehlenswert.

Was Sie noch brauchen

Wir haben in der Einführung schon beschrieben, daß man zum Betrieb der Modelle einen Blockbatterieclip oder das fischertechnik-Netzgerät braucht. Diese Stromversorgung ist jetzt zusätzlich für das Interface erforderlich. Außerdem ist auch hier eine Abisolierzange von Vorteil. Für den Profi-Computing-Baukasten braucht man außerdem natürlich einen Computer und, da diese Teile nicht im Paket sind, das fischertechnik-Interface und die Lucky-Logic-Disketten.

Der Computer

An den Speicherplatz des Computers werden keine besonderen Anforderungen gestellt. Außer dem in diesem Heft beschriebenen Anschluß der Modelle an einen IBM-kompatiblen PC kann der Anschluß auch an einen Atari ST oder einen Commodore Amiga erfolgen. Die Modelle werden unabhängig vom Rechnertyp gleich erstellt. Wollen Sie ständig Modelle über das Interface steuern, so empfiehlt sich der Einbau einer zweiten parallelen Schnittstelle.

Das fischertechnik-Interface

Das fischertechnik-Interface dient der Übermittlung von Sensorsignalen an den Rechner, steuert die Motoren und ihre Laufrichtung und schützt die elektronischen Teile gegen Überspannung. Außerdem schaltet es die Stromversorgung der Motoren aus, wenn längere Zeit kein Steuersignal vom Rechner kommt. Damit werden diese gegen Überlastung geschützt.

Das Interface braucht eine Stromversorgung (sechs bis neun Volt) von der Blockbatterie, dem fischertechnik-Netzgerät oder einem anderen Trafo. Es wird über die parallele (Drucker-)Schnittstelle an den Rechner angekoppelt. Wenn man das lästige Umstecken vermeiden will, sollte man für den Drucker eine zweite parallele Schnittstelle in den Rechner einbauen.

Das Interface hat vier Ausgänge (M1 bis M4) zum Anschluß an Motoren, Elektromagneten oder Lampen. Die Polarität des Ausgangs ist steuerbar. Die Stromstärke des Interface darf sein: 1 Amp Dauerstrom, 1,5 Amp Spitzenstrom. Zu den Motorausgängen kommen acht Eingänge für digitale Signale (E1 bis E8).

Die interne Beschaltung erlaubt hier den Anschluß von Tastern, Schaltern, Relais in positiver Logik sowie den Anschluß von TTL-Ausgängen. Zum Thema TTL lesen Sie bitte bei Bedarf in unserem Abschnitt "Messen, Regeln, Steuern" im Unterabschnitt "Fachbegriffe, kurz erklärt" weitere Einzelheiten. Zu den digitalen Eingängen kommen zwei analoge Eingänge (EX und EY), an die Potentiometer, Fotowiderstände, Wärmesensoren und andere Bauteile mit Widerstandswerten zwischen 0 und 5 kOhm angeschlossen werden können. Aufgrund der gleichen Kennwerte können alle Teile des Profi-Sensoric-Baukastens auch an das Interface angeschlossen werden.

Zur Verbindung des Interface mit dem Rechner dient ein 20-poliges Flachbandkabel, das mit dem Interface fest verbunden ist. An dessen Steckerseite befindet sich eine spezielle 20-polige Buchse. Die Verbindung zum Rechner erfolgt über einen mitgelieferten Adapter, auf dessen einer Seite ein zu der Buchse passender 20-poliger Stecker und auf der anderen Seite ein 25-poliger Sub-D-Stecker den Anschluß an die Buchse im Rechner übernimmt.

An das fischertechnik-Interface kann zur Verdoppelung der Anschlüsse ein zweites Interface mit seinem 20-poligen Stecker angeschlossen werden.

Für die Verbindung zwischen dem Interface und den Modellen befindet sich im Baukasten ein 20-poliges Flachbandkabel mit einem montierten Stecker. Den Anschluß der entsprechenden Buchse an der anderen Seite nimmt man selbst vor (siehe "Nach dem Auspacken").

fischertechnik-Lucky-Logic

Die Steuerung der Modelle des fischertechnik-Profi-Computing-Baukastens kann man zur Zeit mit Turbo-Pascal oder mit dem von den Fischerwerken dafür entwickelten Programmiersystem Lucky-Logic programmieren. Lukky-Logic ist für den PC, Atari ST und Comodore Amiga verfügbar. Auf unserer Diskette finden Sie eine Demoversion des Programmiersystems ohne die Schnittstelle zur Modellsteuerung. Sie können damit aber vollständige, auch mehrmodulige Programme in Lucky-Logic erstellen, auf Fehler überprüfen und speichern. Nutzen Sie diese Möglichkeit zum Probieren.

Interessant ist die Möglichkeit, Symbole auf dem Bildschirm zu plazieren, miteinander zu verbinden und den Programmablauf mit Run auf Fehler zu überprüfen. Um festzulegen, was das Modell tun soll, kann man die einzelnen Symbole für die Schalter, Taster, für Lampen und Motoren auf dem Bildschirm anordnen und dann mit der Maus die Verbindungen ziehen - so ähnlich, wie sie in einer Schaltung mit "echten" Drähten verbunden werden. Das Lucky-Logic-Steuerprogramm folgt entsprechend den eingehenden Signalen an den Interface-Eingängen den gezeichneten Verbindungen und aktiviert so die an den mit M1 bis M4 gekennzeichneten Anschlüssen liegenden Motoren und Lampen.

Ändert sich der Zustand eines Eingangs, so wird gegebenenfalls ein zweiter, durch die Zeichnung vorgegebener Weg verfolgt. Zur Steuerung des Ablaufs gehören auch Symbole für Variablen, deren Änderungswerte und Verzweigungen. Schleifen können durch Rückführung der Verbindungen geschaffen werden.

Wenn Sie "Lucky-Logic" von der Diskette starten, sehen sehen Sie in den Menüs unter anderem, daß Programme auch aus mehreren Modulen bestehen können, die jeweils in einem getrennten Editor (bis zu acht sind möglich) erstellt werden können.

Man kann sowohl Einzelprozesse wie Gesamtprozesse laden, compilieren und speichern. Die Symbole nimmt man in dem Symbolfenster durch Anklicken auf und plaziert sie dann im betreffenden Editor.

Zu Lucky-Logic wird ein Handbuch geliefert, das die Arbeit mit dem Programm und alle Aktionen im Detail beschreibt.

Turbo-Pascal

Das Programmiersystem Turbo-Pascal gehört natürlich nicht zum Zubehör des Baukastens. Da aber einige Programme in dieser Programmiersprache geschrieben sind, soll hier ein Hinweis erfolgen: Die Turbo-Pascal-Programme greifen unter anderem auf das Unit Fischer zu. Dieses findet man auch auf den Disketten zur Lucky-Logic-Vollversion.

Nach dem Auspacken

Auch im Profi-Computing-Konstruktionsbaukasten findet man nach dem Auspacken diverse Plastiktüten mit Kleinteilen. Und auch hier hat man gleich zu Anfang eine Aufgabe, die der Autor für den einzigen Schwachpunkt hält und die zu manchen "Freudenrufen" Anlaß war. Denn am Anfang steht das Zusammenfügen von Ministeckern und einer 20-poligen Steckleiste.

Die dabei anzubringenden Schräubchen sind so klein, daß schwer mit den Händen arbeitende Väter diese Aufgabe lieber ihren Junioren übertragen sollten. Auf jeden Fall sollte man den Tisch frei machen. Ein dunkles Tuch als Abdeckung hilft, die Winzlinge nicht vom Tisch springen zu lassen.

Schließen Sie die Montage der in einem getrennten Beutel vorhandenen Ministecker gleich als weitere Kniffelarbeit an, dann haben Sie diesen Teil hinter sich und können mit dem Aufbau der ersten Modelle beginnen.

Die Modelle

Im Handbuch zum Profi-Computing-Baukasten sind folgende Modelle beschrieben:

Reaktionstester

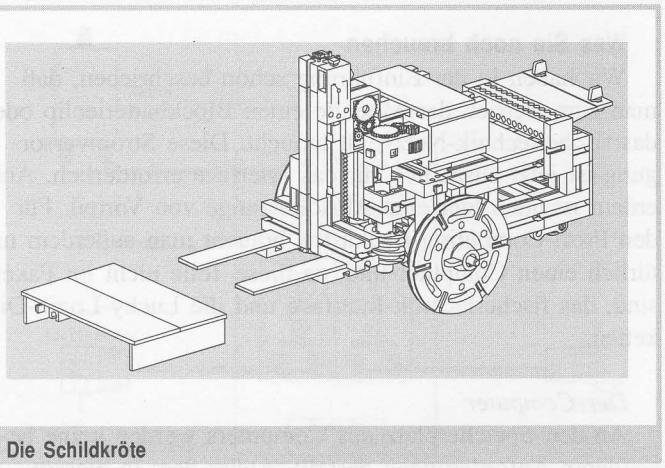
Mit ihm testet man die Reaktionsgeschwindigkeit. Eine Lampe leuchtet zufallsgesteuert auf und es wird die Zeit gemessen von diesem Aufleuchten bis zum Druck auf die Taste. Der Versuchsaufbau ist sehr einfach. Das Programm dazu ist in Turbo-Pascal geschrieben. Sie finden es hier im Listing "Reaktionstester".

Melodie-Spiel

Der Computer spielt zufallsgesteuert eine 30 Töne lange Melodie aus vier Tönen. Es geht darum, diese Melodie nachzuspielen. Dazu geht das Programm wie folgt vor: Es wird zunächst die Melodie und danach deren erster Ton gespielt. Gleichzeitig leuchtet in dem zum Ton gehörenden Taster eine Kugellampe auf.

Mit Druck auf diese Taste bestätigt man, daß man sich Ton und Taste gemerkt hat. Dann werden zwei Töne, drei, vier und so weiter vorgespielt, die man sich merken soll und deren Tasten man danach in der richtigen Reihenfolge drücken muß.

Das Listing in Turbo-Pascal am Ende des Abschnitts zeigt im Detail den Ablauf.



Die Schildkröte

Turtle (Schildkröte)

Auf dieses Modell sind wir schon im Abschnitt "Fuzzy-Logic ..." eingegangen. Der Autor der dort genannten Bücher, Thomas Tilli, hat das Modell genutzt, um eine Fuzzy-Logic-Steuerung zu realisieren. Vorbild für die Turtle ist das gleichnamige Mondauto, mit dem die amerikanischen Astronauten auf dem Mond herumgefahren sind.

Beim Modell besitzt jedes Rad einen eigenen Antrieb, so daß es sich unabhängig vom anderen vorwärts und rückwärts drehen kann. Dadurch sind beliebige Kurven und sogar Drehen auf der Stelle möglich. Das Turtle-Programm finden Sie in Lucky-Logic auf der Diskette. Unser Bild zeigt das fertige Modell.

Tresor mit Codeschloß

Hier finden Sie ein Modell, das weit höheren Sicherheitsanforderungen entspricht als die Schatztruhe aus dem Sensoric-Baukasten. Vier Schalter sind in der richtigen Reihenfolge zu betätigen, damit sich die "Tresortür" öffnet. Auch das Lucky-Logic-Programm zu diesem Modell finden Sie auf der Diskette.

(Wechsel-)Geldautomat

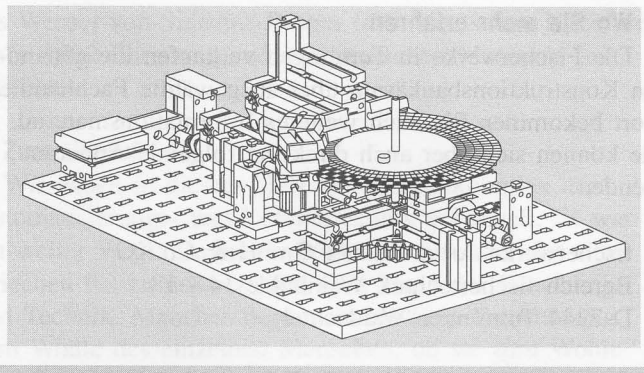
Aus drei mit Münzen gefüllten Röhrchen wird einem der gewünschte Betrag in Einzelmünzen ausgegeben. Das Programm ist in Lucky-Logic geschrieben und steht auf der Diskette.

Codekartenleser mit automatischem Einzug

Bei der Beschreibung des Sensoric-Baukastens sind wir schon auf dieses Thema gestoßen. Auch im Profi-Computing-Modell zu diesem Thema wird anstelle von Infrarot- mit sichtbarem Licht gearbeitet. An zwei Stellen wird die Codekarte überprüft, ob sie dort die Lochungen hat. Die Karte wird zu diesem Zweck eingezogen, dann geprüft und am Ende wieder ausgegeben. Programmiert ist auch die Steuerung hierzu in Lucky-Logic (siehe Diskette).

CD-Player

Die "richtige" Compact Disc enthält digital gespeichert Töne und neuerdings auch Bilder und Texte. Auch das CD-



Der CD-Player

Player-Modell arbeitet digital. Mit Hilfe einer Lichtschranke, die den Lesekopf repräsentiert, werden undurchlässige und durchlässige Segmente auf einer Papierdiskette erkannt und abgespeichert. Jeweils 4 Bit (von maximal 75 auf drei Spuren verteilte) werden zu einem Ton zusammengefaßt, so daß 2^4 = 16 verschiedene Töne darstellbar sind. So kann man eigene Melodien speichern.

Wenn Sie viel mit dem Computer arbeiten, ist Ihnen möglicherweise auch das Paperdisk-Konzept der Zeitschrift MD bekannt. Dort werden Programme und Informationen als helle und dunkle Stellen auf Papier festgehalten. Mit einem Scanner (zum Beispiel einem Scanman von Logitech, Logi GmbH, München) liest man diese Informationen ein und übersetzt sie mit einem von MC beziehbaren Programm.

Ähnlich arbeitet das im Listing stehende Programm READCD.PAS am Ende des Abschnitts.

Sortieranlage

Das Sensoric-Modell ist hier verbessert. Es können jetzt verschiedene Teile erkannt und getrennt voneinander abgelegt werden. Die Software für dieses Modell ist in Lucky-Logic geschrieben. Sie finden die zusammengehörenden Module SORTIER.FTG, SORTIER1.FTL bis SORTIER4.FTL auf der beiliegenden Diskette.

Paketwendeanlage

Damit die Anschrift sichtbar wird, müssen in Postämtern die Pakete gewendet werden. Das Profi-Computing-Modell löst diese Aufgabe, indem eine "Paket" über eine Rutsche geführt und so lange gedreht wird, bis die schwarze Seite oben liegt. Das Paket wird dazu quer über die Rutsche in Richtung Band geschoben. Dort wird es unter einer Reflex-Lichtschranke geprüft.

Über die unterschiedliche Reflexion der gelben und schwarzen Oberflächen stellt das Programm fest, ob die dunkle Seite oben liegt. Muß das Paket gewendet werden, transportiert das Band es weiter zur Wendestation.

Dort wird es angehoben und auf das laufende Band gekippt. Nun wird das Paket zurück zur Reflex-Lichtschranke befördert, wo entweder der Vorgang erneut beginnt oder das richtig liegende Paket auf die Ausgaberutsche befördert wird. Das Programm in Lucky-Logic finden Sie als PAKET auf der Diskette.

Kurvenschreiber

Der Kurvenschreiber ist ein Modell, mit dem man auf zweierlei Art arbeiten kann: Das Papier, auf dem geschrieben wird, kann in festen Zeitabständen bewegt werden. Dann zeichnet der Stift den zeitlichen Ablauf der aufgezeichneten Daten. Beim Vor- und Zurückbewegen zeichnet der Schreiber Schlangenlinien. Sie sehen im folgenden Listing das Turbo-Pascal-Programm.

Plotter

Jeder Computerbesitzer, der Zeichnungen erstellt, wünscht sich einen Plotter. Im Profi-Computing-Modell werden am Beispiel eines DIN-A5-Plotters Aufbau und Wirkungsweise demonstriert. In unserer Bilderleiste sehen Sie die einzelnen Schritte beim Zusammenbau des Plotter-Modells.

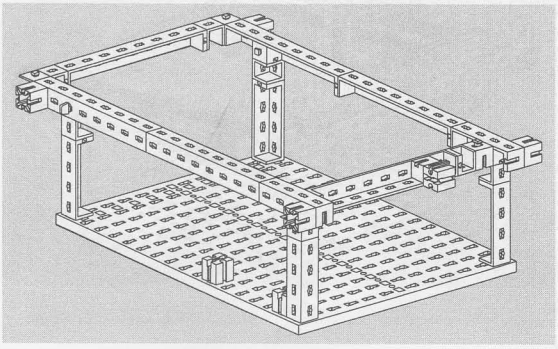
Das Turbo-Pascal-Programm zeigt, wie man die einzelnen Stifte ansteuert.

Roboter

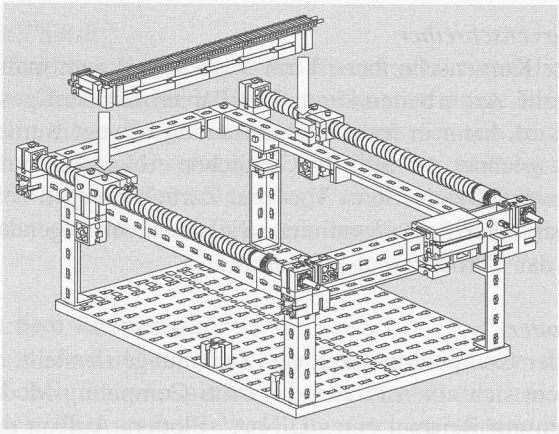
Das letzte im Handbuch beschriebene Modell ist sicher für viele das interessanteste: Ein in zwei Richtungen drehbarer Greifarmroboter mit einem Arbeitsradius von 30 Zentimetern. Wir sind in unserem Abschnitt "Die Umsetzung der Theorie" im Detail auf Roboter - wie man sie baut und wie man sie einsetzt - eingegangen. Hier sollen deshalb nur die Bilder vom Bau des Modells zeigen, wie interessant die Arbeit mit dem Konstruktionsbaukasten fischertechnik-Profi-Computing ist. Das Lucky-Logic-Programm dazu finden Sie auf der beiliegenden Diskette.

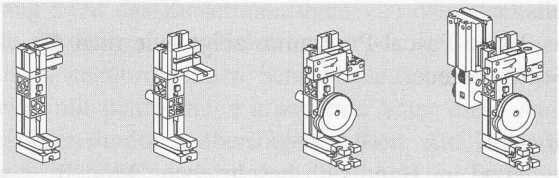
Keine Steuerung unter BASIC?

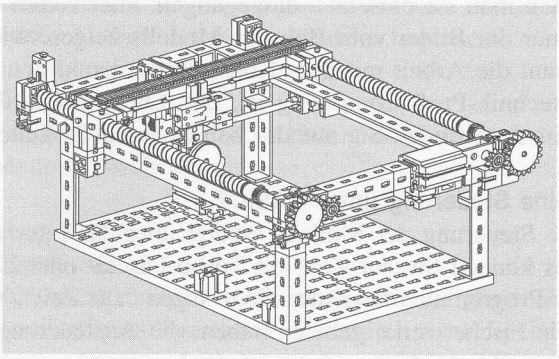
Die Steuerung der Modelle erfolgt über das Interface. Dieses können Sie zur Zeit mit Turbo-Pascal- oder Lucky-Logic-Programmen ansteuern. Wir sagen "zur Zeit", weil wir die Fischerwerke gebeten haben, die Ansteuerung der Schnittstelle, die bisher nicht offengelegt wurde, auch den BASIC-Programmierern zu ermöglichen. Wenn Sie BASIC-Programmierer sind, sollten Sie uns bei unserem Bemühen unterstützen.

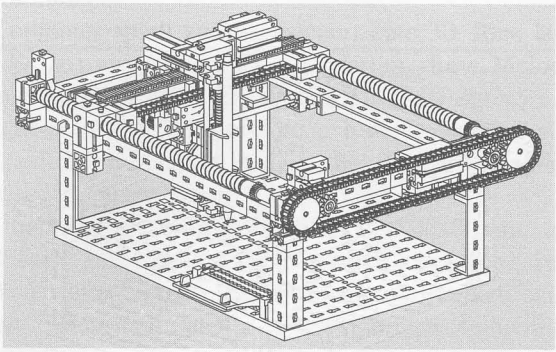


Wie der Plotter entsteht, Ausbaustufe 1









Wie der Plotter entsteht, Ausbaustufe 2 bis 5

Wo Sie mehr erfahren

Die Fischerwerke in Tumlingen verkaufen die genannten Konstruktionsbaukästen über ausgewählte Fachhändler. Dort bekommen Sie auch weiteres Informationsmaterial. Sie können sich aber auch direkt an folgende Adressen wenden:

fischerwerke Artur Fischer GmbH & Co.KG
Bereich fischertechnik Weinhalde 14 - 18
D-7244 Tumlingen/Waldachtal

fischer austria GmbH & Co.KG
Wiener Straße 95
A-2513 Möllersdorf/Traiskirchen

LEMACO SA
Chemin du Croset 9
CH-1024 Ecublens

Noch ein Tip für fischertechnik-Fans: Es gibt auch einen fischertechnik-Fanclub, der eigene Fan-Club-News herausgibt. Mitglied in diesem Fanclub kann jeder ohne Mitgliedsbeitrag werden.

Das Werk und sein Gründer

Es ist immer sinnvoll, daran zu erinnern, wie etwas wurde, was es ist. Betrachten wir deshalb zum Abschluß dieses Abschnitts in Kurzform den Weg, den der Gründer und sein Werk gegangen sind.

Am 31. Dezember 1919, der erste Weltkrieg war gerade ein Jahr vergangen, wurde Artur Fischer als Sohn des Tumlinger Bürgers Georg Fischer in Tumlingen am Ostrand des Schwarzwaldes geboren. 1930 bis 1933 besuchte er im acht Kilometer entfernten Dornstetten die Realschule. Damals gab es übrigens noch keinen Schulbus. Sommers wie winters mußten Artur Fischer und seine Schulkameraden mit dem Fahrrad in die Schule fahren.

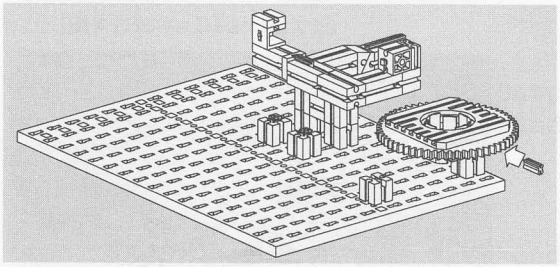
1933 bis 1937 ermöglichten die Eltern unter großen finanziellen Opfern Artur Fischer die Schlosserlehre in Stuttgart. 1938 bis 1946 waren die Zeiten des Arbeits-, später Wehrdienstes und der Gefangenschaft. 1947 arbeitete Artur Fischer für einen Ingenieur in Freudenstadt und heiratete. 1948 machte sich Artur Fischer in Hörschweiler "mit 40,- DM in der Tasche, viel Ideen und noch mehr unternehmerischem Mut" selbstständig, 1949 erhielt Fischer sein erstes Patent für ein "Magnesium-Blitzlichtgerät", das er 1950 auf der Photokina in Köln ausstellte.

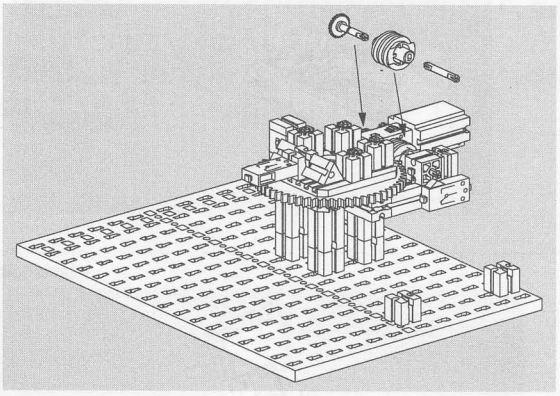
Die Agfa in München wurde dabei auf ihn aufmerksam. Im gleichen Jahr wurde der Sohn Klaus (17.08.1950) geboren. 1958 wurde der Fischer-S-Dübel, der erste Kunststoffdübel patentiert. 1965 wurden die fischertechnik-Konstruktionskästen erstmals vorgestellt. 1969 Der Umsatz überstieg 40 Millionen DM. 1976 Ehrendoktor der Universität Gießen 1977 Bundesverdienstkreuz 1.Klasse und Ehrensenator der Uni Stuttgart 1979 Einweihung des Forschungszentrums Artur Fischer Forschung und Übergabe der Geschäftsführung an Sohn Klaus Fischer. 1991 Verleihung

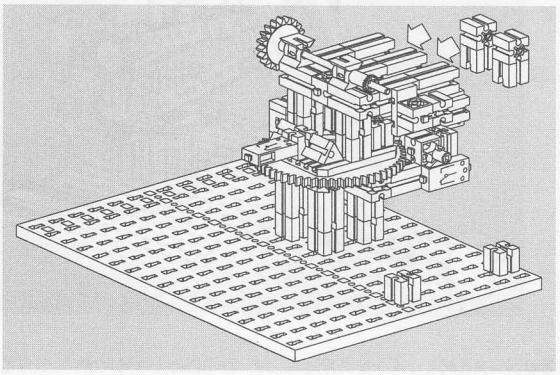
des Werner-von-Siemens-Ringes (des "deutschen Nobelpreises"). 1992 Ehrenmitglied des Deutschen Museums in München.

Zum Schluß

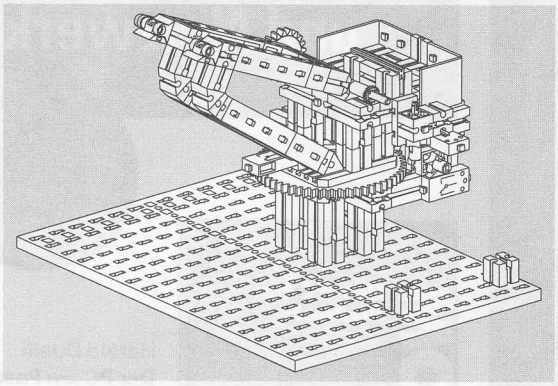
Wir haben in diesem Heft den Gesamtkomplex der Automatisierungs- und Robotertechnik behandelt. Es war ein weiter Weg vom ersten Weihwasserspender der alten Griechen bis zu unserem heutigen Stand der Forschung und Technik. Mancher fragt sich, ob diese Entwicklung zum Wohle des einzelnen Menschen, ob sie zum Wohle der Menschheit beigetragen hat.

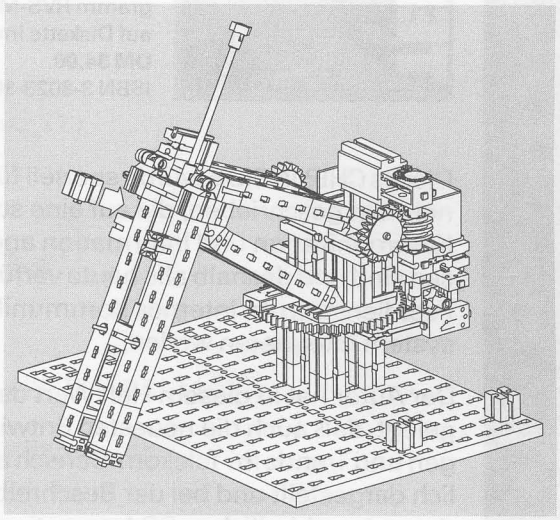


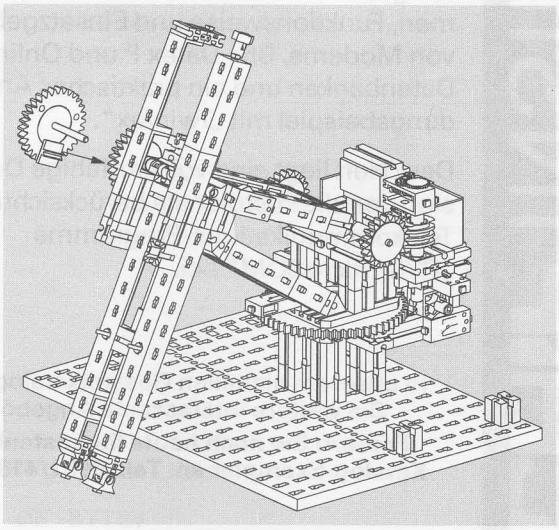




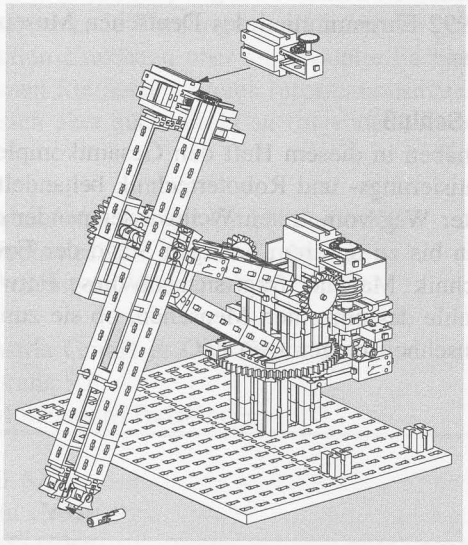
Modell Roboter, Baustufe 1 bis 3

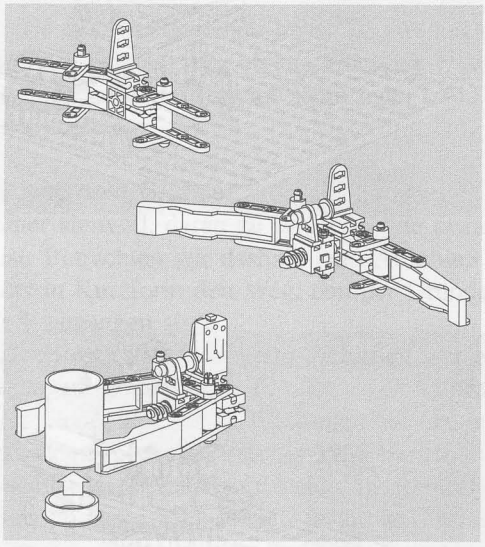


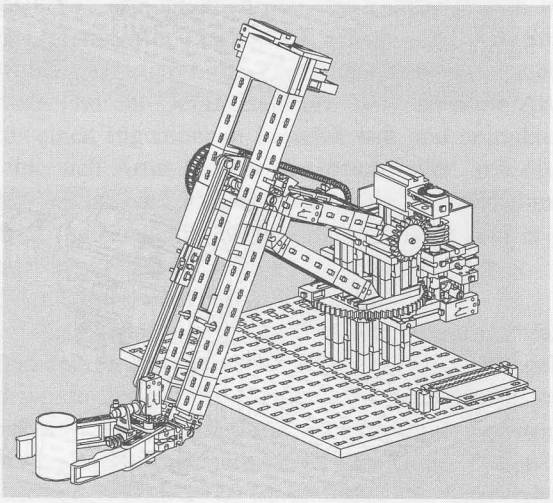




Modell Roboter, Baustufe 4 bis 6







Modell Roboter, Baustufe 7 bis 9

```
PROGRAM Reaktion; { Turbo Pascal 6.0 }

USES  CRT,
      DOS,
      Fischer;

VAR Stunde_1,Minute_1,Sekunde_1,HndstSek_1,
    Stunde_2,Minute_2,Sekunde_2,HndstSek_2 : WORD;
    Zufallszeit                            : INTEGER;
    Inp_Outp_Nr                            : BYTE;

{------------------------------------------------------------------------}

BEGIN { Programm Reaktion }
  CLRSCR;
  RANDOMIZE;
  Zufallszeit := RANDOM(10000);
  DELAY(Zufallszeit);
  Inp_Outp_Nr := RANDOM(3);
  IF Inp_Outp_Nr = 0 THEN Inp_Outp_Nr := 1;
  GETTIME(Stunde_1,Minute_1,Sekunde_1,HndstSek_1);
  REPEAT
    Motor(Inp_Outp_Nr,Links);
  UNTIL Taster(Inp_Outp_Nr);
  GETTIME(Stunde_2,Minute_2,Sekunde_2,HndstSek_2);
  Motor(Inp_Outp_Nr,Aus);
  GOTOXY(1,1);
  WRITELN('Startzeit:
',Stunde_1:2,':',Minute_1:2,':',Sekunde_1,':',HndstSek_1:2);
  WRITELN('Endezeit :
',Stunde_2:2,':',Minute_2:2,':',Sekunde_2,':',HndstSek_2:2);
END.  { Programm Reaktion }
```

Reaktionstester

```
PROGRAM Melodiespiel; { Turbo Pascal 6.0 }

USES  CRT,
      DOS,
      Fischer;

CONST Max_Zufallszahlen = 30;

TYPE  t_Melodie = ARRAY[1..Max_Zufallszahlen] OF BYTE;
```

```
VAR   Melodie : t_Melodie;
      Fehler  : BOOLEAN;
{-------------------------------------------------------------------------------}

PROCEDURE Zufallsmelodie_erzeugen;
VAR i : BYTE;
    Zufallszahl : BYTE;
  BEGIN { Zufallsmelodie_erzeugen }
    RANDOMIZE;
    FOR i := 1 TO Max_Zufallszahlen DO
      BEGIN
        REPEAT
          Zufallszahl := RANDOM(5);
        UNTIL Zufallszahl <> 0;
        Melodie[i] := Zufallszahl;
      END;
  END;  { Zufallsmelodie_erzeugen }
{-------------------------------------------------------------------------------}

PROCEDURE Spiele_Tonreihe_bis_Ton(i:BYTE);
VAR j,k : INTEGER;
  BEGIN { Spiele_Tonreihe_bis_Ton }
    DELAY(1000);
    FOR j := 1 TO i DO
      BEGIN
        SOUND(Melodie[j]*1000);
        FOR k := 1 TO 6000 DO Motor(Melodie[j],Rechts);
        Motor(Melodie[j],Aus);
        NOSOUND;
        DELAY(100);
      END;
  END;  { Spiele_Tonreihe_bis_Ton }
{-------------------------------------------------------------------------------}

PROCEDURE Tonreihe_abfragen(i:BYTE);
VAR j,TasterNr : BYTE;
  BEGIN { Tonreihe_abfragen }
    j := 0;
    REPEAT
      INC(j);
      TasterNr := 0;
      REPEAT
        IF Taster(1) THEN TasterNr := 1;
        IF Taster(2) THEN TasterNr := 2;
        IF Taster(3) THEN TasterNr := 3;
        IF Taster(4) THEN TasterNr := 4;
      UNTIL TasterNr <> 0;
      Motor(TasterNr,Rechts);
      SOUND(TasterNr * 1000);
      REPEAT UNTIL NOT Taster(TasterNr);
      NOSOUND;
      Motor(TasterNr,Aus);
      Fehler := TasterNr <> Melodie[j];
   UNTIL Fehler OR (i=j);
```

```
  END;  { Tonreihe_abfragen }
{------------------------------------------------------------------------------}

PROCEDURE Zufallsmelodie_vorspielen_und_abfragen;
VAR i,j : BYTE;
  BEGIN { Zufallsmelodie_vorspielen_und_abfragen }
    i := 0;
    REPEAT
      INC(i);
      Spiele_Tonreihe_bis_Ton(i);
      Tonreihe_abfragen(i);
      CLRSCR;
      IF NOT Fehler THEN WRITE('Sehr gut !')
                    ELSE WRITE('Fehler !');
    UNTIL Fehler OR (i = Max_Zufallszahlen);
  END;  { Zufallsmelodie_vorspielen_und_abfragen }
{------------------------------------------------------------------------------}

BEGIN { Programm Melodie }
  CLRSCR;
  NOSOUND;
  Zufallsmelodie_erzeugen;
  Zufallsmelodie_vorspielen_und_abfragen;
  NOSOUND;
END.  { Programm Melodie }
```

Melodiespiel

```
PROGRAM READCD; { Turbo Pascal 6.0 }

USES CRT,
     FISCHER;

CONST Max_Felder = 81;

TYPE t_Datenfeld = ARRAY[1..Max_Felder] OF BOOLEAN;

VAR i         : INTEGER;
    Datenfeld : t_Datenfeld;
{}

PROCEDURE CD_Player_initialisieren;
VAR i : BYTE;
```

```
  BEGIN { CD_Player_initialisieren }
    FOR i := 1 TO Max_Felder DO Datenfeld[i] := FALSE;
    Motor(3,Rechts);
    Motor(1,Rechts);
    Motor(2,Links);
    REPEAT UNTIL NOT Taster(4);
    Motor(2,Rechts);
    REPEAT UNTIL Taster(4);
    Motor(2,Aus);
    REPEAT UNTIL NOT Taster(1);
    REPEAT UNTIL Taster(1);
  END;  { CD_Player_initialisieren }
{}

PROCEDURE Scheibe_einlesen;
VAR i,C,Y : BYTE;
  BEGIN { Scheibe_einlesen }
    FOR i := 1 TO 25 DO
      BEGIN
        Datenfeld[i] := NOT Taster(5);
        C := 0;
        REPEAT
          REPEAT UNTIL     Taster(2); DELAY(2);
          REPEAT UNTIL NOT Taster(2); DELAY(2);
          INC(C);
        UNTIL C = 8;
      END;
    Motor(1,Aus);

    Y := 0;
    Motor(2,Links);
    REPEAT
      REPEAT UNTIL     Taster(3);
      REPEAT UNTIL NOT Taster(3);
      INC(Y);
    UNTIL Y = 29;
    Motor(2,Aus);

    Motor(1,Rechts);
    FOR i := 26 TO 50 DO
      BEGIN
        Datenfeld[i] := NOT Taster(5);
        C := 0;
        REPEAT
          REPEAT UNTIL     Taster(2); DELAY(2);
          REPEAT UNTIL NOT Taster(2); DELAY(2);
          INC(C);
        UNTIL C = 8;
      END;
    Motor(1,Aus);

    Y := 0;
    Motor(2,Links);
    REPEAT
```

```
      REPEAT UNTIL     Taster(3);
       REPEAT UNTIL NOT Taster(3);
       INC(Y);
     UNTIL Y = 29;
     Motor(2,Aus);
    Motor(1,Rechts);
     FOR i := 51 TO 75 DO
       BEGIN
         Datenfeld[i] := NOT Taster(5);
         C := 0;
         REPEAT
           REPEAT UNTIL     Taster(2); DELAY(2);
           REPEAT UNTIL NOT Taster(2); DELAY(2);
           INC(C);
         UNTIL C = 8;
       END;
     Motor(1,Aus);
     Motor(3,Aus);
     Motor(2,Rechts);
     REPEAT UNTIL Taster(4);
     Motor(2,Aus);
  END;  { Scheibe_einlesen }
{}

PROCEDURE Tonfolge_erzeugen;
VAR i,
    Frequenz : INTEGER;
  BEGIN { Tonfolge_erzeugen }
    i := 1;
    REPEAT
      Frequenz := 0;
      IF Datenfeld[i  ] THEN Frequenz := Frequenz + 2*2*2;
      IF Datenfeld[i+1] THEN Frequenz := Frequenz +   2*2;
      IF Datenfeld[i+2] THEN Frequenz := Frequenz +     2;
      IF Datenfeld[i+3] THEN Frequenz := Frequenz +     1;
      SOUND(Frequenz * 100);
      DELAY(600);
      NOSOUND;
      INC(i,4);
    UNTIL i > 72;
  END;  { Tonfolge_erzeugen }
{}

BEGIN { Programm READCD }
  CLRSCR;
  CD_Player_initialisieren;
  Scheibe_einlesen;
  Tonfolge_erzeugen;
END.  { Programm READCD }
```

Das Programm zum CD-Player

```
PROGRAM Kennlinienschreiber; { Turbo Pascal 6.0 }

USES  CRT,
      DOS,
      Fischer;

CONST Minimaler_Analogwert_EX =  70;
      Maximaler_Analogwert_EX = 450;
      Analoger_Messbereich    = Maximaler_Analogwert_EX -


Minimaler_Analogwert_EX;
      Maximale_X_Schritte     = 360;
      Faktor                  = Analoger_Messbereich DIV Maximale_X_Schritte;

VAR   X,
      Zielposition_X,
      Analogwert : INTEGER;
      Stunde,Minute,Sekunde,Hndst_Sekunde,
      Sekundenmerker                    : WORD;
{}

PROCEDURE Kennlinienschreiber_initialisieren;
  BEGIN { Kennlinienschreiber_initialisieren }
    WHILE NOT Taster(4) DO Motor(1,Rechts);
    Motor(1,Aus);
    WHILE Taster(2) DO Motor(2,Rechts);
    Motor(2,Aus);
    X                := 0;
    Sekundenmerker := 0;
  END;  { Kennlinienschreiber_initialisieren }
{}

PROCEDURE Zeilenvorschub;
  BEGIN { Zeilenvorschub }
    GETTIME(Stunde,Minute,Sekunde,Hndst_Sekunde);
    IF Sekunde <> Sekundenmerker
      THEN
        BEGIN
          Motor(3,Rechts);
          REPEAT UNTIL Taster(3);
          REPEAT UNTIL NOT Taster(3);
          Motor(3,Aus);
          Sekundenmerker := Sekunde;
        END;

  END;  { Zeilenvorschub }
{}

PROCEDURE Analogeingang_einlesen_und_verarbeiten;
  BEGIN { Analogeingang_einlesen_und_verarbeiten }
    Analogwert := Analog_Eingang(X_Poti);
```

```
    Zielposition_X := (Analogwert - Minimaler_Analogwert_EX) DIV Faktor;
     IF Zielposition_X <> X
       THEN
         BEGIN
           Motor(1,Links);
           REPEAT UNTIL Taster(1);
          REPEAT UNTIL NOT Taster(1);
           INC(X);
         END;
     IF Zielposition_X <> X
       THEN
         BEGIN
           Motor(1,Rechts);
           REPEAT UNTIL Taster(1);
           REPEAT UNTIL NOT Taster(1);
           DEC(X);
         END;
     Motor(1,Aus);
     GOTOXY(1,1);
     WRITELN('X: ',X:3,'  Ziel_X: ',Zielposition_X:3,' Messwert: ',Analogwert:3);
  END;  { Analogeingang_einlesen_und_verarbeiten }
{}

BEGIN { PROGRAM Kennlinienschreiber }
  CLRSCR;
  Kennlinienschreiber_initialisieren;
  REPEAT
    Analogeingang_einlesen_und_verarbeiten;
    Zeilenvorschub;
  UNTIL KEYPRESSED;
  WHILE NOT Taster(2) DO Motor(2,Rechts);
  Motor(2,Aus);
END.  { PROGRAM Kennlinienschreiber }
```

Das Programm zum Kurvenschreiber

```
PROGRAM Plotter; { Turbo Pascal 6.0 }

USES CRT,
     Fischer;

VAR  X_Pos,
     Y_Pos,
     PC_X,
     PC_Y       : INTEGER;
     X_Richtung,
     Y_Richtung : Motor_Richtung_Typ;
{}
PROCEDURE Stift_nach_Unten;
 BEGIN { Stift_nach_Unten }
    Motor(3,Links);
    REPEAT UNTIL NOT Taster(4);
```

```
    Motor(3 ,Aus);
  END; { Stift_nach_Unten }
{}
PROCEDURE Stift_nach_Oben;
  BEGIN { Stift_nach_Oben }

    Motor(3,Rechts);
     REPEAT UNTIL Taster(4);
     Motor(3 ,Aus);
  END;  { Stift_nach_Oben }
{}

PROCEDURE Plotter_initialisieren;
  BEGIN { Plotter_initialisieren }
    Motor(4,Links);
    Stift_nach_Oben;
    Motor(1,Rechts);
    REPEAT UNTIL Taster(3);
    Motor(1,Aus);
    Motor(2,Rechts);
    REPEAT UNTIL Taster(2);
    Motor(2,Aus);

    DELAY(200);
    X_Pos := 0;
    Y_Pos := 0;
    PC_X  := 1;
    PC_Y  := 1;
  END;  { Plotter_initialisieren }
{}

PROCEDURE Fahre_nach_XY(X,Y:INTEGER);

  PROCEDURE X_Position_anfahren;
    BEGIN { X_Position_anfahren }
      CASE PC_X OF
        1 : BEGIN
              IF X > X_Pos THEN BEGIN X_Richtung := Links; Motor(2,Links); END
                           ELSE BEGIN X_Richtung := Rechts; Motor(2,Rechts); END;
              INC(PC_X);
            END;
        2 : IF NOT Taster(5) THEN INC(PC_X);
        3 : IF Taster(5)
              THEN
                BEGIN
                  IF X_Richtung = Links THEN INC(X_Pos)
                                        ELSE DEC(X_Pos);
                  IF X = X_Pos
                    THEN
                      BEGIN
                        Motor(2,Aus);
                        PC_X := 1;
                      END
```

```
                    ELSE PC_X := 2;
                END;
      END; { CASE PC_X OF }
    END;  { X_Position_anfahren }
  {}
  PROCEDURE Y_Position_anfahren;
    BEGIN { Y_Position_anfahren }
      CASE PC_Y OF
        1 : BEGIN
              IF Y > Y_Pos THEN BEGIN Y_Richtung := Links; Motor(1,Links); END
                           ELSE BEGIN Y_Richtung := Rechts; Motor(1,Rechts); END;
              INC(PC_Y);
            END;
        2 : IF NOT Taster(1) THEN INC(PC_Y);
        3 : IF Taster(1)
              THEN
                BEGIN
                  IF Y_Richtung = Links THEN INC(Y_Pos)
                                        ELSE DEC(Y_Pos);
                  IF Y = Y_Pos
                    THEN
                      BEGIN
                        Motor(1,Aus);
                        PC_Y := 1;
                      END
                    ELSE PC_Y := 2;
                END;
      END; { CASE PC_Y OF }
    END;  { Y_Position_anfahren }
  {}

  BEGIN { Fahre_nach_XY }
    REPEAT
      IF X <> X_Pos THEN X_Position_anfahren;
      IF Y <> Y_Pos THEN Y_Position_anfahren;
    UNTIL (X=X_Pos) AND (Y=Y_Pos);
  END;  { Fahre_nach_XY }
{}

BEGIN { PROGRAM Plotter }
  CLRSCR;
  Plotter_initialisieren;

  Fahre_nach_XY(10,10);
  Stift_nach_Unten;
  Fahre_nach_XY(15,10);
  Fahre_nach_XY(15,100);
  Fahre_nach_XY( 5,100);
  Fahre_nach_XY( 5, 10);

  Plotter_initialisieren;
END.  { PROGRAM Plotter }
```

Das Programm zum Plotter

```
DIM Bit(7), lpt(4)
DEF SEG = &H40

lpt(1) = PEEK(&H9) * 256 + PEEK(&H8)
lpt(2) = PEEK(&HB) * 256 + PEEK(&HA)
lpt(3) = PEEK(&HD) * 256 + PEEK(&HC)
lpt(4) = PEEK(&HF) * 256 + PEEK(&HE)

CLS
DO

FOR m = 1 TO 4
  IF lpt(m)  0 THEN
    wert% = INP(lpt(m))
    GOSUB Zerlegen
    reg1$ = erg$
    wert% = INP(lpt(m) + 1)
    GOSUB Zerlegen
    reg2$ = erg$
    wert% = INP(lpt(m) + 2)
    GOSUB Zerlegen
    reg3$ = erg$
    LOCATE 10, 5
    PRINT lpt(m); "  "; reg1$; "    "; reg2$; "    "; reg3$
  END IF
NEXT
  ant$ = INKEY$
LOOP WHILE ant$ = ""
DEF SEG
END

Zerlegen:
 erg$ = ""
 FOR n = 7 TO 0 STEP -1
   Bit(n) = (wert% AND 2 ^ n) / 2 ^ n
   erg$ = erg$ + (STR$(Bit(n)))
 NEXT

RETURN
```

Ein Programm zur Drucker-Steuerung

```
10 DECLARE SUB JoyStick (Stck%(), Strg%())
20 DECLARE SUB JoyStickDemo (Stck%(), Strg%())
30 DECLARE SUB JoyStickAnzeige (Stck%(), Strg%())
40 DEFINT A-Z
50 DIM Stck(3), Strg(3)
60
70
80 JoyStickDemo Stck(), Strg()
90

100 SUB JoyStick (Stck(), Strg())
110   FOR m = 0 TO 3
120     Stck(m) = STICK(m)
130     Strg(m) = STRIG(0)
140   NEXT
150 END SUB

170 SUB JoyStickAnzeige (Stck(), Strg())
180   CLS
190   FOR m = 0 TO 3
200     PRINT Stck(m), Strg(m)
210   NEXT
220 END SUB

240 SUB JoyStickDemo (Stck(), Strg())
250   STATIC Merk0, Merk1
260   DO
270     JoyStick Stck(), Strg()
280       JoyStickAnzeige Stck(), Strg()
290       DO
300         ant$ = INKEY$
310       LOOP WHILE ant$ = ""
320       IF UCASE$(ant$) = "C" THEN Modus = 1
330   LOOP WHILE ant$ <> CHR$(27)
340 END SUB
```

Ein Programm zur Joystick-Steuerung

Die Programme auf der Diskette

Auf der Diskette finden Sie verschiedene unter DOS oder Windows lauffähige Programme.

Die Programme sind aus Platzgründen komprimiert und müssen entpackt werden. Sie können das Entpacken auf zweierlei Art vornehmen.

Nutzen Sie INSTALL.BAT auf der Diskette

Wenn Sie dieses Programm aufrufen, können Sie einen vorhandenen Pfad (Laufwerk und/oder Verzeichnis) zum Abspeichern vorgeben, indem Sie den Pfad als Argument beim Programmaufruf vorgeben. Dies könnte zum Beispiel so aussehen:

A:INSTALL C:Neu\

Das Programm erstellt im vorgegebenen Pfad oder, wenn keiner vorgegeben ist, im aktuellen Pfad verschiedene Unterverzeichnisse unter einem gemeinsamen Verzeichnis \CHIP1292 .

Entpacken in einzelne, eigene Verzeichnisse

Erstellen Sie sich dazu je ein entsprechendes Verzeichnis auf Ihrer Festplatte, gehen Sie in dieses Verzeichnis und rufen Sie das gewünschte Programmpaket auf der Diskette mit dem Namen auf. Es wird dann automatisch in das Verzeichnis entpackt, von dem aus der Aufruf erfolgt ist. Sie können das Entpacken auch auf Disketten vornehmen. Diese sollten aber zur Sicherheit mindestens 720 K bei 3,5"-Disketten und 1,2 M bei 5 1/4"-Disketten Speicherkapazität haben. Nehmen Sie für jedes zu entpackende Programmpaket eine (formatierte) Diskette. Wir haben bewußt darauf verzichtet, alle Programme in eine Datei zu komprimieren.

DOSPROG

In dieser Datei sind die folgenden direkt unter DOS startbaren Programme komprimiert:

Das Startprogramm

Dieses Programm bietet die Möglichkeit, alle unter DOS laufenden Programme über ein Menü zu starten. Das Programm prüft jeweils, ob das zu startende Programm sich im aktuellen Verzeichnus befindet. Ist dies nicht der Fall, kann man auf der DOS-Ebene in das entsprechende Verzeichnis gehen und dann erneut den betreffenden Menüpunkt aufrufen.

Das Programm prüft auch, beispielsweise vor dem Start des Lucky-Logic-Demoprogramms, ob eine Maus aktiviert ist, und weist darauf hin.

Das Pseudocode-Übersetzungsprogramm

Das Übersetzungsprogramm arbeitet mit Bibliotheksmodulen für die Programmiersprache, in die es übersetzen soll.

In Form von DATA-Zeilen ins Programm eingebunden ist ein BASIC-Befehlssatz, der beim Start in das Array Befehle$() eingelesen wird.

Man kann aber nach Programmstart jederzeit auch eine eigene Befehlsatzdatei in dieses Befehlsfeld einlesen. Die Befehlsdatei muß nur folgende Bedingungen erfüllen:

1. Die Datei muß die Befehle in ASCII-Text enthalten. Das ist ein Zeichensatz, der auf jedem DOS-Rechner verfügbar ist.

2. Die erste Zeile darf nur aus dem Kennwort HTRANS bestehen. Diese Kennung wird dazu benutzt, Befehlsbibliotheken eindeutig zu identifizieren. Die Schreibweise ist nicht vorgegeben. Es können Klein- oder/und Großbuchstaben beliebig verwendet werden.

3. Die zweite Zeile muß den Namen der Programmiersprache enthalten. Die Schreibweise ist nicht vorgegeben. Dieser Zeileninhalt wird im Programm als Programmname verwendet.

4. Die Programmbefehle werden nach folgendem Schema eingelesen und sollten so auch in die Datei geschrieben werden:
Erst kommt der Pseudocodebefehl, dann durch ein Komma getrennt, der Befehl in der vorgesehenen Programmiersprache.

5. Die Zahl der Befehle ist nicht festgelegt. Das Programm erkennt selbständig die Anzahl und dimensioniert die Arrays entsprechend.

Sie finden in der mitgelieferten Datei BASIC.LNG einen Vorschlag für die BASIC-Befehle. Sie können diese Datei problemlos erweitern oder reduzieren.

Die Schnittstellenprüfprogramme

Zwei Programme dienen dazu, zu ermitteln, welche Schnittstellen vorhanden und welche aktiv sind.

Sie können diese Programme in Batch-Programme einfügen und so vor dem Start eines anderen Programms prüfen, ob die betreffende Schnittstelle überhaupt aktiv ist.


LEITZ 6271

In dieser Ausgabe

Autor: Horst F. Haupt

Robotik
Die Welt der Roboter

Computing
Schlüssel zur Computertechnik
Wie man Computer programmiert

Messen – Steuern – Regeln
Signale in Informationen umsetzen

Fuzzy-Logic
Fuzzy Logic – eine faszinierende Logic
Die Demoversion von fuzzyTECH

Umsetzung der Theorie
Ein Roboter wird konstruiert

Die fischertechnik-Konstruktionsbaukästen
Profi Sensoric und Profi Computing

ISB N 3-8023-1258-9 DM +034.00

DM 34,–
SFR 34,–
ÖS 275,–
1. Auflage 1992

VOGEL

9 783802 312588 03400