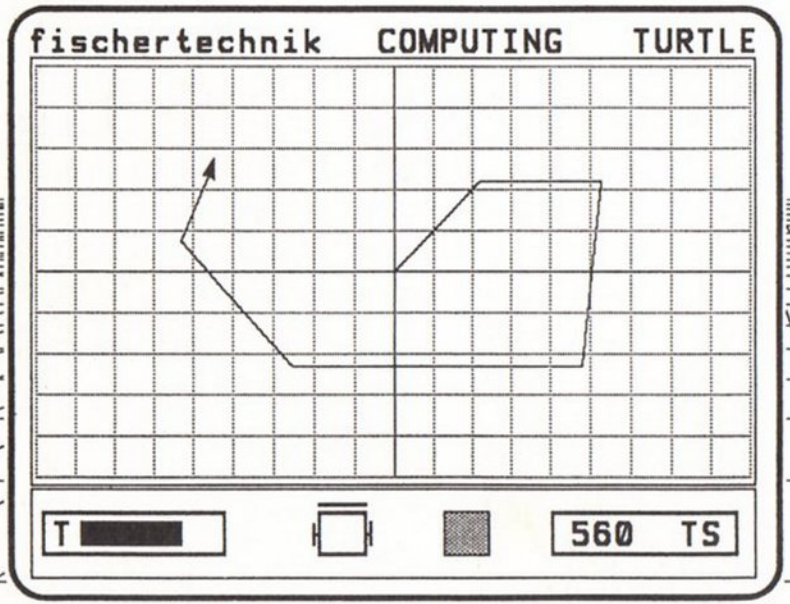


Experimentierhandbuch IBM®-PC und Kompatible

fischertechnik COMPUTING

Lieber fischertechnik-Freund,

Sie kennen die Berichte aus den Zeitungen, Sie sehen die Sendungen im Fernsehen: Die Rede ist von Computern, von Robotern, von Automation und von der Fabrik der Zukunft. Nahtlos fügt sich ein Arbeitsgang an den anderen: Schweißen, Schrauben, Montieren, Lackieren.
Und dies ist schon das Ende eines Prozesses, der im Büro beginnt. Der Konstrukteur füttert die Maschine mit Maßangaben, die der Computer flugs in Darstellungen auf dem Bildschirm umsetzt - aber auch in Detailzeichnungen und Datenströme. On Line, auf direktem Wege, ist der Computer mit einem Kollegen in der Fabrikhalle verbunden, der aus den Daten die Bewegungen und Tätigkeiten eines Roboters errechnet. Und dieser arbeitet dann sein Programm ab - Stunde um Stunde, Tag um Tag.
Als K. Capek im Jahre 1921 das Wort Roboter erfand, da stellte er sich darunter einen künstlichen Menschen, eine Puppe, vor, die Bewegungen scheinbar selbständig ausführt und Funktionen, die der Mensch wahrnimmt, teilweise übernehmen kann. Jahrzehntelang war das menschenähnliche Aussehen ein besonderes Merkmal von Robotern. Hunderte von Science-Fiction-Stories und -Filmen zeugen davon. Die tatsächlichen Roboter haben mit diesen Gebilden wenig gemein, und sie sind auch längst nicht so intelligent.
Moderne Roboter sind Schwerstarbeiter. Sie bauen Autos und transportieren Lasten. Aber denken wie ein Mensch können sie (glücklicherweise) nicht. Und gar Gefühle zeigen, mit Phantasie an ein Problem herangehen, das kann ein Computer oder Roboter schon garnicht. Ein Computer kann nur das, was ihm mit einem Programm gesagt wird. Das Programm müssen wir (mit Phantasie und Kreativität!) entwickeln.
Dieser Experimentierkasten demonstriert praktisch alle Möglichkeiten von Computersteuerungen im Kleinen. Wenn Sie sich immer an die Anleitung halten, werden Sie sehr schnell mit der Programmierung vertraut werden. Und dann geht es schon zu den ersten Versuchen. Wir geben Ihnen aber auch eine Menge weiterer Anregungen zum Experimentieren. Versuchen Sie es einmal, Sie werden viel Spaß haben.

Ihre
Artur und Klaus Fischer

Anmerkung:

Dieses Anleitungsbuch beschreibt die Verwendung des fischertechnik COMPUTING EXPERIMENTAL Baukastens mit einem IBM-Computer oder einem Kompatiblen. Bitte vergewissern Sie sich, daß Ihr Computer ein IBM PC oder 100% kompatibles Gerät ist. Der überwiegende Teil der Software benutzt die hochauflösende Bildschirmausgabe nach dem CGA-Standard (oder kompatiblen Standards wie EGA oder VGA) oder den Monochrom-Standard.

Der fischertechnik COMPUTING EXPERIMENTAL Baukasten kann mit anderer Software und anderem Handbuch auch an den Computern C64/128, Amiga 500 und 2000, Atari ST und Schneider/Amstrad CPC betrieben werden.

Es wurden alle erdenklichenMaßnahmen getroffen, um die Richtigkeit dieser Produkt-Dokumentation zu gewährleisten. Da jedoch die fischerwerke Artur Fischer GmbH&Co.KG ständig an der Verbesserung ihrer Produte arbeiten, können wir keine Garantie für die Vollständigkeit und Richtigkeit dieser Dokumentation seit ihrem Erscheinen übernehmen.

Inhalt

Bevor Sie gleich mit dem schönsten Modell anfangen - lesen Sie bitte weiter. Wir wollen Ihnen hier noch einige Tips mitgeben.

Zunächst möchten wir Ihnen einen Überblick über Ihre COMPUTING EXPERIMENTAL Ausrüstung verschaffen.
Sie haben nun drei Anleitungsbücher in der Hand, von denen eines das vorliegende ist. Dieses Experimentierhandbuch dient als Leitfaden durch alle Experimente und enthält die Programme, Erläuterungen und Vorschläge. Aus drucktechnischen Gründen ist der Aufbau der fischertechnik Modelle in einem getrennten Anleitungsbuch dargestellt, der fischertechnik COMPUTING EXPERIMENTAL Bauanleitung. Das dritte Anleitungsheft ist die fischertechnik Interface-Anleitung. Dieses Anleitungsbuch werden Sie normalerweise nicht benötigen, denn die Benutzung des fischertechnik Interface im Rahmen des fischertechnik COMPUTING EXPERIMENTAL wird ausführlich in dem vorliegenden Experimentierhandbuch beschrieben. Die fischertechnik Interface-Anleitung beschreibt dagegen die Benutzung des Interface im Rahmen eines anderen Softwaremodells. Die Interface-Anleitung werden Sie nur benötigen, wenn Sie sich über die Details der Arbeitsweise des Interface informieren wollen oder noch andere fischertechnik COMPUTING Baukästen erwerben wollen. Legen Sie daher die Interface-Anleitung im Moment zur Seite. Bewahren Sie sie aber gut auf, denn Sie könnten sie später benötigen.
Dann ist da die Hardware des Baukastens, insbesondere die vielen fischertechnik-Bausteine. Wenn Sie die Teile auf Vollständigkeit prüfen wollen, sollten Sie die Teileliste in der fischertechnik COMPUTING EXPERIMENTAL Bauanleitung zu Rate ziehen. Die Teileliste zeigt für jeden Baustein ein Foto, die Teilenummer und Bezeichnung (wichtig für Nachbestellungen) sowie dessen Anzahl im Baukasten.
Der große Kasten mit dem Klarsichtdeckel ist das fischertechnik Interface. Es verbindet das Modell mit dem Computer. Es wird außerdem noch mit dem Steckernetzgerät COMPUTING EXPERIMENTAL verbunden. So leitet das Interface unter Anweisung der Software und des Computers den elektrischen Strom zu dem fischertechnik Modell. Mehr darüber in Kapitel 4.
Das Softwarepaket, das Sie zugeschickt erhielten, enthält noch eine 5¼"-Diskette im 360K-Format. Auf Anforderung erhalten Sie die Software auch auf einer 3½"-Diskette

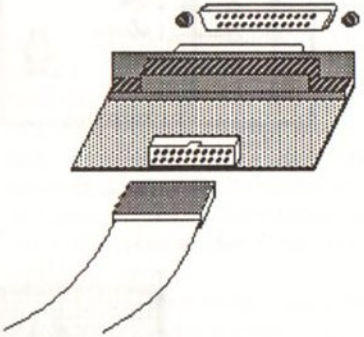


Bild 2.1: Das Interface wird mit Hilfe des Adapters an die Druckerschnittstelle angeschlossen

im 720K-Format. Auf die Software kommen wir im Kapitel 3 zu sprechen.

Das Softwarepaket beinhaltet außerdem noch ein kleines, aber sehr wichtiges Stück Hardware. Genau passend zu Ihrem Computer erhalten Sie einen Adapter für das Interface. Er besteht aus einem Stück Leiterplatte mit zwei Steckverbindern. Ein Steckverbinder besteht aus zwanzig Stiften mit einem Gehäuse darum. Entnehmen Sie das fischertechnik Interface dem Baukasten. An ihm ist ein Anschlußkabel befestigt und daran wieder ein Stecker. Dieser Stekker paßt genau auf die zwanzig Stifte. Eine Aussparung im Gehäuse und eine Nase am Stecker gewährleisten, daß Sie beides richtig herum zusammenstecken. Der andere Stecker des Adapters paßt zu der Druckerschnittstelle Ihres Computers. Die Druckerschnittstelle ist eine 25-polige trapezförmige Buchse. Die V24-Schnittstelle hat zwar auch 25 Pole, weist jedoch Stifte und keine Buchsen auf.

Der PC-Standard sieht bis zu drei Druckerschnittstellen an einem Computer vor (normalerweise als LPT1:, LPT2: und LPT3: bezeichnet). Wenn Ihr Computer mit mehreren Druckerschnittstellen ausgestattet ist, haben Sie freie Auswahl, an welche Druckerschnittstelle Sie das fischertechnik COMPUTING Interface anschließen. Mehrere Druckerschnittstellen sind recht nützlich, weil Sie dann den Drucker nicht abkabeln müssen und gleichzeitig installiert lassen können.

Wenn Ihr Computer nur eine Druckerschnittstelle besitzt, können Sie auch einen Druckerumschalter benutzen, um zwischen dem Drucker und dem fischertechnik COMPUTING Interface hin- und her zu schalten.

Manche PC sind mit anderen Druckerschnittstellen als den zuvor beschriebenen 25-poligen D-Steckbuchsen ausgestattet. Als Ersatzteil können Adapter für 34-polige Leiterplattenverbinder und 36-polige Centronics-Stecker geliefert werden. Der 36-polige Centronics-Stecker wird auch bei den meisten Druckerumschaltern notwendig sein.

Schließen Sie den Adapter an die Druckerschnittstelle an.

Wichtig: Der Computer muß dabei ausgeschaltet sein!

Da die Standard-Druckerschnittstelle verpolungssicher ist, kann der Adapter nicht falsch aufgesteckt werden. Wenn der Adapter nicht zu passen scheint, prüfen Sie daher noch einmal seine Ausrichtung.

Entnehmen Sie nun auch das Steckernetzgerät dem Baukasten. Das Anschlußkabel trägt einen roten und einen grünen Stecker. Der rote Stecker kommt in eine der beiden

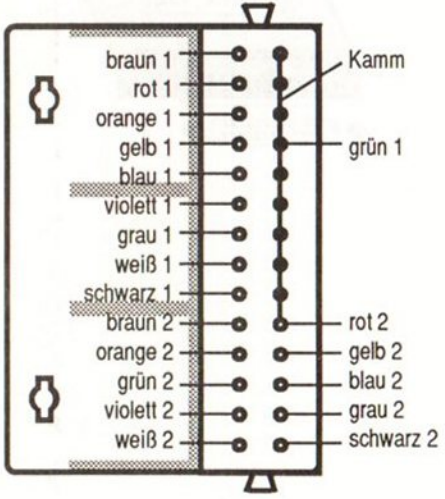


Bild 2.2: Anschluß des Interfacekabels an die 28-polige Steckbuchse. Beachten Sie, daß die Adern grün 1 und rot 2 zusammen mit dem Kamm montiert werden.

Buchsen des Interface, die mit + bezeichnet sind. Die grüne kommt in eine der beiden Buchsen mit dem - Zeichen. Welche Buchse sie jeweils verwenden, ist gleichgültig. Die doppelte Anschlußmöglichkeit ist für größere fischertechnik COMPUTING Modelle vorgesehen, die mehr Strom benötigen.

Nun müssen Sie am Interface noch das Modell anschließen. Im Baukasten finden Sie dazu ein zwanzigpoliges Kabel von 2 Meter Länge, dessen einzelne Adern verschiedenfarbig sind. Am einen Ende ist ein Stecker angebracht, der am Interface eingesteckt werden kann.

Halt - noch nicht einstecken! Zuerst wollen wir das andere Ende des Kabels herrichten. Im Baukasten liegt eine 28-polige Steckbuchse (s. Bild 2.2 und fischertechnik COMPUTING EXPERIMENTAL Bauanleitung), in die fischertechnik Stecker hineinpassen. Dabei ist auch ein metallischer Kamm, der dazu dient, mehrere Buchsen untereinander zu verbinden. Schrauben Sie die zwanzig Adern des Flachbandkabels zusammen mit dem Kamm an die Buchsen an. Studieren Sie dazu Bild 2.2, das genau angibt, welche Ader an welche Buchse kommt. Benutzen Sie die Kabelfarben zur Orientierung: die Adern tragen die Farben Braun, Rot, Orange, Gelb, Grün, Blau, Violett, Grau, Weiß, Schwarz und dann das gleiche noch einmal. Achten Sie beim Anschrauben darauf, daß Sie die Schrauben nicht zu fest anziehen, so daß das Kabel abgequetscht würde. Zu den Buchsen, die den Kamm aufnehmen, kommen eine grüne und eine rote Leitung des Flachbandkabels. Es macht bei dem fischertechnik Interface nichts, daß die beiden Leitungen auf diese Weise miteinander verbunden werden, denn beiden führen +5V.

Nach Abschluß der Arbeiten führen Sie noch eine sorgfältige Sichtkontrolle durch. Auf der Buchsenoberseite ist ein Etikett angebracht, das zu jeder Buchse den Farbcode des angeschlossenen Kabels trägt. Machen Sie sich die Mühe, wirklich sorgfältig und genau zu kontrollieren, ob alles übereinstimmt. Sie sparen sich späteren Ärger oder gar eine Beschädigung des Interface. Erst wenn Sie ganz sicher sind, können Sie den Verbindungsstecker in das Interface einstecken. Vorher sollten Sie aber noch das Flachbandkabel gegen die rote Platte drücken, eine 45 mm lange Strebe darüberlegen und diese mit zwei S-Riegel befestigen. Derart gesichert, ist die Verbindung von Kabel und Buchse vor Zugbelastung geschützt und kann nicht so leicht beschädigt werden.

Die Modelle und wie sie Stück für Stück aufgebaut werden, finden Sie in der Bauan-

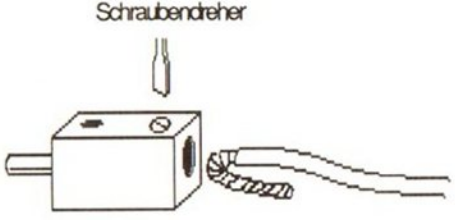


Bild 2.3: Montage eines fischertechnik Steckers

Übrigens:

Manchmal gibt es immer noch recht nützliche Zusatzinformationen. Z.B. eine Übersicht über Kommandos, eine Begriffserklärung, die Wirkungsweise eines Sensors usw. Sie können den Haupttext ruhig weiterlesen, aber vielleicht studieren Sie auch mal diese Hinweise. Sie stehen immer hier am linken Rande der Seite. Hier kommt gleich der erste:
Wenn Sie wissen wollen, welche Modelle zu welchen Textabschnitten gehören, vergleichen Sie einmal die Symbole in der linken oberen Ecke in beiden Heften! Richtig - gleiche Symbole kennzeichnen immer Modelle und Texte zum gleichen Thema.

leitung. Bei jedem Bauabschnitt sind in einem Kasten die Bauteile angezeigt, die in dem betreffenden Abschnitt hinzukommen. Ein Tip: Suchen Sie sich zu jedem Bauabschnitt zuerst die benötigten Teile zusammen, und bauen Sie sie anschliessend erst ein. Gehen Sie erst dann zu dem nächsten Bauabschnitt über, wenn alle Teile aufgebraucht sind. Sind noch welche übrig, studieren Sie noch einmal genau die Fotos; irgendwo müssen Sie zu sehen sein. Achten Sie bei den Bausteinen ggf. auf die Orientierung, damit Ihnen in späteren Bauabschnitten nicht der Weg verbaut ist.
Alle Modelle enthalten irgendwelche elektrischen Bauelemente: Schalter, Motoren, Sensoren. Diese werden mit der zuvor hergerichteten 28-poligen Steckbuchse verbunden. Dafür stehen eine Reihe von zweiadrigen Kabeln und fischertechnik Steckern zur Verfügung. Die benötigten Kabellängen (6 cm, 18 cm oder 44 cm) gibt Ihnen die Bauanleitung an. Die Steckerfarben wählen Sie am besten so, daß sie den Farbkennzeichnungen des Flachbandkabels und der Steckerbuchse entsprechen. Das erleichtert Ihnen die Kontrolle der Verkabelung bei größeren Modellen. Versehen Sie die Kabel mit fischertechnik Steckern.
Ziehen Sie zur Steckermontage ggf. die Isolation am Kabelende ab, verdrillen Sie ein wenig die Litzen und biegen Sie die Litzen auf die Isolation um (s. Bild 2.3). Schieben Sie das Kabelende dann so in den Steckeranschluß, daß das Schräubchen auf die Isolation drückt, wenn es angezogen wird. Wiederum: nicht zu fest anziehen, das Kabel könnte abgequetscht werden.
Selbstverständlich kommen an die beiden Enden einer Ader jeweils Stecker gleicher Farbe. Die Farbmarkierung in der Kabelisolation hilft Ihnen bei der Unterscheidung der beiden Adern.
Bevor es nun mit dem Laden der Software weitergeht, noch ein paar Hinweise zur Anleitung. Blättern Sie die Anleitung ruhig jetzt schon mal durch, schmökern Sie hier und da. Wenn es dann aber an das Experimentieren geht, sollten Sie Punkt für Punkt bearbeiten. Warum? In der Anleitung werden die Programme entwickelt, ganz so wie Programme auch in Wirklichkeit entstehen. Immer wieder wird ein Experiment durchgeführt, dann die nächste Verbesserung eingebaut. Wenn Sie etwas überspringen, wissen Sie nicht, wo und was Sie einfügen sollen.
Aber nicht nur für Programme trifft dies zu: Sie werden beim Durcharbeiten des Experimentierhandbuchs eine Menge erfahren. Und wenn Sie einen Abschnitt überspringen, werden Sie vielleicht die späteren Hinweise nicht so gut verstehen und einordnen können.

Für die folgenden Versuche benötigen Sie fast immer einen oder mehrere Motoren, mit denen z.B. Seilwinde, Radarauge, Roboterarm oder der Fahrroboter bewegt werden. Sie werden über das Interface, das 20-polige, farbcodierte Kabel und die 28-polige Steckbuchse an den Computer angeschlossen. Die Stromversorgung erfolgt aus dem Netzgerät. Noch rührt sich nichts. Klar, die Software zur Ansteuerung fehlt noch.

Wenn alle Verbindungen hergestellt sind, schalten Sie erst alle Zusatzgeräte wie Bildschirm und fischertechnik COMPUTING Netzgerät ein und dann erst den Computer.

Als erstes sollten Sie sich eine Sicherungskopie der fischertechnik Diskette anfertigen. Sie fragen sich warum? Stellen Sie sich einmal vor, was Ihnen Ihr fischertechnik COMPUTING EXPERIMENTAL Baukasten noch nützen würde, wenn Sie die Diskette verlieren, beschädigen oder löschen würden! Die Software von fischertechnik ist nicht kopiergeschützt. Sie brauchen also keine ausgefeilten Kopierprogramme, sondern können das Programm DISKCOPY Ihrer DOS-Systemdiskette oder ein beliebiges anderes Kopierprogramm benutzen. Verfahren Sie so, wie es das Handbuch Ihres Computers vorschreibt. Vielleicht sollten Sie sich sogar zwei Kopien ziehen. Eine Kopie dient zum Gebrauch der Beispielprogramme auf der Diskette. Von der anderen Kopie löschen Sie die Beispielprogramme, jedoch nicht die Interface-Systemprogramme (s. nachstehende Beschreibung). Diese Diskette dient Ihnen als Arbeitsdiskette, wenn Sie die in diesem Experimentierhandbuch beschriebenen Experimente durchführen. Dies soll allerdings kein Freibrief sein; nach der geltenden Rechtsprechung sind nur Kopien zu Ihrem persönlichen Gebrauch gestattet.

Arbeiten Sie fortan nur noch mit der Kopiediskette. Verstauen Sie die Originaldiskette an einem sicheren Platz, an den keine natürlichen Feinde der Disketten, wie Sand, Hitze, Katzen oder Magnetfelder hinkommen können. Benutzen Sie die fischertechnik Originaldiskette nur, um gegebenenfalls eine weitere Kopie zu ziehen.

Einige fischertechnik Programme legen Daten auf Diskette ab. Wenn Sie schon dabei sind, sollten Sie sich auch noch mindestens eine leere Datendiskette formatieren.

Legen Sie die Kopie der fischertechnik Diskette ins Laufwerk. Wählen Sie nun das Diskettenlaufwerk A als Standardlaufwerk

des Betriebssystems. Geben Sie, wenn nötig, ein:

C>**A:**

um, wie in diesem Beispiel, von der Festplatte auf das Diskettenlaufwerk A umzuschalten. Die fettgedruckten Schriftzeichen stellen Ihre Eingabe dar. Die Eingabe wird mit Tastendruck auf die Eingabetaste abgeschlossen, die mit "Return", "Enter" oder einem Winkelpfeil "↵" bezeichnet ist. Wenn Sie Fragen zu der Eingabe von Kommandos haben, so sollten Sie den betreffenden Abschnitt in Ihrem MS-DOS-Handbuch durchnehmen.
Geben Sie nun ein:

A>**FISCHER**

Das Diskettenlaufwerk rührt sich eine Weile; es erscheint eine kurze Meldung am Bildschirm, die besagt, daß der fischertechnik Interfacetreiber aktiviert wurde.
Als nächstes müssen Sie den BASIC-Interpreter Ihres Computers laden. Wenn Ihr PC nur ein Diskettenlaufwerk besitzt, entfernen Sie die fischertechnik COMPUTING EXPERIMENTAL Diskette und legen die Diskette mit dem BASIC-Interpreter in das Diskettenlaufwerk ein. Geben Sie dann ein:

A>**BASICA**

Wenn BASIC geladen ist, entfernen Sie die Diskette wieder und legen wieder die fischertechnik COMPUTING EXPERIMENTAL Diskette ein.
Wenn Sie einen PC mit zwei Diskettenlaufwerken besitzen, legen Sie die Diskette mit dem BASIC-Interpreter in das Laufwerk B ein und geben Sie ein:

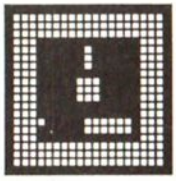
A>**B:BASICA**

Wenn Ihr Computer mit einem Festplattenlaufwerk ausgestattet ist, geben Sie ein:

A>**C:BASICA**

Bei Festplatten richtet man meist hierarchische Dateiverzeichnisse ein. Ist der BASIC-Interpreter nicht in dem Wurzelverzeichnis enthalten sondern z.B. in dem Verzeichnis mit dem Namen DOS, so geben Sie ein:

A>**C:DOS\BASICA**



Wenn Sie keinen IBM-PC sondern einen kompatiblen PC besitzen, wird Ihr BASIC-Interpreter höchstwahrscheinlich GW-BASIC von Microsoft sein. In diesem Fall müssen Sie den Namen BASICA der obigen Befehle durch GWBASIC ersetzen. Z.B:

A>**C:GWBASIC**

Obwohl die zwei BASIC-Interpreter BASICA und GW-BASIC nicht genau gleich sind und zudem in einer Vielzahl von Versionen vorkommen, so können doch BASIC-Programme in einer "defensiven" Art und Weise programmiert werden, so daß sie unter allen Versionen korrekt arbeiten. Wenn Sie jedoch vermuten, daß die Ursache für ein nicht korrektes Funktionieren eines Programms bei einer Inkompatibilität des BASIC-Interpreters zu suchen sei, so empfehlen wir Ihnen, die BASIC-Interpreter BASICA von IBM oder GW-BASIC Version 3.2 von Microsoft zu benutzen.
Unter den BASIC-Interpretern bzw. -Compilern BASIC-2, Quick-BASIC, Turbo-BASIC oder True BASIC arbeiten die fischertechnik COMPUTING EXPERIMENTAL Programme nicht.
Wenn BASIC geladen ist, wird der Schirm gelöscht und eine Copyright-Notiz des BASIC-Interpreters erscheint. Am unteren Bildschirmende erscheint eine Leiste, die die Benutzung der Funktionstasten angibt. Das Bereitzeichen ist nun:

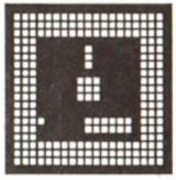
Ok

Als erstes BASIC-Programm sollten Sie das Programm FISCHER.BAS ausführen. Geben Sie daher ein:

LOAD"FISCHER"
RUN

Das Programm zeigt einige Bildschirme voll Informationen. Studieren Sie die Informationen genau. Sie enthalten viele wichtige Hinweise, die **nicht** in diesem Experimentierhandbuch stehen. Eventuell sind auch Hinweise auf Weiterentwicklungen der Software enthalten, die in diesem Handbuch nicht mehr berücksichtigt werden konnten. Wir empfehlen Ihnen, die Informationen auf Ihrem Drucker zum weiteren Nachschlagen auszugeben. Drücken Sie dazu bei jeder neuen Bildschirmseite die Tasten "Shift" und "PrtScr". Die Tasten können auch anders beschriftet sein, z.B. "⇧" und "Druck".

Nach der Ausgabe der Informationen werden Sie von dem Programm gefragt, ob Sie den Interfacetreiber an Ihren Computer, Ihre Betriebssystemversion und den benutzten BASIC-Interpreter anpassen wollen.
Vergewissern Sie sich noch einmal, daß sich die Kopie und nicht das Original der fischertechnik COMPUTING EXPERIMENTAL Diskette in Laufwerk A befindet und antworten Sie mit "J" für "ja". Die Anpassung läuft vollständig automatisch ab. Das Programm löscht den Bildschirm und zeigt schnell wechselnde Zahlen an. In einigen Fällen wird das Programm auch Buchstaben auf dem Schirm anzeigen und Sie nach der Zahl und der Lage der Buchstaben befragen. Beantworten Sie die Fragen des Programms.
Nach einer Weile läuft das Diskettenlaufwerk wieder an. Die angepaßte Version des Interfacetreibers wird wieder auf die Diskette geschrieben . Sollte an dieser Stelle eine Fehlermeldung auftreten, so vergewissern Sie sich, daß die Diskette nicht schreibgeschützt ist. Die Originaldiskette ist z.B. schreibgeschützt und soll ja auch nicht verwendet werden.
Da die Anpassung auf die Diskette geschrieben wird, benötigen Sie daher keinen weiteren Durchlauf durch das Anpassungsprogramm, solange Sie weder Computer, Betriebssystem oder BASIC-Interpreter wechseln.
In seltenen Fällen, wenn der PC nicht 100% kompatibel zu dem Original-PC ist, kann das Anpassungsprogramm mit einer Fehlermeldung enden. Schreiben Sie in diesem Fall alle Zahlen und die Fehlermeldung auf dem Bildschirm auf. Ergänzen Sie die Fehlermeldung mit der Copyright-Notiz und der Versionsnummer des BASIC-Interpreters, einer Kurzbeschreibung Ihres Computers (insbesondere auch Art des Bildschirmadapters) und seines BIOS. Die BIOS-Information ist der Text, der beim Anfahren des Computers auf dem Bildschirm erscheint. Schicken Sie diese Information an die fischerwerke. Die fischerwerke bemühen sich ständig, das fischertechnik COMPUTING System auch auf weniger kompatiblen Computern zur Verfügung zu stellen. Vielleicht können Sie von den fischerwerken einen Hinweis oder das Angebot einer neueren Version erhalten.
Nach dem Anpaßvorgang ist es unbedingt notwendig, den Computer wieder neu anzufahren. Drücken Sie dazu gleichzeitig die Tasten "Ctrl", "Alt" und "Del" (bzw. "Strg",



"Sonderzeichen" und "Entf"). Durchlaufen Sie nochmals die ersten der zuvor beschriebenen Schritte: Aktivieren Sie den Interface-Treiber (der jetzt angepaßt ist) und laden Sie BASIC.
Nun benötigen Sie einige Zeilen eines BASIC-Programms, die den Schlüssel zu dem Interfacetreiber darstellen. Diese Zeilen sind in der BASIC-Datei INIT.BAS auf der Diskette enthalten. Laden Sie also:

LOAD"INIT"

Wenn Sie das Programm mit

RUN

starten, wird der Computer mit einer Reihe neuer Befehle versehen, die bislang noch nicht vorhanden waren. Diese Befehle erlauben es Ihnen, die fischertechnik Bauelemente über das Interface per Programm zu steuern. Danach meldet sich der Computer wieder mit seinem Bereitzeichen

Ok

zurück. Ob der Interfacetreiber ordnungsgemäß läuft, können Sie mit der Eingabe:

CALL IIN

feststellen. Erscheint am Bildschirm wieder Ok, ist alles in Ordnung. Außerdem muß die Leuchtdiode auf der Leiterplatte des Interface kurz aufleuchten. Wenn sie nicht leuchtete, prüfen Sie, ob das Interface ordnungsgemäß an des Netzgerät angeschlossen ist.
Bei Fehlermeldungen beginnen Sie noch einmal von vorn, indem Sie den Computer aus- und wieder einschalten. Wenn auch dies nicht hilft, versuchen Sie noch einmal den Anpassungsdurchlauf.
Alle Befehle zum Interface wie der obige sind Aufrufe eines Programms in Maschinensprache und beginnen deshalb mit dem Befehl CALL.
In den folgenden Kapiteln werden Sie eine Menge Experimente finden, alle mit dazugehörigem Programm für Ihren Computer. Die Programme werden im Text Schritt für Schritt entwickelt. Die Programme sollten Sie, wenn sie funktionieren, auf eine Arbeitsdiskette (nicht die fischertechnik Diskette!) abspeichern. Kommen Sie mal mit einem Programm gar nicht klar oder wollen Sie geschwind ein Programm vorführen oder sich ein paar Anregungen zum Verschönern der Programme holen, so greifen

Die Arbeitsdiskette sollte mindestens folgende Dateien umfassen:

FISCHER.COM *(Der Interfacetreiber, angepaßt wie im Text beschrieben.)*

INIT.BAS *(Der Schlüssel zu dem Interfacetreiber)*

Bei PC mit nur einem Diskettenlaufwerk sollten Sie auch den BASIC-Interpreter auf die Diskette kopieren, um nicht häufig die Diskette wechseln zu müssen.
Außerdem wird empfohlen das interaktive Diagnoseprogramm zum Austesten der Modell- und Interface-Funktionen ebenfalls auf die Arbeitsdiskette zu kopieren. Es besteht aus den Dateien:

DIAGNOSE.BAS *(Das Diagnoseprogramm wie es in der Interface-Anleitung beschrieben ist.)*
und

INTERFAC.COM *(Der spezielle Interfacetreiber des Diagnoseprogramms, ebenfalls in der Interface-Anleitung beschrieben.)*

Sie zur fischertechnik Diskette (bzw. zu deren Kopie). Dort finden Sie Beispielprogramme zu den Experimenten. Das Programmstück, das jeweils als Hauptprogramm gekennzeichnet ist, ähnelt meist den im folgenden abgedruckten Programmen. Der Rest des Beispielprogramms, manchmal gar der größte Teil, dient der Bedienerführung, der Gestaltung des Bildschirms usw.
In der nachfolgenden Beschreibung der Experimente wird davon ausgegangen, daß Sie grundlegende Kenntnisse in der Erstellung von BASIC-Programmen besitzen. Sie sollten in der Lage sein, die Kommandos einzugeben, die Eingaben gegebenenfalls zu korrigieren, ein Programm zu starten, es auf dem Bildschirm auszugeben und auf Diskette zu speichern. Wenn auch im Folgenden etliche Hinweise und Hilfen zur Programmiertechnik gegeben werden, so darf dies nicht als BASIC-Kurs verstanden werden.
Wenn Sie also mehr experimentieren wollen als nur die fertigen Programme auf der Diskette zu benutzen und Sie sich nicht ganz sicher bezüglich Ihrer Programmierfertigkeiten sind, so sollten Sie zunächst gründlich die Anleitung Ihres Computers studieren und eventuell auch einen BASIC-Programmierkurs durchnehmen.

Wenn nun alles richtig funktioniert, können wir mit den ersten Experimenten beginnen. Zunächst bauen wir das Modell Seilwinde 1 nach der Bauanleitung auf. Auf einem Grundrahmen befindet sich ein Motor mit Übersetzung. Auf eine vom Motor angetriebene Querachse ist eine Seiltrommel gesteckt. Der Motor ist über das orange und das gelbe Kabel mit dem Interface verbunden. Wenn Sie das Anschlußbild auf dem Interface ansehen, werden Sie bei diesen beiden Kabeln die Bezeichnung M1 (= Motor 1) finden. Stecken Sie das Modellkabel ins Interface. Vergewissern Sie sich, daß der Interfactreiber geladen ist (s. Kapitel 3) und geben Sie ein:

LOAD"INIT"
RUN

Wenn das Bereitzeichen des BASIC-Interpreters wieder erscheint, ist alles in Ordnung. Geben Sie jetzt ein:

CALL IIN
CALL I1R

Der Motor dreht sich kurz. Der Befehl CALL IIN (initialisiere Interface) versetzt das Interface in den Grundzustand; dies muß man immer am Anfang eines Programms machen.
Mit dem Befehl CALL I1R wird der Motor 1 angesprochen. Er soll sich rechts herum drehen (CALL I1R=Motor 1 rechts). Wenn rechts möglich ist, dann sicher auch links! Probieren Sie's aus:

CALL I1L

Jetzt läuft der Motor kurz links herum. Aber warum läuft er nicht ständig? Im Interface ist eine Schutzschaltung eingebaut, die die Motorsteuerung nach ½ Sekunde abbricht, wenn das Interface vom Computer keine Kommandos mehr erhält. Dies kann auch am Erlöschen der Leuchtdiode beobachtet werden. Die Schutzschaltung soll verhindern, daß bei Fehlschaltungen, Programmierfehlern oder falschem Aufbau das Modell beschädigt wird. Stellen Sie sich vor, ein Motor wäre falsch gepolt angekabelt und würde sich deswegen verkehrt herum drehen und sich anschicken, das schöne Modell, das Sie mit Sorgfalt gebaut haben, zu zerstören. Ihre natürliche erste Reaktion ist die Ctrl- und Break-Taste zu drücken (die Ctrl-Taste kann auch mit Strg, die Break-Taste mit Scroll-Lock oder Pause beschriftet sein). Das bewirkt zwar den Ab-

Man nennt diesen Vorgang auch Initialisierung. Er ist auch für Computer sehr wichtig und wird bei jenen im allgemeinen immer nach dem Einschalten ausgeführt. Die Initialisierung bewirkt bei manchen Computersystemen, daß sogar gleich das passende Programm in den Computer geladen wird. Bei dem Interface ist dagegen ein eigener Befehl nötig. Um sicher zu sein, daß die Initialisierung auch wirklich durchgeführt wurde, werden andere Befehle wie CALL I1R nur angenommen, wenn zuerst CALL IIN gegeben wurde.

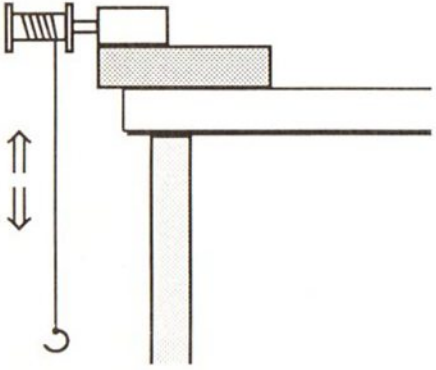


Bild 4.1: Eine Seilwinde kann man zum Experimentieren auch an einer Tischkante anbringen.

bruch des Programms, bedeutet aber nicht unbedingt, daß auch der Motor stehen bleibt. Die Schutzschaltung des fischertechnik Interface bemerkt jedoch die Unterbrechung der Datenübertragung und schaltet selbsttätig den Motor ab.
Wie kann man den Motor nun dauernd laufen lassen? Dazu erstellen wir uns eine sog. Programmschleife, geben Sie ein:

```
10 CALL IIN
20 CALL I1R
30 GOTO 20
```

Wir möchten Sie daran erinnern, daß Sie zuvor das Programm INIT geladen hatten und jetzt die obigen drei Zeilen hinzugefügt haben. Ihr vollständiges Programm ist also beträchtlich länger. Geben Sie es auf dem Bildschirm aus, um sich davon zu überzeugen. Geben Sie ein:

```
LIST
```

Das folgende Programm wird auf dem Bildschirm erscheinen:

```
1 DEF SEG=0:DEFINT E,G,I,T
2 IIN=PEEK(&H1FC)+PEEK(&H1FD)*256
    :SEG%=PEEK(&H1FE)+PEEK(&H1FF)*256
3 IF IIN=SEG% THEN PRINT "Es muß
    erst der Treiber geladen werden !"
    :END
4 DEF SEG=SEG%:CALL IIN
5 IIN=309:I1R=319:I1L=323:I1A=327
    :I2R=331:I2L=335:I2A=339
    :I3R=343:I3L=347
6 I3A=351:I4R=355:I4L=359:I4A=363
    :I1V=367:I1Z=371:I2V=375
    :I2Z=379:I3V=383:I3Z=387
    :I4V=391:I4Z=395:ITB=399
    :ITI=407:ITV=417
7 ITZ=431:ITR=445:ITL=459:IDE=473
    :IEX=481:IEY=489:ITK=497
    :ITX=505:ITY=513:IGV=521
    :IGZ=537:IGR=553:IGL=569
    :IGC=585:IG0=597:IG1=601
8 IGS=605:IGA=613:IGE=617:IGK=629
    :IGX=637:IGY=645:IGF=653:IGH=661
9 IGSAVE=669:IGLOAD=681:IHA=693
    :IHE=700:IW40=704:IW80=708
10 CALL IIN
20 CALL I1R
30 GOTO 20
```

Für die nachfolgenden Experimente müssen Sie nicht die Bedeutung der Zeilen 1 bis 9 verstehen. Denken Sie nur daran, daß sie den notwendigen Schlüssel zu den Unter-

programmen in Maschinensprache darstellen, indem sie die Anfangsadressen der Unterprogramme bereitstellen. Prüfen Sie immer bei jedem Programm für fischertechnik COMPUTING EXPERIMENTAL, daß die obigen Zeilen 1 bis 9 in dem Programm enthalten sind und vor dem Aufruf eines jeglichen Interface-Kommandos ausgeführt werden. Beachten Sie auch noch: der genaue Wortlaut der Zeilen 1 bis 9 kann je nach Version der Software auch von dem hier abgedruckten abweichen.
Starten Sie jetzt Ihr Programm mit dem Kommando

RUN

Der Motor läuft jetzt dauernd. Mit Zeile 20 läuft er ein Stück, durch Zeile 30 springt das Programm wieder nach Zeile 20 - der Motor dreht sich weiter. In dieser Schleife läuft das Programm nun endlos. Da wir aber noch andere Versuche durchführen wollen, halten wir es mit der Ctrl-Break-Taste an. Dies löst die Schutzschaltung aus und der Motor hält an. Im Interface ist jedoch noch gespeichert, daß der Motor zuletzt eingeschaltet wurde. Würden Sie jetzt

CONT

eingeben, so würde er sofort wieder laufen. Deshalb sollten wir den Motor korrekt mit dem Befehl

CALL I1A

abschalten. Damit ist auch der Steuerbefehl im Interface gelöscht. Sie werden sich fragen, warum die Schutzschaltung erst ½ Sekunde verzögert einsetzt. Schauen Sie sich das Programm an. Der Befehl in Zeile 30 benötigt einen winzigen Augenblick Zeit. In anderen Programmen werden vielleicht noch viel mehr Befehle zwischen den Steuerungen des Interface vonnöten sein. Die Schutzschaltung gewährt daher ½ Sekunde Rechenzeit.
Wenn Sie anstelle von CALL I1R in Zeile 20 den Befehl CALL I1L eingeben, läuft der Motor nach RUN dauernd links herum.
Jetzt wollen wir den Motor steuern: er soll sich eine Weile rechts herum drehen und dann stehen bleiben. Damit wird eine Seilwinde nach Bild 4.1 an der Tischkante betrieben.
Wickeln Sie dazu auf die Seiltrommel eine ca. 30 cm lange Schnur und hängen an das Ende ein Gewicht (z.B. ein Förderkorb aus Bausteinen). Um den Befehl CALL I1R nur eine bestimmte Anzahl - hier 2000

mal - auszugeben, setzen wir ihn in eine sog. FOR...NEXT-Schleife:

```
10 CALL IIN
20 FOR Z=1 TO 2000
30 CALL I1R
40 NEXT Z
50 CALL I1A
```

Geben Sie das Programm ein und starten es mit RUN. Der Motor dreht sich jetzt eine Zeit lang nach rechts, das Seil läuft nach unten. Dann stoppt der Motor - das Programm ist zuende.
Wie arbeitet nun diese FOR...NEXT-Schleife? Der Befehl in Zeile 20 enthält einen Zähler Z. Er erhält den Anfangswert 1 (...Z=1...). Danach wird Zeile 30 ausgeführt: CALL I1R d.h. der Motor 1 startet im Rechtslauf (bzw. setzt den Rechtslauf fort). In der Zeile 40 wird der Zähler um 1 erhöht (NEXT Z). Außerdem wird der Zählerstand geprüft. Solange der Zähler den Grenzwert (2000 in diesem Fall: ... TO 2000) nicht überschritten hat, springt die Programmausführung in Zeile 30 zurück, der Zeile nach der FOR-Anweisung. Hat der Zähler Z den Grenzwert überschritten, wird die Schleife verlassen und der nächste Befehl ausgeführt. Hier ist es CALL I1A in Zeile 50: der Motor wird ausgeschaltet. Wenn Sie die Schleife nur 1000 mal durchlaufen lassen wollen, ändern Sie die Zahl in der Schleifenanweisung:

```
20 FOR Z=1 TO 1000
```

Jetzt soll das Seil wieder aufgewickelt werden. Der Motor muß sich genau so lange nach links drehen, wie vorher nach rechts. Ändern Sie Zeile 30 ab:

```
30 CALL I1L
```

Nach RUN läuft das Seil wieder nach oben. Damit die Seilwinde beide Bewegungen hintereinander ausführt, setzen wir im Programm auch zwei FOR...NEXT-Schleifen hintereinander. Die erste ist für die Abwärtsbewegung, die zweite für den Weg zurück nach oben.

```
10 CALL IIN
20 FOR Z=1 TO 2000
30 CALL I1R
40 NEXT Z
50 FOR Z=1 TO 2000
60 CALL I1L
70 NEXT Z
80 CALL I1A
```

Im Folgenden werden wir Sie nicht an das Speichern der Programme erinnern; es liegt bei Ihnen, das aufzubewahren, was Sie für wert halten. Es wird auch davon ausgegangen, daß zu Beginn eines jeden neuen Kapitels Sie mit leerem Programmspeicher anfangen und zuerst die Schlüsselzeilen laden.

Mit RUN starten Sie das Programm. Wenn das Seil wieder oben ist, ist das Programm zu Ende. Sie können das Seil auch dauernd auf- und abwärts laufen lassen, indem Sie dem Programm sagen, daß es am Ende wieder vorn anfangen soll:

90 FOR Z=1 TO 1000
100 NEXT Z
110 GOTO 20

Vor dem erneuten Abwärtslauf wartet das Programm eine Zeitlang. Auch dafür benutzen wir wieder eine FOR...NEXT-Schleife (Zeile 90-100). Hier wird nichts anderes gemacht, als von 1 bis 1000 gezählt, da innerhalb der Schleife keine Anweisung wie im vorherigen Versuch steht. Man nennt solche Schleifen, die den Programmablauf eine gewisse Zeit verzögern sollen, daher auch "Warteschleifen".
Mit Zeile 110 springt das Programm jetzt wieder zum Anfang (Zeile 20), worauf der Vorgang von neuem beginnt.
Mit RUN wird die Seilwinde in Gang gesetzt, mit der Ctrl-Break-Taste läßt sie sich anhalten.
Wie Sie dem Bild auf dem Interface entnehmen können, lassen sich maximal vier Motoren ansteuern. Für jeden Motor sind zwei Leitungen vorgesehen, für Motor 2 z.B. grün und blau. Schließen Sie den Motor an diese Leitungen an, so ist der bisherige Befehl CALL I1R nicht mehr wirksam. Motor 2 wird mit

CALL I2R bzw. **CALL I2L**

angesprochen. Analog dazu lassen sich Motor 3 und 4 betreiben:

CALL I3R bzw. **CALL I3L**
CALL I4R bzw. **CALL I4L**

Bevor Sie mit dem nächsten Programm beginnen, vergewissern Sie sich bitte, ob Sie nicht das vorige Programm aufbewahren möchten. Um es auf Diskette zu schreiben, benutzen Sie den Befehl:

SAVE"MOTOR1"

Anstelle MOTOR1 können Sie natürlich auch einen anderen Namen Ihrer Wahl verwenden. Nun löschen Sie den Programmspeicher und laden wieder die Schlüsselzeilen durch Eingabe von:

LOAD"INIT"

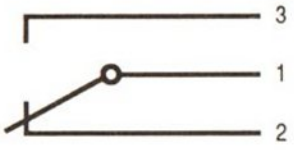


Bild 4.2: Schaltzeichen eines Tasters.

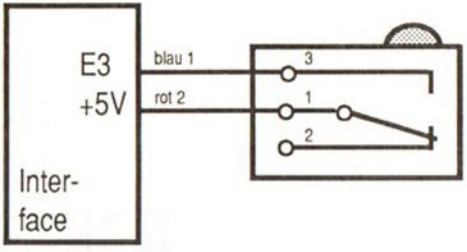


Bild 4.3: Verbindung vom Taster zum Interface.

4.2. Schaltkontakte und Taster: Eingabe

Neben Motoren werden bei den Modellen aus dem Baukasten oft Schalter bzw. Taster benutzt. Wie diese Bauteile funktionieren und wie sie vom Computer abgefragt werden können, soll im Folgenden gezeigt werden.

Nehmen Sie zunächst einen Taster aus Ihrem Baukasten. Er hat drei Anschlußbuchsen, neben denen die Schaltfunktion dargestellt ist, wie Bild 4.2 zeigt.

Dieser Taster hat zwei Schaltkontakte. In Ruhestellung, wenn der Taster nicht betätigt wird, besteht eine leitende Verbindung zwischen den Anschlüssen 1 und 2. Drükken Sie auf den Tastknopf, wechselt der Schaltkontakt an Anschluß 1 zur anderen Seite. Jetzt haben wir eine Verbindung zwischen Anschluß 1 und 3 - der Weg von 1 nach 2 ist unterbrochen. Dieser Zustand bleibt solange bestehen, wie der Taster gedrückt wird. Läßt man ihn los, geht er wieder in Grundstellung (Verbindung 1-2). Damit kennen wir nun auch den Unterschied zwischen Schaltern und Tastern. Ein Schalter - z.B. der Lichtschalter in Ihrem Zimmer - wird einmal betätigt und bleibt dann selbständig in dieser Stellung (AUS oder EIN). Ein Taster hat eine Grundstellung; er schaltet nach Betätigung um und geht nach Loslassen wieder von selbst in die Grundstellung zurück.

Wir wollen den Taster jetzt am Interface betreiben und schließen ihn dazu nach Bild 4.3 an. Anschluß 1 verbinden wir mit dem +5V-Anschluß am Interface (Leitung Rot 2), den Schaltkontakt 3 schließen wir an einem Eingang an. Wir können z.B. Eingang E3 (Blau 1) wählen. Um die Eingabeleitung E3 vom Computer abzufragen, geben Sie ein:

```
LOAD"INIT"
10 CALL IIN
30 CALL IDE(E1,E2,E3,E4,E5,E6,E7,E8)
```

Der Befehl CALL IDE(...) liest die Werte aller Eingabeleitungen des Interfaces ein. Auf dem Bildschirm angezeigt wird der Wert E3 mit

```
50 PRINT E3
```

Nach dem Start des Programms erscheint auf dem Bildschirm eine "0", d.h. die Verbindung zwischen 1 und 3 ist unterbrochen, wie wir aus Bild 4.3 auch sehen können. An E3 liegt keine Spannung. Drücken Sie jetzt auf den Taster und starten Sie das Programm nochmals. Jetzt wird eine "1" ange-

zeigt: der Schaltkontakt ist geschlossen, an E3 liegen +5V an.
Man nennt den Kontakt 3 einen "Schließerkontakt", da er beim Betätigen des Tasters schließt.
Um nicht jedesmal das Programm starten zu müssen, wollen wir zum Einlesen des Schaltzustandes das Programm verändern:

```
20 CLS
40 LOCATE 1,1
60 GOTO 30
```

Die Anweisung CLS (clear screen) in Zeile 20 löscht den Bildschirm. In Zeile 30, die wir zuvor schon eingegeben hatten, wird der Wert von E3 gemessen (CALL IDE). Bevor er angezeigt wird, wird die Druckposition noch auf die linke obere Bildschirmecke gesetzt. Dies wird durch die Anweisung LOCATE bewirkt. Die Zahlen nach dem Befehlswort bedeuten Zeilen- und Spaltennummer.
In Zeile 60 steht ein Sprungbefehl nach Zeile 30, worauf das Programm wieder von vorn beginnt. Wir kennen diese Schleife bereits aus dem letzten Kapitel. Sie wird solange wiederholt, bis wir sie mit der Ctrl-Break-Taste unterbrechen.

Starten Sie das Programm mit RUN. Der Bildschirm wird gelöscht und anschließend der Schaltzustand laufend angezeigt. Drücken Sie den Taster, so wird dies sofort vom Programm erkannt. Damit Sie auch wissen, was "0" und "1" bedeuten, ergänzen wir das Programm:

```
50 IF E3=0 THEN PRINT "Taster aus"
55 IF E3=1 THEN PRINT "Taster ein"
```

Nach RUN wird der entsprechende Text angezeigt. Die Zuordnung erreichen wir mit einer IF...THEN-Abfrage in Zeile 50 und 55. Dieser Befehl prüft eine Bedingung: wenn (IF) etwas zutrifft (E3=0), dann (THEN) führe aus: PRINT "Taster aus".
Wenn die Bedingung nicht erfüllt ist (E3 nicht 0 ist), dann wird der nächste Befehl ausgeführt. Dieselbe Abfrage führen wir auch bei E3=1 durch, also für den gedrückten Taster.
Wie wir oben gesehen haben, besitzt der Taster noch einen zweiten Kontakt, den Anschluß 2. Stecken Sie das Kabel von 3 nach 2 und starten das Programm erneut mit RUN. Auch jetzt wird wieder der Schaltzustand des Tasters angezeigt, nur ist er am Anfang "ein". Zuvor war er bei Verwendung des Kontaktes 3 "aus". Prüfen Sie's

zur Kontrolle noch einmal nach!
Der Taster ist also in Ruhestellung zwischen Anschluß 1 und 2 geschlossen, wie wir auch aus Bild 4.3 sehen können. Diesen Kontakt nennen wir daher auch "Öffnerkontakt", da er bei Betätigung des Tasters öffnet. Probieren Sie die unterschiedlichen Schalterstellungen mit dem Programm aus. Später werden wir sie öfter für Steuerungen und Zählungen benötigen.
Zum Schluß wollen wir noch eine praktische Anwendung mit dem Taster durchführen: Prüfen Sie Ihre Reaktion! Sind Sie noch fit genug, um die weiteren Versuche durchzustehen? Geben Sie folgendes Programm ein:

```
LOAD"INIT"
10 CALL IIN
20 CLS
30 PRINT "Wenn ein Feld erscheint,"
35 PRINT "drücken Sie den Taster!"
40 PRINT
45 I=0
50 FOR Z=1 TO 2000
55 NEXT Z
60 COLOR 0,7
65 PRINT" "
70 COLOR 7,0
75 CALL IDE(E1,E2,E3,E4,E5,E6,E7,E8)
80 I=I+1
85 IF E3=1 THEN GOTO 75
90 PRINT
95 PRINT"Stop nach: ";I
100 END
```

Am Taster sind die Kontakte 1 und 2 angeschlossen. Starten Sie das Programm und drücken Sie den Taster, wenn das Feld erscheint. Wenn Sie ein Ergebnis unter 20 erreichen, dann sind Sie wirklich topfit! Die Zahl 20 ist nichts anderes als die Anzahl der Schleifendurchläufe durch die Zeilen 75, 80 und 85 bis der Taster gedrückt wird. Die Zahl gilt in diesem Fall für den IBM-XT. Wenn Sie einen schnelleren Computer besitzen, z.B. ein IBM-AT, wird der Computer in der gleichen Zeit wesentlich mehr Schleifendurchläufe bewerkstelligen können. Es wird Ihnen dann schwer fallen, die 60 zu unterbieten. Mit RUN können Sie einen erneuten Versuch starten.
Die Bedeutung der meisten Programmzeilen kennen Sie bereits aus den bisherigen Beispielen. Das Feld wird mit Zeile 60 erzeugt, es handelt sich um ein inverses Leerzeichen. Um es zu erzeugen, muß bei der Bildschirmdarstellung die Schriftfarbe mit der Hintergrundfarbe vertauscht wer-

den. Dies wird durch die COLOR-Anweisung bewirkt, die dieFarben für Schrift und Hintergrund festlegt. In Zeile 60 wird die normale Zuordnung gerade vertauscht. Danach wird das Leerzeichen gedruckt, das nun sichtbar als weißes Rechteck erscheint. In Zeile 70 wird wieder auf die alte Zuordnung zurückgestellt.
In den nächsten Kapiteln sehen Sie noch mehr Beispiele, wie man einen Taster in einem Programm sinnvoll einsetzen kann.

4.3. Motorsteuerung mit Tastern: Seilwinde

Wir haben bisher Anschluß und Steuerung eines Motors sowie Handhabung von Tastern kennengelernt. Jetzt wollen wir beide Bauteile miteinander verbinden. Vom Programm soll die Stellung eines Tasters abgefragt und damit ein Motor gesteuert werden.
Zu diesem Versuch bauen wir das Modell Seilwinde 2 aus der Bauanleitung zusammen. Auf dem Grundrahmen befindet sich der Motor mit der Seilwinde, auf die wir wieder eine Schnur von ca. 30 cm Länge mit Gewicht am Ende wickeln. Außen am Rahmen sind zwei Taster angebracht. Die Anschlüsse 1 beider Taster liegen auf +5V (Kabel Rot 2 vom Interface). Der rechte Taster ist mit E3 (Kabel Blau 1), der linke mit E2 (Kabel Rot 1) verbunden. Wir benutzen jeweils Anschluß 3 der Taster, also den Schließerkontakt.
Wenn alles angeschlossen ist, entwerfen wir das Programm. Zunächst soll der Motor solange laufen, bis ein Taster gedrückt wird. Geben Sie ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL I1R
30 CALL IDE(E1,E2,E3,E4,E5,E6,E7,E8)
40 IF E3=0 THEN GOTO 30
50 CALL I1A
```

Nach RUN wird der Motor dauernd laufen. Drücken Sie den rechten Taster, bleibt er stehen. Die Abfrage des Schaltzustandes erfolgt in Zeile 40. Solange E3=0 ist, also der Schließerkontakt (1-3) offen ist, springt das Programm immer wieder nach Zeile 30. Dort wird erneut der Eingang eingelesen. Der Motor läuft derweil weiter, auch wenn er kein neues Kommando erhält. Um den Motor am Laufen zu halten, genügt es auch, die Eingänge des Interface abzufragen. Erst wenn weder Ein- noch Ausgänge abgefragt werden, schaltet das Interface nach einer halben Sekunde alle Motoren ab, weil es annimmt, daß das Programm angehalten wurde (z.B. durch die Ctrl-Break-Taste oder eine Fehlermeldung).
Trifft die Bedingung E3=0 nicht mehr zu (Taster gedrückt), wird der Motor ausgeschaltet (Zeile 50).
Wir können für die Steuerung auch den anderen Taster benutzen:

40 IF E2=0 THEN GOTO 30

Nach RUN muß jetzt der linke Taster betätigt werden, um den Motor zu stoppen. Wir können auch beide Taster benutzen:

40 IF E3=0 AND E2=0 THEN GOTO 30

Bei dieser IF...THEN-Abfrage müssen zwei Bedingungen zutreffen, damit der folgende Befehl (... GOTO 20) ausgeführt wird. Die beiden Bedingungen sind E3=0 und (AND) E2=0, d.h. beide Taster dürfen nicht gedrückt sein, damit der Motor läuft. Man nennt dies eine logische UND-Verknüpfung.
Dies läßt sich auch anders lösen:

40 IF E3=1 OR E2=1 THEN GOTO 60
50 GOTO 30
60 CALL I1A

In Zeile 40 prüfen wir jetzt, ob Taster 3 oder (OR) Taster 2 gedrückt sind. Wenn ja, springt das Programm nach Zeile 60 (...THEN GOTO 60), und der Motor hält an. Ansonsten läuft er weiter (GOTO 30). Dies nennt man eine logische ODER-Verknüpfung, bei der die eine oder die andere Bedingung zutreffen muß, damit der Befehl weiter ausgeführt wird.
Nach RUN ist es egal, welchen Taster Sie zum Anhalten des Motors benutzen.
Jetzt wollen wir den linken Taster für "Seilwinde aufwärts" und den rechten für "Seilwinde abwärts" einsetzen.

20 CLS

```
40 IF E3=1 AND E2=0 THEN CALL I1R
50 IF E3=0 AND E2=1 THEN CALL I1L
60 GOTO 30
```

Für die Bewegungsrichtung müssen wir aus Sicherheitsgründen jeweils beide Taster abfragen. Einer muß frei sein, wenn der andere gedrückt wird. Sonst würden zwei sich widersprechende Befehle kurz hintereinander zum Interface geschickt werden, wenn man beide Taster drückt.
Geben Sie die Zeilen ein und starten das Programm mit RUN. Sie können das Seil jetzt mit den beiden Tastern beliebig hinauf- und herunterfahren. Anhalten läßt sich der Motor mit der Ctrl-Break-Taste. In den Zeilen 40 und 50 finden wir wieder die UND-Verknüpfung (AND) von zwei Bedingungen, die für die Ausführung des Befehls beide erfüllt sein müssen.
Man kann auch die Taster als Startknopf für eine vollständige Auf- oder Abwärtsbewegung der Seilwinde benutzen. Dazu geben Sie ein:

```
40 IF E3=1 AND E2=0 THEN GOTO 70
50 IF E3=0 AND E2=1 THEN GOTO 120
70 FOR Z=1 TO 2000
80 CALL I1R
90 NEXT Z
100 CALL I1A
110 GOTO 30
120 FOR Z=1 TO 2000
130 CALL I1L
140 NEXT Z
150 CALL I1A
160 GOTO 30
```

Wenn das Seil vollständig aufgewickelt ist, starten Sie das Programm mit RUN. Durch kurzes Drücken des rechten Tasters läuft das Seil nach unten. Dazu dient der Programmteil in Zeile 70-110, den wir bereits aus dem letzten Kapitel kennen.
Zurückziehen läßt sich das Seil wieder durch kurzes Drücken des linken Tasters. Die Programmschritte hierzu finden wir in Zeile 120-160. Aufgerufen werden diese Programmteile mit den IF...THEN-Abfragen in Zeile 40 und 50. Treffen die Bedingungen zu, wird der Befehl ...THEN GOTO 70 bzw. ...THEN GOTO 120 ausgeführt.
Was passiert, wenn das Seil unten ist und Sie drücken die Taste "Seil abwärts"? Die Trommel dreht sich so, als wolle sie das Seil abwickeln, rollt es aber falsch herum wieder auf. Damit das nicht passiert, merken wir uns die Position des Seils und lassen das Programm nur die richtige Folgebewegung ausführen. Wenn das Seil oben ist, geht's nur nach unten und umgekehrt. Halten Sie

das laufende Programm mit der Ctrl-Break-Taste an und geben ein:

```
25 PO=0
40 IF E3=1 AND E2=0 AND PO=0 THEN
   GOTO 70
50 IF E3=0 AND E2=1 AND PO=1 THEN
   GOTO 120
95 PO=1
145 PO=0
```

Am Anfang muß das Seil oben sein; die Position halten wir in Zeile 25 fest: die Variable PO wird auf 0 gesetzt. Die IF...THEN-Abfrage in Zeile 40 und 50 haben wir um eine zusätzliche Bedingung erweitert. So kann das Programm nur nach Zeile 70 springen, wenn Taster 3 gedrückt ist, Taster 2 frei ist und das Seil oben ist (PO=0). Nur dann kann das Seil nach unten laufen. In Zeile 50 ist es genau umgekehrt: Taster 3 frei, Taster 2 gedrückt und Seil unten (PO=1). Dann wird das Seil hochgezogen. Die jeweilige Position halten wir in Zeile 95 bzw. 145 fest, nachdem die entsprechende Bewegung durchgeführt wurde. Beenden läßt sich das Programm wieder mit der Ctrl-Break-Taste.
Sie sehen, wie man mit den Tastern gezielt den Motor steuern kann - entweder durch direkte Laufbefehle (CALL I1R, CALL I1L) oder durch Aufruf eines Programmteils für einen längeren Lauf. Ebenso lassen sich Taster- und Motorstellung miteinander verbinden (logisch verknüpfen).
Auf Diskette finden Sie ein Programm mit dem Namen TASTER.BAS. Es führt Sie ausführlich in die Steuerung der Motoren ein.

4.4. Kommandos und Positionen: Schritt für Schritt

Wer die vorigen Versuche aufmerksam beobachtet hat, wird sicher bemerkt haben, daß das Seil beim Vor- und Rücklauf der Winde nicht immer an der gleichen Stelle anhielt. Der Grund dafür liegt in der Zeitsteuerung des Motors. Genauer ist die Steuerung nach dem Schrittsteuerprinzip. Hier wird jede Umdrehung des Motors gezählt. Man kann so das Seil genau 10 cm nach unten laufen lassen, indem man die Motorschritte vorgibt.
Wie man die Schritte zählt und damit den Motor steuert, soll im folgenden Versuch gezeigt werden.
Dazu bauen wir das Modell Seilwinde 3 aus der Bauanleitung zusammen. Auf dem Grundrahmen befindet sich wieder der Motor mit der Seilwinde. Auf der Seite der Seiltrommel ist ein Taster montiert, der mit dem Eingang E2 (Kabel Rot 1) des Interface verbunden ist. Er ist so angebracht, daß die Nocken der Seiltrommel ihn nach jeder halben Umdrehung betätigen.
Unser Programm soll nun so aussehen, daß der Motor anläuft und nach Betätigung des Tasters durch den Schaltnocken anhält. Geben Sie dazu ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL I1L
50 CALL IDE(E1,E2,E3,E4,E5,E6,E7,E8)
60 IF E2=0 THEN GOTO 50
70 CALL I1A
```

Stellen Sie die Seiltrommel so, daß der Taster nicht gedrückt ist. Nach RUN bewegt sich der Motor, bis der Taster schaltet. Danach bleibt er stehen und hat dabei eine halbe Umdrehung hinter sich gebracht. Die Programmschritte 10 bis 70 kennen wir bereits aus dem letzten Kapitel: hier hatten wir den Taster mit der Hand betätigt.
Schauen Sie nun genau auf den Taster. Vielleicht ist der Taster schon wieder freigegeben, weil der Betätigungsnocken schon über das Ziel hinausgeschossen ist. In diesem Fall können Sie mit dem Kommando RUN den nächsten Schritt aufrufen. Stellen Sie jetzt aber die Seiltrommel mal so, daß der Nocken den Taster drückt und geben Sie jetzt das Kommando RUN. Der Motor wird nur einen kurzen, kaum merklichen Ruck ausführen und schon wieder stehen. Der Grund: Da der Taster schon gedrückt war, war die Abfrage in Zeile 40 sofort erfüllt und das Programm wurde sofort beendet. Geben Sie folgende Zeilen ein:

```
30 CALL IDE(E1,E2,E3,E4,E5,E6,E7,E8)
```

Dieses Schrittsteuerprinzip findet man in der Digitaltechnik häufig. Neben Robotern, Diskettenlaufwerken und Druckern werden auch so alltägliche Dinge wie Quarz-Armbanduhren mit Zeigern mit einem schrittgesteuerten Motor betrieben. Wenn man genau hinschaut, kann man auch sehen, wie sich der Sekundenzeiger in ganz winzigen Schritten weiterbewegt.

Es kann vorkommen, daß Sie beim Verkabeln oder dem Aufbau eines Modells einen Fehler gemacht haben und der Schaltnocken nicht den (richtigen) Taster betätigt. Der Motor läuft dann ständig. Wenn Sie jetzt das Programm mit Ctrl-Break stoppen wollen, werden Sie feststellen, daß sich das Programm nicht anhalten läßt. Drücken Sie dann noch die Funktionstatse F10. Jetzt bleiben Programm und Motor stehen. Wir geben dafür auch die Erklärung:
Die Taste Ctrl-Break wirkt nur auf BASIC-Anweisungen. Wenn aber auf die Tasterbetätigung durch den Schaltnocken gewartet wird, befindet sich der Computer in dem Unterprogramm in Maschinensprache. Der Interfacetreiber stellt in dieser Situation die Taste F10 als zusätzliche Stoptaste zur Verfügung.

```
40 IF E2=1 THEN GOTO 30
```

In Zeile 30 wird jetzt zunächst gewartet, bis der Taster freigegeben ist. Danach kann in Zeile 60 geprüft werden, ob der Taster wieder gedrückt wird. Dieses Programm arbeitet nun bei jeder beliebiger Anfangsstellung der Seiltrommel.
Da dieser Programmablauf sehr oft benötigt wird, gibt es dafür ein eigenes Kommando:

CALL I1V (Motor 1 vorwärts)

Das Kommando arbeitet viel schneller als die fünf Zeilen BASIC-Programm, denn es enthält die gleichen Befehle in Maschinensprache. Die Positionierung wird also mit diesem Kommando auch genauer sein.
Wenn wir den Motor z.B. zehn Umdrehungen laufen lassen wollen, müssen wir 20 mal diesen Befehl ausgeben (1 Befehl = ½ Umdrehung):

```
LOAD"INIT"
10 CALL IIN
20 FOR Z=1 TO 20
30 CALL I1V
40 NEXT Z
50 END
```

Ebenso läßt er sich in die andere Richtung drehen; ändern Sie:

30 CALL I1Z (Motor 1 zurück)

Nach RUN läuft der Motor zehn Umdrehungen zurück.
Neben den beiden Kommandos CALL I1V und CALL I1Z gibt es diese Kommandos natürlich auch noch für die Motoren 2, 3 und 4. Also:

CALL I2V	und	**CALL I2Z**
CALL I3V	und	**CALL I3Z**
CALL I4V	und	**CALL I4Z**

Mit diesen neuen Befehlen läßt sich unsere Seilwinde nun genau positionieren. Das Programm dazu lautet:

```
LOAD"INIT"
10 CALL IIN
20 CLS
30 PO=0
40 A$=INKEY$
50 IF A$="" THEN GOTO 40
60 IF PO=1 THEN GOTO 120
70 FOR Z=1 TO 20
80 CALL I1Z
90 NEXT Z
```

```
100 PO=1
110 GOTO 40
120 FOR Z=1 TO 20
130 CALL I1V
140 NEXT Z
150 PO=0
160 GOTO 40
```

Das Seil auf der Winde befindet sich oben; starten Sie das Programm mit RUN. Der Motor bewegt sich noch nicht! Sie müssen jetzt eine Taste drücken, damit das Seil nach unten läuft. Die Abfrage der Tastatureingabe erfolgt in Zeile 40. Die INKEY$-Funktion liest den Code der gedrückten Taste ein und speichert ihn in der Variablen A$.
Wird keine Taste gedrückt, ist A$ leer (A$=""). Sobald eine Taste gedrückt wird, verläßt das Programm die Schleife, und das Seil läuft nach unten. Die Anfangsposition war oben (PO=0); damit kann das Programm nur nach Zeile 70 springen. Wenn das Seil unten ist, wartet das Programm wieder auf eine Tastatureingabe. Drücken Sie eine Taste, und das Seil läuft wieder nach oben, da jetzt PO=1 ist (Zeile 60). Das Seil ist dabei im Uhrzeigersinn auf die Trommel gewickelt. Mit der Ctrl-Break-Taste wird das Programm beendet.

Wir können das Seil jetzt auch auf halber Strecke anhalten, indem wir nur zehn Drehschritte vorgeben:

```
70 FOR Z=1 TO 10
120 FOR Z=1 TO 10
```

Nach RUN und Tastendruck läuft das Seil bis zur Mitte und zurück.
Wenn wir die Schrittzahl erst nach Programmstart eingeben, läßt sich das Seil gezielt auf jede Position fahren:

```
35 INPUT"Schritte (1-20):";S

70 FOR Z=1 TO S
120 FOR Z=1 TO S
```

Nach RUN geben Sie die Schrittzahl ein; sie wird in S gespeichert. Nach einem Tastendruck läuft das Seil diese Anzahl Schritte vor und auch wieder zurück. Der Endwert der FOR...NEXT-Schleifen in Zeile 70 und 120 ist der Wert von S.
Das Erreichen des Ziels können wir auch durch einen Soll-/Istwert-Vergleich der Schrittzahl kontrollieren. Wir geben wieder die Schrittzahl S vor und starten das Seil durch Tastendruck. Geben Sie zuvor folgende Zeilen ein:

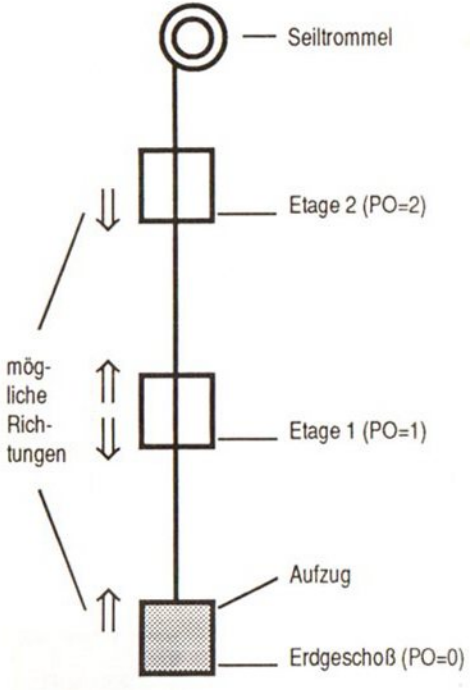


Bild 4.4: Arbeitsweise eines Aufzugs.

```
45 Z=0

70 CALL I1Z
80 Z=Z+1
90 IF Z<S THEN GOTO 70

120 CALL I1V
130 Z=Z+1
140 IF Z<S THEN GOTO 120
```

In Zeile 45 wird der Schrittzähler Z auf 0 gesetzt. Bei der Abwärtsbewegung wird in Zeile 80 jeder Schritt gezählt (Z=Z+1) und in Zeile 90 mit dem Sollwert S verglichen. Solange Z kleiner als S ist (Z<S), fährt das Programm weiter mit Zeile 70 fort: das Seil läuft abwärts. Wenn die Bedingung in Zeile 90 nicht mehr erfüllt ist - also Z die vorgegebenen Schritte erreicht hat -, hält der Motor an. Das Programm springt zum Anfang nach Zeile 40 zurück. Für die Aufwärtsbewegung gilt das Gleiche: Motordrehung solange, bis die Schrittzahl erreicht ist.
Auf diesen Soll-/Istwert-Vergleich mit Abfrage "größer als..." oder "kleiner als..." werden wir noch öfters stoßen.
Fahren Sie das Seil nun um 20 Schritte nach unten und beenden das Programm mit der Ctrl-Break-Taste. Wir wollen mit den neuen Befehlen jetzt eine praktische Anwendung kennenlernen: einen Fahrstuhl mit drei Etagen. Mit Hilfe der Cursortasten ↑ und ↓ lassen wir den Lift nach oben und unten fahren. Als Aufzug benutzen wir unsere Seilwinde. Bild 4.4 zeigt den Fahrstuhl.
Für das Fahrstuhlprogramm löschen Sie den Programmspeicher und geben ein:

```
LOAD"INIT"
10 CALL IIN
20 CLS
30 PO=0
40 AUF$=CHR$(0)+CHR$(72)
50 AB$=CHR$(0)+CHR$(80)
55 LOCATE 1,1
60 PRINT "Etage:";PO
70 A$=INKEY$
80  IF A$=AUF$ AND PO<>2 THEN
       GOTO 160
90  IF A$=AB$ AND PO<>0 THEN
       GOTO 110
100 GOTO 55
110 FOR Z=1 TO 10
120 CALL I1Z
130 NEXT Z
140 PO=PO-1
150 GOTO 55
160 FOR Z=1 TO 10
170 CALL I1V
```

180 NEXT Z
190 PO=PO+1
200 GOTO 55

Diesmal ist das Seil ganz ausgefahren, der Lift steht in Grundstellung ganz unten (Position PO=0 in Zeile 30). Starten Sie das Programm mit RUN. Auf dem Bildschirm erscheint die Anzeige, auf welcher Etage Sie sich befinden. Geben Sie die Richtung des Fahrstuhls ein: z.B. ↑, wenn Sie nach oben möchten. Der Lift fährt hoch nach Etage 1 und zeigt Ihnen das an.
Von Etage 1 können Sie nach oben und unten fahren, von den Etagen 0 und 2 nur in den angegebenen Richtungen nach Bild 4.4. Wenn der Lift auf Etage 2 steht, kann er nicht nach oben fahren, von Etage 0 kann er nicht weiter nach unten fahren.
Nach Bestimmung der Fahrtrichtung erfolgt in den Zeilen 80 und 90 die Auswahl des entsprechenden Programmteils. Die Etagenposition ist in den IF...THEN-Abfragen mit dem Cursortastencode UND-verknüpft. Die Codes für die Cursortasten sind in den Zeilen 40 und 50 festgelegt. Anders als bei den Buchstaben und Ziffern kann der Code für die Cursortasten nicht mit Anführungsstrichen eingegeben werden. Stattdessen muß in dem Anleitungsbuch des Computers die interne Verschlüsselung der Taste nachgeschlagen werden und in der Funktion CHR$(...) angegeben werden. Diese wandelt sie in den Tastencode um. Im Moment brauchen Sie sich um solche Details nicht zu kümmern, die richtigen Codes sind bereits in den Zeilen 40 und 50 angegeben. Sie sollten nur aufpassen, daß Sie nicht die Num-Lock-Taste gedrückt haben, denn die schaltet die Cursoreingabe auf Zahleneingabe um.
Probieren Sie selbst, in das Programm weitere Etagen einzubauen. Eine andere Anwendung von Motoren mit Schrittsteuerung ist ein Förderband, das schrittweise um einen bestimmten Betrag vorwärts läuft. Zwischendurch hält es immer wieder eine Zeitlang an. Versuchen Sie auch hierzu ein Programm zu schreiben, in dem Sie z.B. mit den Cursortasten rechts und links das Band jeweils 5 Schritte vor- und zurücklaufen lassen.
Eine Variante der Schrittsteuerung ist die Positionierung des Motors durch Angabe der Zielposition (Sollwert). Dabei merkt sich das Programm die momentane (Ist-) Position und entscheidet durch Soll-/Istwert-Vergleich, in welche Richtung sich der Motor drehen soll, um ans Ziel zu kommen. Als Positionsangabe wird die Schrittzahl

benutzt.
Wir verwenden für unseren Versuch wieder das vorige Modell und wickeln das Seil ganz auf die Rolle. Dies soll die Position 0 sein. Geben Sie folgendes Programm ein:

```
LOAD"INIT"
10 CALL IIN
20 Z=0
30 CLS
40 INPUT"Position (0-20):";S
50 IF Z<S THEN GOTO 80
60 IF Z>S THEN GOTO 130
70 GOTO 30
80 FOR I=Z+1 TO S
90 CALL I1Z
100 NEXT I
110 Z=S
120 GOTO 30
130 FOR I=Z-1 TO S STEP -1
140 CALL I1V
150 NEXT I
160 Z=S
170 GOTO 30
```

Die Anfangsposition 0 wird in Zeile 20 mit Z=0 markiert. Nach Programmstart mit RUN geben Sie die gewünschte Position ein: eine Zahl zwischen 0 und 20. Das Programm prüft durch Vergleich (Zeile 50 und 60), ob die Zielposition (Sollwert) grösser oder kleiner ist als die Startposition (Istwert). Ist Z<S, läuft der Motor die entsprechende Schrittzahl vor (Zeile 80-100). Stören Sie sich nicht daran, daß hier CALL I1Z steht: das Seil ist im Uhrzeigersinn aufgewickelt und läuft nach unten. Danach merkt sich das Programm die neue Position als Istwert in der Variablen Z (Zeile 110) und springt wieder nach Zeile 30. Anschließend erwartet es eine neue Eingabe.
Ist die Sollposition kleiner als der Istwert, läuft der Motor in die andere Richtung zurück (Zeilen 130-170). In Zeile 130 zählt die Schleife rückwärts (...STEP -1), also vom höheren Wert Z zum kleineren Wert S in 1er-Schritten. Auch hier wird die Zielposition festgehalten, um mit ihr nach der nächsten Eingabe die Drehrichtung des Motors zu ermitteln.
Zur Übung finden Sie auf der Diskette zwei Programme: mit Programm SCHRITT.BAS können Sie einen Motor schrittweise vor- und zurückdrehen, mit POSITION.BAS läßt er sich auf bestimmte Positionen bewegen. Der Ablauf wird jeweils auf dem Bildschirm dargestellt.

5. Schalten mit Licht

5.1. Berührungslos schalten: Gabellichtschranke

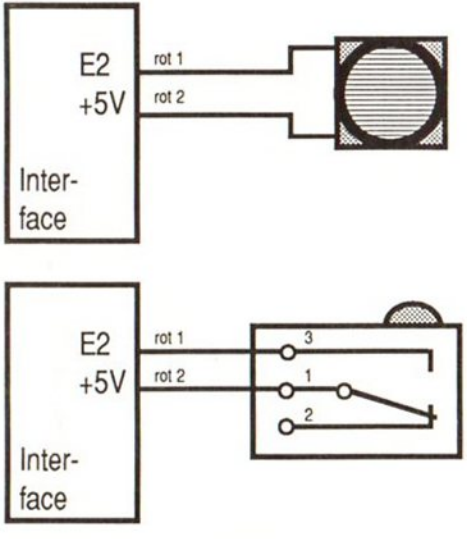


Bild 5.1: Ein Fotowiderstand ersetzt den Taster aus Bild 4.3.

Anstelle des mechanischen Tasters zur Messung der Schrittzahl und Steuerung eines Motors kann man auch eine Lichtschranke einsetzen. Sie besteht aus einem Lichtsender und einem Empfänger. Wird der Lichtstrahl zwischen den beiden Bauteilen unterbrochen, liefert sie ein Signal. Wie man die Lichtschranke mit dem Motor betätigen und ihn damit steuern kann, wird im Folgenden gezeigt.

Wir bauen zunächst das Modell Gabellichtschranke aus der Bauanleitung auf. Der Motor auf dem Grundrahmen treibt jetzt eine senkrechte Achse an, auf der eine Scheibe mit sechs Schlitzen sitzt. Am Außenrand der Scheibe befindet sich die Lichtschranke. Von oben leuchtet eine Lampe, die am Ausgang M3 des Interfaces angeschlossen ist, auf den Lichtempfänger. Dieses Bauteil, ein lichtempfindlicher Widerstand (Fachausdruck: Fotowiderstand), ist am Eingang E2 angeschlossen. Mit folgendem kurzen Testprogramm wird die Funktion des Aufbaus geprüft:

```
LOAD"INIT"
10 CALL IIN
20 FOR I=1 TO 1000
30 CALL I1R
40 CALL I3R
50 NEXT I
60 CALL I1A
70 CALL I3A
```

Nach dem Start des Programms mit RUN dreht der Motor das Rad einige Zentimeter vor. Dabei leuchtet die Lampe auf.

Bevor wir die Funktion des Fotowiderstandes prüfen, müssen wir zunächst wissen, wie er arbeitet.

Der Fotowiderstand ist ein lichtabhängiger Widerstand. Er ändert seinen Widerstand, also seine Leitfähigkeit für den elektrischen Strom, mit der Stärke des einfallenden Lichtes. Drehen Sie das Rad auf der Achse so, daß ein Spalt über dem Fotowiderstand steht. Geben Sie das nächste Testprogramm ein:

```
LOAD"INIT"
10 CALL IIN
50 CALL IDE(E1,E2,E3,E4,E5,E6,E7,E8)
60 PRINT E2
```

Am Bildschirm erscheint die Anzeige "0". Wenn nicht, ist vielleicht die Raumhelligkeit zu groß. Achten Sie darauf, daß kein direktes Licht auf den Fotowiderstand fällt.

Im nächsten Schritt schalten wir die Lampe des Modells wieder ein und lassen das Programm in einer Schleife laufen:

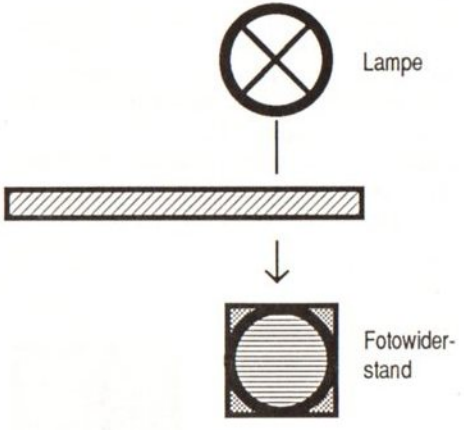


Bild 5.2: Prinzip einer Gabellichtschranke

20 CLS
30 CALL I3R
70 GOTO 50

Nach RUN wird die Wirkung des Fotowiderstandes angezeigt: am Bildschirm erscheinen einige "0", dann eine "1". Was bedeutet das? Der Fotowiderstand ist nach Bild 5.1 wie der Taster zuvor angeschlossen.
Aus den Experimenten des vorigen Kapitels wissen wir, daß ein geöffneter Taster die Anzeige einer "0" bewirkt. Wird der Taster gedrückt, wechselt die Anzeige auf "1". Dies bedeutet, daß elektrischer Strom von dem Anschluß +5V über den Schalterkontakt in den Eingang E2 fließt. Beim Fotowiderstand zeigt sich dasselbe: bei Lichteinfall leitet er den elektrischen Strom, die Anzeige ist "1". Man sagt auch, er wird niederohmig. Ohne Lichteinwirkung leitet er schlechter, die Anzeige ist "0", was der Schalterstellung "offen" entspricht. Hier redet man von einem hochohmigen Widerstand. Wir benutzen diesen Effekt, indem wir dem Fotowiderstand entweder ganz helles oder gar kein Licht zuführen. Streulicht, auch von hellen Glühlampen, kann unsere Versuche stören. Daß zunächst noch eine "0" kam, liegt daran, daß die Lampe nach dem Kommando CALL I3R noch einen kurzen Moment braucht, um die volle Helligkeit zu erreichen. Diesen Effekt werden wir in den nachfolgenden Experimenten beachten müssen und die Lampe immer eine Weile vorher einschalten.
Die Schaltwirkung der Lichtschranke, die man als Gabellichtschranke bezeichnet, können wir durch Unterbrechen des Lichtstrahls überprüfen (Bild 5.2). Halten Sie bei laufendem Programm ein dunkles Blatt zwischen Lampe und Fotowiderstand, so wechselt die Anzeige auf "0". Der Fotowiderstand sperrt - er wird hochohmig.
Er verhält sich bei großen Lichtsprüngen wie ein Taster (Schließerkontakt). Sein Vorteil liegt darin, daß er berührungslos arbeitet, reaktionsschnell ist und keine Abnutzung wie ein Taster hat. Der Nachteil ist natürlich die zusätzlich erforderliche Lichtquelle.
Mit dieser Lichtschranke wollen wir jetzt wieder den Motor steuern. Drehen Sie zunächst das Rad so, daß der Weg zwischen Lampe und LDR-Widerstand versperrt ist. Nach RUN muß das Programm "0" anzeigen. Nach Halt mit der Ctrl-Break-Taste erweitern wir das Programm mit

40 CALL I1R
60 IF E2=0 THEN GOTO 50

Lichtempfindliche Bauelemente haben in der modernen Elektronik und in fast allen Bereichen unseres Lebens weiten Eingang gefunden. Das Bauelement in unseren Experimenten wird auch LDR - light dependent resistor - genannt, was soviel wie lichtabhängiger Widerstand bedeutet. Daneben gibt es noch Photodioden, Phototransistoren und Solarzellen, die alle auch auf Licht reagieren. Diesen Bauelementen liegt zugrunde, daß in atomaren Prozessen die Lichtteilchen (Photonen) elektrische Ladungsträger (Elektronen) im Halbleitermaterial freisetzen können. Lichtempfindliche Bauelemente finden wir z.B. in Belichtungsmessern, in solarbetriebenen Uhren, in der Fernsteuerung von Fernsehgeräten, aber auch in der hochmodernen Glasfasertechnik zur Informationsübertragung.

70 CALL I1A
80 CALL I3A

und starten es mit RUN. Der Motor dreht die Scheibe jetzt solange, bis durch einen Spalt des Rades Licht auf den Fotowiderstand fällt. Motor und Lampe werden ausgeschaltet. Die Zeilen 40-70 haben wir bereits im vorigen Kapitel kennengelernt und dafür einen neuen Befehl eingesetzt:

CALL I1Z

Geben Sie folgendes neues Programm ein:

```
LOAD"INIT"
10 CALL IIN
20 FOR Z=0 TO 200
30 CALL I3R
40 NEXT Z
60 CALL I1Z
80 CALL I3A
```

Starten Sie das Programm mit RUN. Die Lampe wird in der FOR...NEXT-Schleife eine Weile vorgeheizt. Der Motor läuft anschließend solange, bis die Lampe wieder über einem Spalt in der Scheibe steht. Da sie sechs Einkerbungen hat, können wir sie mit

50 FOR I=1 TO 6
70 NEXT I

und RUN einmal komplett drehen lassen. Mit der Programmänderung:

60 CALL I1V

dreht sich das Rad in die andere Richtung. Auf der Diskette finden Sie das Programm WINKEL.BAS, das die Drehscheibe in ähnlicher Weise steuert, wie das Programm POSITION.BAS die Seilwinde. Allerdings weist das Programm eine Verfeinerung auf. Da bei der Drehscheibe nach sechs Schritten die Ausgangslage wieder erreicht wird, akzeptiert das Programm nur fünf verschiedene Positionsangaben (die Ausgangsposition ist als Zielposition nicht sinnvoll!). Die Positionen werden übrigens im Gradmaß eingegeben: 0°, 60°, 120°, 180°, 240° und 300°. 360° gibt es nicht mehr, diese Position lautet wieder 0°. Darüber hinaus sucht das Programm immer den kürzesten Weg zu der Zielposition. Von 60° auf 240° geht es also rückwärts über die 0°.

5.2. Schalten auf Distanz: Reflexionslichtschranke

Eine Variante des Versuchs ist die Reflexionslichtschranke. Hier wird das Licht nicht direkt zum LDR-Widerstand geführt, sondern von einer hellen Fläche reflektiert. Bild 5.3 zeigt den Unterschied:
Unterbrochen wird der Strahl, indem man die Reflexionsfläche entfernt. Ändern Sie das Modell Gabellichtschranke aus dem letzten Versuch in das Modell Reflexionslichtschranke aus der Bauanleitung ab. Die Lampe befindet sich jetzt ebenfalls auf der Unterseite des Rades. Sie strahlt auf die Scheibe, von der das Licht über aufgeklebte Segmentflächen reflektiert wird. Diese hellen Flächen befinden sich an denselben Stellen, an denen vorher das Licht durch das Rad gelangen konnte. Unser Programm müßte genauso laufen wie vorher. Starten Sie es mit

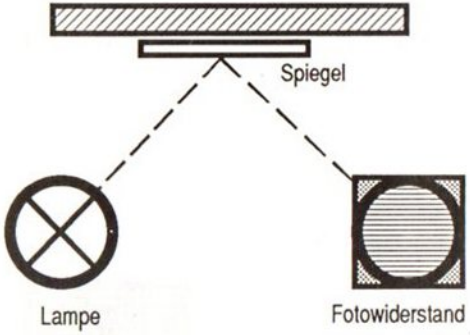


Bild 5.3: Arbeitsweise einer Reflexionslichtschranke

RUN

Das Rad dreht sich einmal ganz und bleibt dann stehen. Wenn das nicht der Fall ist, stimmt vielleicht der Abstand zwischen Rad und Lampe nicht. Exakt einstellen läßt er sich mit folgendem Programmstück, das wir dem bestehenden hinzufügen:

```
15 GOTO 200
200 CALL I3R
210 CLS
220 LOCATE 1,1
230 CALLIDE(E1,E2,E3,E4,E5,E6,E7,E8)
240 PRINT E2
250 GOTO 220
```

Drehen Sie das Rad so, daß ein helles Segment zwischen Lampe und Fotowiderstand steht, wie Bild 5.3 zeigt.
Jetzt starten Sie das Programm mit RUN. Schieben Sie das Rad auf der Achse so weit nach oben oder unten, bis die Bildschirmanzeige "1" ist. Wenn Sie nun das Rad nach rechts und links drehen, muß die Anzeige zwischen "1" und "0" wechseln. Damit ist der Abstand zwischen Lampe und Rad richtig. Mit der Ctrl-Break-Taste halten Sie das Testprogramm an. Entfernen Sie die Zeile 15. Nun können Sie das Programm erneut mit RUN starten. Diesmal erfolgt kein Sprung in den Programmabschnitt ab Zeile 200 und das Hauptprogramm wird ausgeführt.
Auch hier lassen sich wieder verschiedene Anwendungsbeispiele wie zuvor ausprobieren: man kann das Rad abwechselnd nach rechts und links laufen lassen. Geben Sie dafür ein:

```
50 FOR I=1 TO 3
60 CALL I1V
70 NEXT I
80 FOR I=1 TO 6
90 CALL I1Z
100 NEXT I
110 FOR I=1 TO 6
120 CALL I1V
130 NEXT I
140 GOTO 80
```

Lichtschranken werden in der Praxis an vielen Stellen eingesetzt: bei Alarmanlagen im Haus, in der Autowaschstraße oder als Zähler am Fließband. Auch in manchen Diskettenlaufwerken befindet sich eine Lichtschranke. Sie erkennt anhand eines Loches in der Diskette den Anfang einer Datenspur. Drehen Sie eine alte 5¼"-Diskette von Hand in der Hülle. An einer bestimmten Stelle werden Sie das Loch finden, das zum Schalten der Lichtschranke dient.

Markieren Sie einen Punkt auf dem Rad und starten das Programm mit RUN. Der Punkt wird sich um 360° hin- und herbewegen.

Natürlich kann man den Motor auch wieder mit Hilfe der Lichtschranke auf eine bestimmte Position drehen. Versuchen Sie selbst, die Grenzen der Lichtschranke festzustellen. Bei welcher Raumhelligkeit arbeitet sie noch einwandfrei?

Ist sie empfindlich genug, um den Motor immer wieder an der gleichen Stelle anzuhalten?

Es erweist sich, daß die Reflexionslichtschranke sehr sorgfältig einjustiert werden muß, um zuverlässig zu arbeiten. Die Erklärung dafür folgt im nächsten Kapitel.

6. Messen und Auswerten

6.1. Analogwerterfassung: Belichtungsmesser

Eingangsschaltungen an Computern, die analoge Werte erfassen können, werden AD-Wandler (Analog-Digital-Wandler) genannt. Die AD-Wandler arbeiten nach unterschiedlichen Verfahren, die sich im Aufwand, der Präzision und der Arbeitsgeschwindigkeit teils erheblich unterscheiden. Der AD-Wandler im fischertechnik Interface benutzt das gleiche Prinzip wie die Eingangsschaltung für stufenlose Joysticks, sog. Paddles. Je nach Widerstandswert erzeugt ein Zeitgeberbaustein Impulse entsprechender Dauer, die an einen Computereingang weitergegeben werden. Der Computer bestimmt wiederum die Impulsdauer durch ein Zählverfahren.

Wie wir aus dem letzten Versuch gesehen haben, arbeitet die Lichtschranke nicht so sicher bei der Motorsteuerung wie ein Taster. Besonders die Reflexionslichtschranke war anfällig gegen Fremdlichteinfall. Sie schaltete dadurch manchmal auch an nicht gewollten Stellen oder überhaupt nicht.
Um das genaue Verhalten des Fotowiderstandes bei Lichtänderung zu erfassen, bauen wir das Modell Belichtungsmesser aus der Bauanleitung zusammen. Sie brauchen dazu den Grundrahmen mit dem Motor nicht zu zerlegen. Die verbleibenden Bausteine genügen für den Belichtungsmesser. An der Halterung sitzt vorn der Fotowiderstand, der diesmal am Analogeingang EX (oranges Kabel) angeschlossen ist. Dieser Eingang erfaßt im Gegensatz zum Digitaleingang, den wir bei dem Versuch mit der Lichtschranke benutzt haben, analoge Meßwerte. Dies sind z.B. Spannungen, die sich zwischen einem Minimum und Maximum stetig ändern. Unser Interface ist so konstruiert, daß zwischen den Analogeingang und +5V ein veränderlicher Widerstand geschaltet werden kann. Der Widerstandswert wird in einen Zahlenwert umgesetzt, den der Computer lesen und verarbeiten kann.
Schließen Sie das Modell am Interface an und laden Sie gegebenenfalls den Interfacetreiber und BASIC (s. Kapitel 3). Zum Einlesen eines Analogwertes - hier also eines Widerstandswertes - mittels des Eingangs EX geben Sie ein:

```
LOAD"INIT"
10 CALL IIN
30 CALL IEX(EX)
40 PRINT EX
```

Mit CALL IIN wird das Interface in den Grundzustand gebracht, mit CALL IEX der Wert des Fotowiderstands im Computer in der Variablen EX gespeichert. Nach dem Programmstart mit RUN wird der Analogwert durch die PRINT-Anweisung am Bildschirm angezeigt.
Daß der Fotowiderstand auf Lichtänderungen reagiert, können wir mit folgenden Programmänderungen feststellen:

```
20 PRINT "Meßwert:";
50 A$=INKEY$
60 IF A$="" THEN GOTO 50
70 GOTO 20
```

Geben Sie die Zeilen ein und starten das Programm mit RUN. Schwenken Sie nun den Fotowiderstand hin und her, so daß er

Meßwert:	Licht
58	hell
87	
123	
174	
255	dunkel

Bild 6.1: Meßreihe einer Lichtmessung.

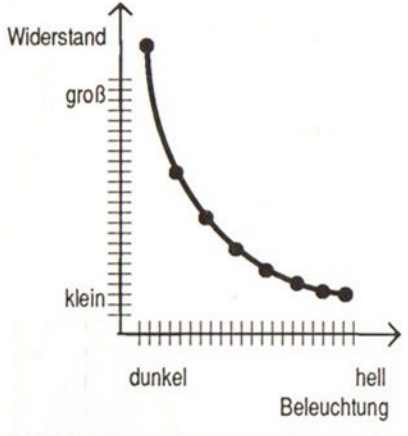


Bild 6.2: Allgemeiner Kennlinienverlauf des Fotowiderstandes.

unterschiedlich beleuchtet wird. Drücken Sie dabei immer wieder auf eine Taste des Computers; dann wird auf dem Bildschirm der nächste Meßwert angezeigt. Man erkennt deutlich, daß bei jedem Helligkeitswechsel der Zahlenwert größer oder kleiner wird. Wir haben es hier nicht mit zwei stabilen Zuständen wie beim Taster zu tun (EIN und AUS). Dadurch erklärt sich auch das unterschiedliche Schaltverhalten der Lichtschranke bei Helligkeitsschwankungen.

Den genauen Zusammenhang zwischen Lichtstärke und Widerstandswert (Meßwert) ermitteln wir durch eine Meßreihe. Wir halten den Fotowiderstand in die hellste Richtung im Zimmer (z.B. zum Fenster) und starten das Programm wieder mit RUN. Der entsprechende Meßwert wird angezeigt. Jetzt drehen wir den Fotowiderstand etwas aus dem Licht und messen durch Drücken einer Taste erneut. Auch dieser Wert wird unter dem ersten angezeigt. Wir drehen den Lichtsensor einen Schritt weiter zum Dunklen hin und messen auch hier die Lichtstärke. Das Ganze wiederholen wir noch einige Male, bis der Fotowiderstand in die dunkelste Richtung im Zimmer zeigt.

Jetzt beenden wir das Programm mit der Ctrl-Break-Taste. Die Tabelle auf dem Bildschirm sollte wie in Bild 6.1 aussehen. Hier ist noch zusätzlich die entsprechende Helligkeit angegeben.

Die Werte können bei Ihnen natürlich etwas anders sein, weil in jedem Zimmer andere Lichtverhältnisse herrschen. Wichtig ist nur die Abstufung von hell nach dunkel. Den Verlauf der Widerstandsänderung können wir am besten in einem X/Y-Koordinatenfeld erkennen (Bild 6.2).

Hier ist für jede Lichtstärke der entsprechende Meßwert aus der Tabelle als Punkt aufgetragen. Die Verbindung der Punkte ergibt eine Kurve, die sog. Kennlinie des Fotowiderstands. Man sieht, daß sie in Richtung größerer Helligkeit nichtlinear abfällt - man sagt, sie ist logarithmisch. In Datenbüchern finden wir solche Kennlinien mit genauen Lichtstärken und Widerstandswerten für den jeweiligen Fotowiderstand (Bild 6.3). Der Entwickler kann danach den richtigen Fotowiderstand für seine Schaltung, z.B. einen Belichtungsmesser, aussuchen und abgleichen.

Als praktische Anwendung wollen wir mit unserem Fotowiderstand nun auch einen Belichtungsmesser aufbauen. Wir messen die Lichtstärke im Zimmer und zeigen Sie als Balken auf dem Bildschirm an. Dabei

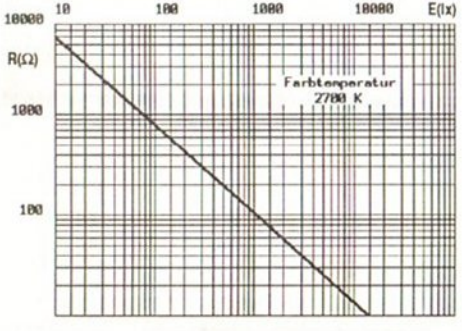


Bild 6.3: Kennlinie des fischertechnik Fotowiderstandes aus einem Datenbuch.

interessiert uns zunächst noch nicht der genaue Lichtwert in Lux oder Candela; bei uns ist ein langer Balken viel Licht, ein kurzer wenig. Geben Sie folgendes Programm ein:

```
LOAD"INIT"
10 CALL IIN
20 IF INKEY$="" THEN GOTO 20
30 CALL IEX(EX)
40 HM=EX
50 HE=EX
60 CALL IEX(EX)
70 IF EX=HE THEN GOTO 60
80 S=80*HM/HE
90 CLS
100 COLOR 0,7
110 FOR I=1 TO S
120 PRINT " ";
130 NEXT I
140 COLOR 7,0
150 GOTO 50
```

Damit die hellste Stelle im Raum einen Balken ergibt, der die ganze Bildschirmbreite ausfüllt, messen wir zunächst diese Lichtstärke. Drehen Sie den Belichtungsmesser in diese Richtung und starten das Programm mit RUN. In Zeile 20 finden wir das Programmstück zur Messung der hellsten Stelle im Raum. Wir starten den Vorgang durch Tastendruck (Zeile 20). In dieser Zeile haben wir zwei bislang getrennte Kommandos in eine Zeile geschrieben. Der Wert der INKEY$-Funktion wird nicht erst in eine Variable geschrieben, sondern gleich in der IF...THEN-Anweisung ausgewertet. Weiter geht's in Zeile 30. Dort wird die Lichtstärke gemessen und in der Variablen HM abgespeichert. Diesen Wert benötigen wir später bei jeder Anzeige als Bezugsgröße.

In Zeile 50 beginnt das Programmstück, das nun immer wiederholt wird. Der jeweils letzte Meßwert wird der Variablen HE zugewiesen; am Anfang ist er HM. Wir messen dann die Helligkeit aus der Richtung, in die der Fotowiderstand gerade zeigt, und vergleichen diesen Wert mit dem vorigen in Zeile 70. Nur wenn der Meßwert sich von dem vorigen unterscheidet, muß etwas angezeigt werden. In Zeile 80 wird die Balkenlänge errechnet. Bei halber Lichtstärke ist HE ca. doppelt so groß wie HM. Daraus ergibt sich

S = 80*1/2 =40

Der Balken hat die halbe Länge wie vorher. In Zeile 90 wird der Bildschirm gelöscht und

In der Lichtmessung gibt es eine Reihe von Maßsystemen. Jedes bezieht sich auf eine andere Fragestellung bzw. Meßanordnung:
Der Lichtstrom wird in Lumen (lm) angegeben.
Die Lichtstärke, das ist der Lichtstrom, der in eine bestimmte Richtung geschickt wird, wird in Candela (cd) gemessen.
Die Beleuchtungsstärke, das ist nun der Lichtstrom, der auf eine bestimmte Fläche einfällt, wird in Lux (lx) gemessen (1 lx = 1 lm/m²).
Um zu beurteilen, wie hell unser Auge oder aber unser Fotowiderstand etwas sieht, ist nicht nur maßgeblich, wie hell der Gegenstand beleuchtet ist, sondern auch wie nahe wir uns an dem Gegenstand befinden und wie "groß" unsere Augen sind. Dies kann als Stilb (sb) angegeben werden (1 sb = 1 cd/cm²). Ein Stilb entspricht heller Tagesbeleuchtung, das menschliche Auge kann aber noch Eindrücke von einem millionstel Stilb registrieren.

darauf S Zeichen (FOR I=1 TO S) hintereinander angezeigt. Dieser Balken besteht aus inversen Leerzeichen (s. COLOR-Anweisung). Nach der Anzeige wird wieder auf normale Farbdarstellung zurückgeschaltet. Es erfolgt eine neue Messung (GOTO 50 in Zeile 150).
Drehen Sie bei laufendem Programm den Belichtungsmesser in verschiedene Richtungen. Es wird die jeweilige Helligkeit als Balken angezeigt. Natürlich braucht das seine Zeit, so daß Sie den Fotowiderstand nicht zu schnell drehen dürfen.
Versuchen Sie selbst, die Anzeige Zeile für Zeile auszugeben um so den Verlauf der Lichtintensität anzuzeigen.
Wenn Sie die wirkliche Lichtstärke anzeigen wollen, müssen Sie den Fotowiderstand bzw. die Meßeinrichtung abgleichen. Dazu benötigen Sie u.a. die entsprechende Kennlinie des Fotowiderstandes. Welcher Zahlenwert dann zu welchem Widerstandswert gehört, könnte man mit Vergleichswiderständen ermitteln, die man anstelle des Fotowiderstandes einsetzt. Sie sehen, hier kann man noch viel experimentieren!
Bislang hatten wir immer vor dem Fotowiderstand die Abdeckkappe zusammen mit dem Röhrchen montiert. Diese Anordnung dient dazu, den Sichtwinkel des Belichtungsmessers einzuschränken, so daß er genau auf ein Objekt ausgerichtet werden kann. Eine solche Anordnung wird auch Kollimator genannt.
Manchmal stellt sich jedoch auch eine andere Meßaufgabe. Dann soll nicht die Helligkeit eines Objekts gemessen werden, sondern die Beleuchtung, die auf ein Objekt einwirkt. In diesem Fall wird der Belichtungsmesser zum Objekt gebracht und gegen die Lichtquelle(n) ausgerichtet. Gerade bei mehreren Lichtquellen muß der Belichtungsmesser den Lichteinfall von allen Seiten messen. Zu diesem Zweck ersetzen wir den Kollimator durch eine Streuscheibe in Form einer halb durchsichtigen, weißen Abdeckkappe.
Eine solche Abdeckung des Fotowiderstands wird auch Diffusor genannt.
Verwenden Sie die oben entwickelte Software oder das Programm BELICHT.BAS von der Diskette, um sich davon zu überzeugen, daß der Belichtungsmesser jetzt nicht mehr so empfindlich auf seine Ausrichtung reagiert.

6.2. Automatische Lichtmessung: Computerauge

Mit dem Belichtungsmesser aus dem letzten Kapitel konnten wir die Helligkeit in unserem Zimmer in jeder Richtung messen und am Bildschirm anzeigen. Dazu mußten wir den Fotowiderstand von Hand drehen; das Meßergebnis wurde als unterschiedlich langer Balken auf dem Monitor angezeigt.

Jetzt wollen wir diese Messung automatisch ablaufen lassen und bauen dazu das Modell Computerauge aus der Bauanleitung auf. Auf dem Grundrahmen sitzt ein Motor, der über einen Schneckenantrieb eine senkrechte Achse dreht. Oben auf der Achse befindet sich der Fotowiderstand, den wir durch den Antrieb jetzt in alle Richtungen blicken lassen können. Der Motor arbeitet im Schrittsteuerprinzip, was wir an dem Taster an der Seite des Grundrahmens erkennen können.

Zunächst testen wir die Funktion von Motor und Fotowiderstand, bevor wir mit der automatischen Lichtmessung beginnen. Geben Sie folgendes Testprogramm ein:

```
LOAD"INIT"
10 CALL IIN
20 FOR I=1 TO 10
30 CALL I1V
40 NEXT I
```

Nach RUN muß sich die senkrechte Achse um 90° drehen. Achten Sie darauf, daß sich das Kabel zum Fotowiderstand nicht verklemmt. Damit kennen wir auch schon das Verhältnis zwischen Motorschritten und Drehwinkel der Meßeinrichtung: 10 Schritte (FOR I=1 TO 10) drehen den Fotowiderstand um 90°; damit führt ein Schritt eine 9°-Drehung aus. Dieser Winkel ergibt sich aus dem Übersetzungsverhältnis. Der Befehl CALL I1V dreht das Schneckenrad um ½ Umdrehung. 1 Umdrehung des Schnekkenrades dreht das Zahnrad um 1 Zahn. Es hat 20 Zähne; damit ergibt sich:

360°/20/2 = 9°.

Eine ganze Umdrehung von 360° erreichen wir mit:

```
20 FOR I=1 TO 40
```

und RUN. Denken Sie an das Kabel zum Fotowiderstand! Im Notfall halten Sie das Programm mit der Ctrl-Break+F10-Taste an. Zurückdrehen läßt sich das Computerauge mit:

```
30 CALL I1Z
```

und RUN. In eine bestimmte Position drehen läßt sich der Lichtsensor mit:

```
LOAD"INIT"
10 CALL IIN
20 RE$=CHR$(0)+CHR$(77)
30 LI$=CHR$(0)+CHR$(75)
40 A$=INKEY$
50 IF A$="" THEN GOTO 40
60 IF A$=RE$ THEN CALL I1Z
70 IF A$=LI$ THEN CALL I1V
110 GOTO 40
```

In den Zeilen 20 und 30 werden den Variablen RE$ und LI$ die Tastaturcodes für Cursor nach rechts und Cursor nach links zugewiesen.
Starten Sie das Programm mit RUN. Jetzt können Sie die Cursortaste ← oder → betätigen. In den Zeilen 60 und 70 wird die gedrückte Taste erkannt und ein entsprechender Drehschritt ausgeführt. Das Programm läuft solange in der Schleife (GOTO 40), bis Sie die Ctrl-Break-Taste drücken.
Der Fotowiderstand ist wie zuvor am Analogeingang EX angeschlossen. Abgefragt wird er mit der Anweisung CALL IEX(EX). Der Wert der Variablen EX entspricht der Helligkeit, die das Computerauge sieht. Drehen wir den Fotowiderstand, so muß sich auch die Anzeige je nach Lichtstärke wieder ändern. Ergänzen Sie das Programm:

```
80 CALL IEX(EX)
90 CLS
100 PRINT EX
```

und starten Sie es mit RUN. Mit den beiden Cursortasten können Sie die Blickrichtung des Fotowiderstandes ändern. Die gemessene Lichtstärke wird jedesmal angezeigt. Auch eine komplette, selbständige Drehung um 360° mit Messung und Anzeige ist möglich. Geben Sie nach Betätigen der Ctrl-Break-Taste ein:

```
LOAD"INIT"
10 CALL IIN
20 R=0
30 IF INKEY$="" THEN GOTO 30
40 CLS
50 IF R=1 THEN GOTO 130
60 FOR I=1 TO 40
70 CALL I1V
80 CALL IEX(EX)
90 PRINT EX,
100 NEXT I
110 R=1
120 GOTO 30
```

```
130 FOR I=1 TO 40
140 CALL I1Z
150 CALL IEX(EX)
160 PRINT EX,
170 NEXT I
180 R=0
190 GOTO 30
```

Nach Programmstart mit RUN wartet das Programm auf eine Tastenbetätigung. Danach geht das Programm weiter bis Zeile 50. Hier wird zunächst geprüft, in welche Richtung sich der Fotowiderstand drehen soll. Wenn er sich beim ersten Mal nach rechts gedreht hat, läuft er jetzt nach links und umgekehrt. Dazu führen wir einen sog. Merker ein: die Variable R. Sie wird am Anfang auf 0 gesetzt (Zeile 20). Damit verläuft die erste Drehung nach rechts (Zeile 60-120). Danach erhält der Merker R den Wert 1 (Zeile 110). Bei der nächsten Abfrage in Zeile 50 springt das Programm nach Zeile 130. Der Fotowiderstand dreht sich zurück. Nun folgt wieder eine Rechtsdrehung, da R auf 0 gesetzt wurde (Zeile 180). Bei der Drehung werden die Meßwerte nebeneinander auf den Bildschirm geschrieben - für jeden Schritt (9°) ein Wert. Die kleinste Zahl entspricht dabei der größten Helligkeit, wie wir im letzten Kapitel gesehen haben. Auf der Diskette befindet sich ein fertiges Programm namens SCAN.BAS, das ähnlich arbeitet.

Übersichtlicher wird das Bild, wenn man den Helligkeitswert der entsprechenden Richtung zuordnet. Am besten eignet sich dazu eine zeichnerische Darstellung. Wir könnten z.B. in der jeweiligen Richtung (angefangen bei 0° oben am Schirm) die Helligkeit auftragen. Je heller, desto weiter entfernt wollen wir eine Markierung auf dem Schirm anzeichnen. Dies erfordert die Benutzung der hochauflösenden Grafik. Wir stellen Ihnen dazu ein einfaches Malinstrument zur Verfügung.

Auf der Diskette befindet sich ein Programm mit dem Namen SCANGRA.BAS, das eine grafische Darstellung der o.a. Lichtmessung erzeugt. Es benutzt das Malinstrument; und wie jenes funktioniert, erfahren wir im nächsten Kapitel.

6.3. Darstellung von Meßwerten: Computergrafik

Neben dem normalen Textbildschirm mit 25 Zeilen und 80 Zeichen pro Zeile kann unser Computer mit einem Grafik-Bildschirmadapter ausgestattet sein. In dem sogenannten CGA-Standard (color graphics adapter) wird er z.B. in 320 x 200 Bildpunkte (Pixel) aufgeteilt, von denen jeder einzeln angesteuert und in einer von vier Farben dargestellt werden kann. Damit lassen sich schon sehr gute Bilder und Grafiken erstellen.

Die Grafikausgabe wird in zwei Schritten eingeschaltet. Zuerst müssen Sie die grafische Darstellungsmöglichkeit mit der BASIC-Anweisung SCREEN 1 einschalten. Danach wird das Malinstrument aktiviert. Dies geschieht durch den Aufruf CALL IGE. Mit der ersten Anweisung wird der Bildschirm gelöscht und mit der zweiten Anweisung erscheint ein kleines Dreieck in der Mitte des Bildschirms. Wir nennen das Dreieck Grafik-Schildkröte (engl. graphics turtle). Zunächst weist die Grafik-Schildkröte nach oben und zeigt damit ihre Marschrichtung an.

Die untersten vier Zeilen des Bildschirms bleiben weiterhin für Texteingabe und Textausgabe erhalten.

Stellen Sie sich vor, diese Schildkröte wäre Ihre Hand und der Bildschirm ein Zeichenblatt. In der Hand halten Sie einen Bleistift, mit dem Sie auf der Unterlage zeichnen können. Den Stift können Sie an jede Stelle auf dem Blatt bewegen und dort Punkte oder Linien zeichnen. Genau das kann die Grafik-Schildkröte auch

Nun zu dem ersten Beispielprogramm: die Schildkröte soll eine senkrechte Linie ziehen.

```
LOAD"INIT"
10 CALL IIN
20 SCREEN 1
30 CALL IGE
40 S%=20 : CALL IGV(S%)
```

In Zeile 40 wurden zwei BASIC-Anweisungen in eine Zeile geschriebn und durch einen Doppelpunkt getrennt. Bislang hatten wir von dieser Möglichkeit noch keinen Gebrauch gemacht, um die für BASIC typischen unleserlichen Programme zu vermeiden (meist Spaghetti-Programme genannt). In der Zeile 40 handelt es sich jedoch um einen besonderen Fall, denn die beiden Anweisungen gehören eng zusammen. Nach den Regeln der BASIC-Anweisung CALL muß jeglicher Parameter, der an das Unterprogramm übergeben werden soll, in einer Variablen enthalten sein.

Wer mit der Programmiersprache LOGO vertraut ist, wird dieses Symbol sicher wiedererkennen. Die Grafik-Schildkröte wurde in dieser sehr leistungsfähigen Programmiersprache zuerst eingeführt. Die Schildkrötengrafik oder Turtlegrafik wurde aber auch in andere Programmiersprachen übernommen, so z.B. PASCAL, COMAL und jetzt auch BASIC, weil sie mit wenig mathematischem Aufwand die Erstellung von Grafiken erlaubt. Wer gern mehr mit Schildkrötengrafik experimentieren möchte, sollte sich Literatur zu LOGO besorgen.

Zusammenfassung aller Grafikbefehle:

CALL IGE
Grafik einschalten / löschen
CALL IGA
Grafik ausschalten, zurück zum Textschirm
CALL IGV(S%)
Grafik-Schildkröte um S% Schritte vor
CALL IGZ(S%)
Grafik-Schildkröte um S% Schritte zurück
CALL IGR(D%) :
Grafik-Schildkröte um D% Grad rechts schwenken
CALL IGL(D%)
Grafik-Schildkröte um D% Grad links schwenken
CALL IG1
Grafikstift einschalten
CALL IG0
Grafikstift ausschalten
CALL IGC
Grafik kopieren (Hintergrundgrafik wird angezeigt)
CALL IGS(F%)
Farbe F% (0-3) des Grafikstiftes wählen

Direkte Zahleneingaben sind nicht erlaubt. Deswegen wird in der Zeile 40 zunächst der Wert 20 der Variablen S% zugewiesen und anschließend das Unterprogramm mit der Variablen S% als Parameterangabe in Klammern aufgerufen. Der Wert 20 ist die Zahl der Schritte, die die Grafik-Schildkröte nach oben krabbeln soll.
Die Regeln der CALL-Anweisung weisen noch eine weitere Einschränkung auf: Wertangaben dürfen nur in Ganzzahlvariablen übergeben werden (Variablen vom Typ INTEGER). Dies wird durch das Prozentzeichen am Ende des Variablennamens ausgewiesen.
Bevor wir das Programm starten, wollen wir uns überlegen, wie wir uns aus dem Grafikschirm wieder zu dem vertrauten Textschirm zurückziehen. Das Malinstrument wird mit der Anweisung CALL IGA wieder ausgeschaltet. Danach schalten wir mit der Anweisung SCREEN 0 wieder in den Textschirm zurück. Dies bringt Vorteile in der Geschwindigkeit der Zeichenausgabe, also insbesondere bei der Anzeige der Programmlisten. Danach kommt die Anweisung CALL IW80, die bewirkt, daß wieder 80 Zeichen pro Zeile dargestellt werden (der Grafikschirm stellt nur 40 Zeichen pro Zeile dar). Diese Umschaltung hätte auch mit der BASIC-Anweisung WIDTH 80 durchgeführt werden können; allerdings ist das Kommando CALL IW80 eine Erweiterung, denn es funktioniert auch mit den sogenannten Monochrom-Bildschirmen nach dem Hercules-Standard.
Fügen Sie daher folgende Zeilen dem Programm hinzu:

```
160 LOCATE 22,1
170 PRINT "Weiter mit bel. Taste!"
180 IF INKEY$="" THEN GOTO 180
190 CALL IGA
200 SCREEN 0
210 CALL IW80
```

Wie Sie sehen, wurde eine Abfrage der Tastatur programmiert, die auf einen Tastendruck wartet. So haben Sie ausreichend Zeit, die Grafik zu studieren, bevor das Programm wieder auf den Textschirm zurückschaltet.
Nach RUN bewegt sich die Schildkröte nach oben und hinterläßt auf dem Bildschirm eine Linie. Dies wird durch die Anweisung CALL IGV in Zeile 40 bewirkt. Die Schildkröte geht 20 Schritte vor und zeichnet dabei eine Linie von 20 Punkten. In der nächsten Erweiterung des Programms wird der Grafikstift ausgeschaltet -

CALL IGH(F%)
Farbe F% (0-3) des Hintergrundes wählen; wird mit dem nächsten CALL IGE wirksam.
CALL IGK(GK)
aktueller Kurs der Grafik-Schildkröte nach GK speichern
CALL IGX(GX)
X-Koordinate der Grafik-Schildkröte nach GX speichern
CALL IGY(GY)
Y-Koordinate der Grafik-Schildkröte nach GY speichern
CALL IGF(GF)
Abfrage Punkt unter Grafik-Schildkröte; Punktfarbe nach GF
CALL IGLOAD(F$)
Grafikbild aus Datei F$ der Diskette laden
CALL IGSAVE(F$)
aktuelles Grafikbild auf Diskette als Datei F$ speichern
Verwandte Kommandos:
CALL IW40
Auf 40-Spalten Darstellung umschalten
CALL IW80
Auf 80-Spalten Darstellung umschalten
CALL IHE
300-Zeilen Modus der CGA-Emulation des Monochrom-Adapter einschalten.
CALL IHA
300-Zeilen-Modus der CGA-Emulation des Monochrom-Adapters ausschalten.

der Bleistift vom Papier angehoben.

```
50 CALL IG0
60 S%=10 : CALL IGV(S%)
```

Beachten Sie, daß das erste Kommando mit der Ziffer Null und nicht mit dem Buchstaben O endet.
Wenn Sie das Programm mit RUN starten, wird zunächst wieder der 20 Punkte lange Strich gezeichnet. Im nächsten Abschnitt hinterläßt die Grafik-Schildkröte keine Spur (der Grafikstift ist ja abgeschaltet). Damit kann man den Zeichenstift zu jeder Bildschirmposition führen und dort Punkte und Linien zeichnen. Wollen Sie wieder weiterzeichnen, müssen Sie den Grafikstift wieder einschalten, den Stift sozusagen aufs Papier setzen. Als nächstes wollen wir die Grafik-Schildkröte um 90° nach rechts drehen und wieder einen Strich zeichnen:

```
70 CALL IG1
80 D%=90 : CALL IGR(D%)
90 S%=20 : CALL IGV(S%)
```

Jetzt zeichnet das Programm zwei Linien auf den Bildschirm. In der nächsten Erweiterung drehen wir die Grafik-Schildkröte gegen den Uhrzeigersinn und lassen sie rückwärts fahren.

```
100 D%=45 : CALL IGL(D%)
110 S%=30 : CALL IGZ(S%)
```

Starten Sie das Programm mit RUN. Die Grafik-Schildkröte zeichnet nun zusätzlich eine diagonale Linie von 30 Punkten Länge. Sie können auch ein Viereck zeichnen. Löschen Sie die Zeilen 20 bis 110 und geben Sie ein:

```
20 SCREEN 1 : CALL IGE
30 FOR I=1 TO 4
40 S%=30 : CALL IGV(S%)
50 D%=90 : CALL IGR(D%)
60 NEXT I
70 CALL IG0
```

Bei der Eingabe werden immer die letzten vier Zeilen angezeigt. Nach RUN zeichnet die Grafik-Schildkröte ein Viereck auf den Bildschirm. Wir können auch ein größeres malen:

```
40 S%=40 : CALL IGV(S%)
```

Nach Programmstart mit RUN erscheint jetzt ein größeres Quadrat auf dem Schirm. Geben Sie

```
25 FOR K=1 TO 9
70 D%=40 : CALL IGR,40
```

80 NEXT K

ein, und es werden nach RUN mehrere Vierecke auf den Bildschirm gezeichnet, die jeweils um 40° (Zeile 70) verdreht sind. Sie sehen, wie einfach man mit der Grafik-Schildkröte malen kann.
Wenn Sie das gezeichnete Bild dauerhaft aufbewahren wollen, können Sie dazu das Kommando CALL IGSAVE zusammen mit einem Dateinamen in einer Textvariablen verwenden, z.B.:

90 F$="STERN" : CALL IGSAVE(F$)

Wenn Sie das Programm jetzt wieder starten, wird das Bild unter dem Dateinamen STERN.PIC auf der Diskette gespeichert. Die Erweiterung ".PIC" des Dateinamens weist die Datei als Bilddatei aus (PIC von picture = Bild). Die Erweiterung ".PIC" wird automatisch an den Dateinamen angefügt, Sie müssen sich darum nicht kümmern.
Wenn Sie das abgespeicherte Bild wieder einladen möchten, geben Sie das Kommando CALL IGLOAD zusammen mit dem Dateinamen STERN ein. Auch jetzt entfällt die Dateitypkennzeichnung ".PIC".
Wir wollen das das Prinzip durch ein Programm aufzeigen, das erst das Bild erzeugt, speichert, dann den Schirm löscht und nach einer kurzen Weile das Bild wieder lädt:

100 CALL IGE
110 FOR I=1 TO 2000
120 NEXT I
130 F$="STERN" : CALL IGLOAD(F$)

Wenn Sie das Programm starten, werden Sie zunächst noch nicht das geladene Bild sehen. Es steht sozusagen hinter den Kulissen. Erst wenn Sie das Programm mit der Zeile:

140 CALL IGC

erweitern, wird es sichtbar. Es wird, um es genau zu sagen, von dem unsichtbaren Ladespeicher in den sichtbaren Bildspeicher kopiert. Dies bedeutet eine große Hilfe. Stellen Sie sich vor, daß Sie eine Grafik, wie z.B. den Schirm des Computerauges, geladen haben. Auf dem Schirm werden nun Meßergebnisse eingetragen. Wenn Sie die Meßergebnisse wieder löschen wollen, brauchen Sie nicht den ganzen Bildschirm zu löschen oder das Bild wieder von Diskette zu laden. Das Kommando CALL IGC kopiert Ihnen einen frischen Radar-

schirm aus dem Ladespeicher und Sie können die nächsten Meßdaten aufzeichnen.
Auch die Farben lassen sich verändern. Sie können vier verschiedene Farben für den Zeichenstift und den Hintergrund auswählen. Bei dem IBM-PC gilt folgende Zuordnung:

Palette 1 (Standard)	Palette 0
0 : Schwarz	0 : Schwarz
1 : Cyan	1 : Grün
2 : Magenta	2 : Rot
3 : Weiß	3 : Gelb

Mit den folgenden Zeilen können Sie den Hintergrund violett einfärben (natürlich nicht auf einem Monochrom-Schirm):

```
22 F%=2 : CALL IGH(F%)
24 CALL IGE
```

Wie Sie sehen, benötigt der Wechsel der Hintergrundfarbe ein zusätzliches Einschalt- bzw. Bildlöschkommando CALL IGE um wirksam zu werden. Die Farbe des Zeichenstifts läßt sich dagegen jederzeit ändern und wird mit dem nächsten Zeichenkommando wirksam:

```
35 F%=I-1 : CALL IGS(F%)
```

Passen Sie aber auf, daß Sie der Schildkröte nicht die gleiche Farbe wie dem Hintergrund zuweisen. Wie man an dem Beispiel sieht, wären die Spuren der Schildkröte unsichtbar, obwohl der Zeichenstift eingeschaltet ist!
Die verschiedenen Stiftfarben sind sogar auf einem Monochrom-Schirm nicht ganz nutzlos, denn sie erzeugen helle, graue und schwarze Striche.
Die Zuordnung der Farben läßt sich durch das COLOR-Kommando in jene der Palette 0 ändern. Schlagen Sie dazu in dem BASIC-Handbuch Ihres Computers nach. Beim Wechseln der Palette springen jedoch immer alle Farben um, auch jene der schon früher gezeichneten Linien.
Man kann auch die aktuelle Position der Grafik-Schildkröte mit folgenden Kommandos abfragen:

```
CALL IGK(GK)
CALL IGX(GX)
CALL IGY(GY)
```

Dabei wird in der Variablen GK der momentane Kurs der Grafik-Schildkröte in Winkelgraden zur Startrichtung, in GX die X-Koordinate und in GY die Y-Koordinate der Grafik-Schildkröten-Position festgehalten.

Um diese Kommandos auszuprobieren, löschen Sie die Zeilen 22 bis 160 und geben ein:

```
30 S%=20 : CALL IGV(S%)
40 D%=90 : CALL IGR(D%)
50 S%=10 : CALL IGV(S%)
60 LOCATE 21,1
```

Die Grafik-Schildkröte hat sich nach vorn und nach rechts bewegt und zeigt auch nach rechts. Mit der nachstehenden Programmerweiterung werden die Informationen über die Grafik-Schildkröte abgerufen:

```
70 CALL IGK(GK)
80 PRINT "Schildkrötenkurs: ";GK
90 CALL IGX(GX)
100 CALL IGY(GY)
110 PRINT "Schildkrötenposition (X/Y):
      ";GX,GY
```

Auf dem Bildschirm wird der Kurs mit 90 (°) und die Position mit X=10, Y=20 ausgewiesen.
Mit dem Befehl CALL IGF(GF) fühlt die Grafik-Schildkröte ihren Weg ab. Das Kommando hinterlegt in der Variablen GF den Farbwert des Punktes, auf den die Schildkröte zuletzt bewegt wurde. Damit können z.B. früher schon einmal gezeichnete Spuren erkannt werden. Erweitern wir das Programm abermals:

```
120 CALL IGF(GF)
130 PRINT "Farbcode";GF
```

Starten Sie noch einmal unser Programm mit RUN. Auf dem Bildschirm erscheint als Farbcode die Nummer 0, was definitionsgemäß dem Hintergrund entspricht.
Das Bild können Sie auch auf Ihren Drucker bringen. Die Fachleute sagen dazu "Hardcopy". Um eine Hardcopy auszugeben, drücken Sie die Tasten "Shift" und "PrtScr" gleichzeitig. Normalerweise dient dieser Tastendruck dazu, um den Inhalt des Textschirms auf dem Drucker auszugeben. Wenn Sie jedoch vor dem Start des BASIC-Interpreters das Programm GRAPHICS des MS-DOS-Systems aufgerufen haben, ist es auch möglich, den Grafikschirm auszudrucken. Auf diese Weise können Sie Meßprotokolle Ihrer Experimente auch übersichtlich zu Papier bringen.

6.4. Messung des reflektierten Lichts: Radar

Bisher hatten wir bei der Lichtmessung Fremdlicht benutzt. Es wurden durch die Sonne oder Zimmerlampe beleuchtete Gegenstände abgetastet und deren Helligkeit angezeigt. Wenn wir nun einen Gegenstand vom Modell aus anstrahlen und die reflektierte Lichtmenge messen, könnte man daraus ableiten, wie weit der Gegenstand vom Modell entfernt ist. Vergrößern wir den Abstand zum Fotowiderstand, wird die Lichtstärke geringer. Ob sich daraus ein Radar entwickeln läßt, wollen wir mit dem folgenden Versuch ausprobieren.
Bauen Sie zunächst das Modell Radar aus der Bauanleitung zusammen. Es sieht ähnlich aus wie das Modell Computerauge, hat aber zusätzlich eine Lampe über dem Fotowiderstand. Sie ist am Anschluß M3 des Interfaces angeschlossen.
Nehmen Sie jetzt einen hellen Gegenstand, z.B. einen weißen Schuhkarton und stellen ihn 10 cm vor dem Modell nach Bild 6.4 auf. Die Lichtintensität, die von der Lampe ausgeht und durch den Karton wieder in die Fotozelle reflektiert wird, wird mit folgendem Programm gemessen:

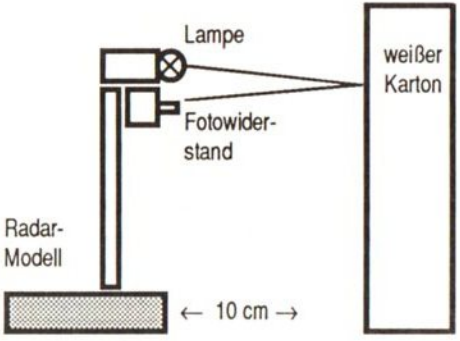


Bild 6.4: Abstandsmessung mit einem Radarmodell.

```
LOAD "INIT"
10 CALL IIN
20 CLS
30 CALL I3R
40 CALL IEX(EX)
50 PRINT EX
60 A$=INKEY$
70 CALL I3R
80 IF A$="" THEN GOTO 60
90 GOTO 20
```

Bei dem Versuch darf kein Fremdlicht auf den Karton fallen. Dunkeln Sie deshalb den Raum etwas ab. Der Anzeigewert entspricht jetzt einer Entfernung von 10 cm. Dabei liefert nicht die erste Messung das richtige Ergebnis! Die Meßeinrichtung muß sich erst einpegeln, wie Sie nach Programmänderung von

```
60 GOTO 30
```

sehen können. Durch die Trägheit von Lampe und Fotowiderstand (s. auch Kapitel 5) ist die Anzeige erst nach 10 bis 15 Messungen stabil.
Zeile 60 wird wieder richtiggestellt:

```
60 A$=INKEY$
```

Den Wert, den das Programm ab dem zweiten Tastendruck anzeigt, merken wir

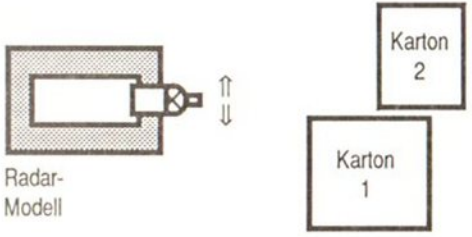


Bild 6.5: Radarabtastung mehrerer Gegenstände.

uns und vergrößern den Abstand zwischen Modell und Karton auf 20 cm. Nun wird ein Wert angezeigt, der ca. doppelt so hoch ist wie vorher. Wenn wir auf 5 cm an den Karton heranrücken, beträgt der Anzeigewert nur noch ca. die Hälfte des ersten Meßwertes bei 10 cm. Uns sollen hier keine genauen Zahlen interessieren, wichtig ist nur das Prinzip. Allerdings sollten die Zahlen im normalen Arbeitsbereich des Interface liegen, also zwischen etwa 20 und 300. Erhalten Sie bei den Versuchen immer Werte über 300, so entfernen Sie einfach die Hülse 15 aus der Störlichtkappe des Fotowiderstands. Danach sinken die Zahlenwerte, denn der Fotowiderstand erhält eine größere Lichtmenge.

Man kann somit anhand der Lichtstärke, die von einem Gegenstand reflektiert wird, eine grobe Aussage über seine Entfernung vom Meßpunkt machen. Dabei dient als Vergleichswert eine bekannte Entfernung und die dazugehörende Lichtstärke. Wenn man mehrere Stellen abtastet, ist es wichtig, daß alle den gleichen Farbton (Grauwert) haben und kein Fremdlicht auf sie fällt. Wir wollen das jetzt mit zwei Kartons, die nach Bild 6.5 aufgestellt sind, ausprobieren. Dabei wollen wir auch unsere Radaranlage schwenken, wie wir es schon vorher mit dem Computerauge durchgeführt haben. Es ist noch dasselbe Programm, nur die Auswertung wird abgeändert:

```
LOAD"INIT"
10 CALL IIN
20 SCREEN 1 : CALL IGE
30 FOR I=1 TO 100
40 CALL I3R
50 NEXT I
60 FOR I=1 TO 40
70 CALL I1V
80 D%=9 : CALL IGL(D%)
90 CALL IEX(EX)
100 CALL IG0
110 S%=EX/4 : CALL IGV(S%)
120 CALL IG1
130 D%=90 : CALL IGR(D%)
140 S%=5 : CALL IGV(S%)
150 S%=5 : CALL IGZ(S%)
160 CALL IG0
170 D%=90 : CALL IGL(D%)
180 S%=EX/4 : CALL IGZ(S%)
190 NEXT I
200 FOR I=1 TO 40
210 CALL I1Z
220 NEXT I
230 PRINT "Ende mit beliebiger Taste!"
240 A$=INKEY$
```

```
250 IF A$="" THEN GOTO 240
260 CALL IGA : SCREEN 0 : CALL IW80
270 CALL I3A
280 END
```

Die gemessene Helligkeit wird in diesem Programm als Bewegungsmaß für die Schildkröte umgesetzt. Wie wir schon gesehen hatten, fällt umso weniger Licht auf den Fotowiderstand zurück, je weiter der Gegenstand entfernt steht. Umso größer wird der Analogwert EX ermittelt. Wir lassen also die Schildkröte umso weiter aus der Bildschirmmitte vorschreiten, je höher der Analogwert EX liegt. Dies erfolgt mit abgeschaltetem Schreibstift. Anschließend malt die Schildkröte einen kleinen Balken auf den Schirm und springt in ihre Ausgangslage zurück. Für den nächsten Meßpunkt dreht sich die Radaranlage um 9°. Um den gleichen Betrag lassen wir auch die Schildkröte drehen.
Ein ähnliches Programm ist RADAR.BAS, das Sie von der Diskette laden können.

Dieser Versuch zeigt sehr anschaulich das Grundprinzip eines Radargerätes. Nur arbeiten diese Geräte nicht mit Licht, sondern mit Funkwellen. Dies sind etwas längere elektromagnetische Wellen, aber im Prinzip die gleiche Wellen wie die des Lichts.
Auch das Meßprinzip unterscheidet sich. Während in unserem Versuch der Rückgang der Helligkeit mit der Entfernung ausgenutzt wird, mißt ein Radargerät die Laufzeit der Wellen vom Senden bis zum erneuten Eintreffen. Funkwellen breiten sich genauso wie Lichtwellen mit 300 000 km pro Sekunde aus. Die Laufzeit ist also wegen der hohen Geschwindigkeit meist sehr kurz, kann aber durch geeignete elektronische Schaltungen zuverlässig bestimmt werden.

Jetzt kommen wir zu einer weiteren physikalischen Größe: der Temperatur. Wir werden sie messen und in elektrische Werte umsetzen, damit unser Computer sie versteht. Und dann befassen wir uns mit der Temperaturregelung. Sie begegnet uns heute überall: z.B. im Kühlschrank, der eine eingestellte Temperatur beibehält, beim Heizlüfter, der für eine konstante Raumtemperatur sorgt, oder beim Kühlgebläse im Auto, das darauf achtet, daß der Motor nicht zu heiß wird.

Für die Messung der Temperatur benutzen wir einen temperaturabhängigen Widerstand - Fachleute nennen ihn NTC-Widerstand oder Heißleiter.

Sein Wert ändert sich also, wenn man die Umgebungstemperatur erhöht oder erniedrigt. Legt man an diesen Widerstand eine konstante Spannung, so stellt sich je nach Temperatur ein unterschiedlicher Strom ein.

Wie schön die Temperaturmessung mit einem Heißleiter funktioniert, wollen wir im folgenden Versuch ausprobieren. Dazu bauen wir das Modell Thermometer aus der Bauanleitung zusammen. Der Widerstand wird am Interface zwischen +5V (grünes Kabel) und den Analogeingang EY (gelbes Kabel) geschaltet. Der Eingang EX, den wir natürlich auch hätten benutzen können, wird zur Vermeidung von Störeffekten stillgelegt (Brücke von EX nach +5V). Wenn alles richtig angeschlossen ist, kann die erste Messung beginnen.

Geben Sie in den Computer ein:

```
LOAD"INIT"
10 CALL IIN
40 CALL IEY(EY)
50 PRINT EY
```

Nach dem Start des Programms sollte auf dem Bildschirm eine Zahl etwa um 100 erscheinen. Wenn das der Fall ist, haben Sie zum ersten Mal eine Temperatur elektronisch per Computer gemessen.

Sollte eine Fehlermeldung erscheinen, so überprüfen Sie, ob der Interface-Treiber geladen ist.

Nun aber zu unserem Meßergebnis. Die Zahl, die wir auf dem Bildschirm ablesen, entspricht der Umgebungstemperatur. Daß sich der NTC-Widerstand auch wirklich mit der Temperatur ändert, können wir durch folgenden Test beweisen.

Geben Sie die Programmzeilen ein:

```
20 CLS
30 LOCATE 22,1
```

NTC kommt vom Englischen Negative Temperature Coefficient und bedeutet negativer Temperatur-Koeffizient. Und was das heißt, sagt das deutsche Wort Heißleiter sehr anschaulich: der Widerstand leitet umso mehr, je heißer er wird.

Ein Heißleiter besteht aus einem keramischen Material. Ausgangsprodukt sind die Oxide von Mangan, Eisen, Kobalt, Nickel, Kupfer und Zink, die eine starke Abhängigkeit ihrer Leitfähigkeit von der Temperatur aufweisen.

```
50 PRINT USING"###";EY
60 GOTO 30
```

und starten Sie das Programm mit RUN.
Mit Zeile 20 wird der Bildschirm gelöscht. Zeile 30 setzt die Ausdruckposition an den unteren Rand des Bildschirms, um bereits jetzt Platz für eine Bildschirmgrafik zu lassen. Zeile 40 liest den Wert am Analogeingang EY - also den Widerstandswert unseres Heißleiters - ein, und Zeile 50 zeigt diesen Wert am Bildschirm an. Danach springt das Programm wieder nach Zeile 30. In dem Programm ist vermieden, auf den Befehl zum Bildschirmlöschen zurückzuspringen, dies hätte eine flackernde Bildschirmanzeige ergeben. Stattdessen wird eine formatierte Bildschirmausgabe benutzt, so daß das bisherige Resultat überschrieben wird. Die drei Zeichen "#" in der Formatangabe erzwingen die Benutzung von drei Spalten am Bildschirm. So brauchen wir nicht zu berücksichtigen, daß die Anzeige mal zweistellig, mal dreistellig ist. Nach RUN wird fortlaufend der aktuelle Temperaturwert gemessen und angezeigt. Der Zahlenwert ändert sich, wie Sie sehen, allerdings nicht sehr schnell. Klar, unsere Raumtemperatur ist ja konstant. Fassen Sie nun einfach mal den NTC-Widerstand zwischen Daumen und Zeigefinger fest an, damit er sich auf Ihre Körpertemperatur erwärmt. Der Anzeigewert ändert sich nach kurzer Zeit: der Zahlenwert wird kleiner. Nach Loslassen des Heißleiters erreicht der Meßwert allmählich wieder den alten Wert, nämlich die Zimmertemperatur. Beenden Sie jetzt das Programm mit der Ctrl-Break-Taste.
Wie der Versuch zeigt, reagiert der NTC-Widerstand auf die Umgebungstemperatur - nur entspricht der Anzeigewert bei weitem nicht der wahren Temperatur in Grad Celsius. Wie kommt das?
Das Interface zwischen Heißleiter und Computer, das die Widerstandswerte in digitale, für den Computer verständliche Signale umwandelt, ist nicht kalibriert. Da das Kalibrieren des Interface aber auch viel zu kompliziert wäre - da müßten schon Elektroniker ran -, lassen wir den Computer mit Hilfe eines geeigneten Programms einfach die Umrechnung vornehmen; er soll die ursprünglichen Werte in Grad Celsius umrechnen. Der Fachmann spricht hier von einer Software-Lösung. Hätten wir das Interface umgebaut, wäre dies eine Hardware-Änderung gewesen.
Doch nun zur Kalibrierung selbst. Dazu

Temperatur (Grad Celsius)	Wert EY
0	207
5	175
10	148
15	126
20	106
25	90
30	76
35	65
40	55
45	46
50	39

Bild 7.1: Meßwerttabelle der Temperatur, gemessen mit Thermometer und mit NTC-Widerstand.

brauchen wir ein mit Wasser gefülltes Gefäß, in das wir den NTC-Widerstand und ein Haushaltsthermometer tauchen. Wir lassen das oben abgedruckte Programm laufen und notieren Bildschirmanzeige und Temperaturwert des Thermometers.
Bevor wir jedoch mit dem Versuch starten, muß der Heißleiter noch in eine Plastiktüte eingepackt werden, damit durch das Wasser kein Kurzschluß entsteht. Mit in die Tüte stecken wir das Thermometer, um für beide gleiche Meßbedingungen zu haben. Das Bild in der Bauanleitung zeigt den Aufbau der Abgleichanordnung. Achten Sie bei dem Versuch unbedingt darauf, daß keine Wasserspritzer an Ihren Computer kommen - sie könnten einen Kurzschluß auslösen. Bauen Sie daher das Gefäß möglichst weit entfernt von Ihrem Computer auf. Stellen Sie das Gefäß noch einmal in eine Schale, die bei Umkippen des Gefäßes das Wasser auffangen kann. Halten Sie Handtücher und Fließpapier bereit.
Zu Beginn füllen wir das Gefäß halb mit Wasser und tun ein paar Eiswürfel aus dem Kühlschrank mit hinein. Dadurch wird das Wasser bis an die 0-Grad-Grenze abgekühlt. Starten Sie nun das Programm mit RUN und notieren sich Thermometer- und Bildschirmwert auf einem Zettel. Jetzt erwärmen Sie das Wasser, indem Sie etwas heißes Wasser dazu gießen (Vorsicht, daß nichts überläuft!). Wenn sich die Temperatur stabilisiert hat, messen und notieren Sie die neuen Werte. Die Messungen führen Sie solange fort, bis das Wasser etwa 50 Grad erreicht hat.
Beenden Sie nun das Programm mit der Ctrl-Break-Taste. Sie haben jetzt eine Meßreihe vorliegen, die den Meßwert des NTC-Widerstandes in Abhängigkeit von der Temperatur zeigt. Sie sollte wie in Bild 7.1 aussehen. Leichte Abweichungen sind zulässig, denn die Umwandlung des Widerstandswertes in einen Zahlenwert erfolgt bei den verschiedenen Computern (XT, AT usw.) nur nach einer groben Einteilung.
Nehmen Sie den NTC-Widerstand wieder aus dem Gefäß und lassen ihn auf Zimmertemperatur abkühlen. Nach erneutem Programmstart (RUN) wird ein Zahlenwert angezeigt, den der Computer in den richtigen Temperaturwert umrechnen soll. Dazu benutzen wir die zuvor ermittelte Meßwerttabelle. Ein Anzeigewert von 90 entspricht 25 Grad Celsius, ein Anzeigewert von 55 entspricht 40 Grad Celsius, und ein Anzeigewert von 207 entspricht 0 Grad Celsius. Wie wir sehen, sind die beiden Zahlenreihen gegenläufig, also zum höchsten

In Zeile 15 haben wir nun berücksichtigt, daß der Temperaturkoeffizient des Heißleiters negativ ist. Wir erinnern uns: Negative Temperature Coefficient - je höher die Temperatur, desto kleiner der Meßwert. Daß zur Umrechnung der Logarithmus benutzt wird, liegt daran, daß die Widerstandskurve mit einer Exponentialfunktion gegen Null sinkt:

$R=R_N*e^{B/T}$

R ist der Widerstandswert und T ist die absolute Temperatur in Kelvin (s.nächste Seite); B und R_N in dieser Formel sind Materialkonstanten. Das Symbol e bezeichnet die Exponentialfunktion.
Wenn Sie ein guter Mathematiker sind, werden Sie herausfinden, daß die Kalibrationsformel nicht exakt ist aber eine ganz gute Näherung darstellt.

Temperaturwert gehört der niedrigste EY-Wert und umgekehrt.
Dazu kommt, daß - wie der Fachmann sagt - die Kennlinie des NTC-Widerstandes nicht linear ist. Anschaulich wird das, wenn wir die Kennlinie in einem X/Y-Koordinatenfeld aufzeichnen. Versuchen Sie's doch einmal! Sie werden sehen, daß die Kennlinie keine Gerade ist.
Man könnte sich nun durch Einzelversuche an die richtige Gleichung für die Umrechnung herantasten - wir wollen Ihnen jedoch gleich die Lösung verraten. Geben Sie ein:

```
15 DEF FNT(X)=160-30*LOG(X)
50 PRINT "Temperatur= ";FNT(EY);
     "Grad Celsius"
```

und starten Sie das Programm mit RUN. In der eingefügten Zeile 15 wird eine Umrechnungsformel von Analogwerten in °C definiert (DEF FNT...). Die Funktion LOG wird darin benutzt; sie ist der natürliche Logarithmus. In Zeile 40 wird nun nicht EY, sondern der umgerechnete Temperaturwert ausgedruckt. Jetzt stimmt die Anzeige schon besser.
Aber auch diese Formel muß noch nicht ganz genau sein. Wegen der obengenannten Unterschiede von Computer zu Computer sollten Sie sich Ihre ganz individuelle Umrechnungsfunktion ermitteln. Verwenden Sie das Programm KALIBR.BAS von der Diskette und geben Sie die Tabellenwerte ein (°C und Analogwert EY), so wie sie bei Ihrem Versuch ermittelt wurden. Das Programm errechnet die am besten passende Umrechnungsfunktion und druckt Sie auf dem Bildschirm aus. Notieren Sie sich die Funktion. Sie können Sie überall, bei den im Experimentierhandbuch beschriebenen Versuchen und bei den Programmen auf Diskette, anstelle von

```
DEF FNT(X)=160-30*LOG(X)
```

verwenden.
Wir haben damit ein elektronisches Thermometer mit Computeranzeige gebaut, das uns die Umgebungstemperatur in Grad Celsius anzeigt. Wir könnten dieses Modell ohne weiteres als Thermometer in einer Wetterstation benutzen. Und wenn Sie dann noch die Uhrzeit mitanzeigen und das Ganze vielleicht noch fortlaufend ausdrukken, dann haben Sie einen Temperaturschreiber, mit dem Sie z.B. Ihre Heizung kontrollieren können. Doch soweit wollen wir hier nicht gehen.

Stattdessen wollen wir Ihnen noch zeigen, wie Sie den Temperaturverlauf grafisch auf dem Bildschirm anzeigen können. Die notwendigen Grafikbefehle hatten wir ja schon in Kapitel 6 kennengelernt. Hierzu werden wir zwei Unterprogramme anhängen, die wir im Hauptprogramm aufrufen:

```
27 GOSUB 300
58 GOSUB 200
```

Unterprogramme werden immer dann verwendet, wenn bestimmte Programmsequenzen mehrmals benötigt werden. Unterprogramme sind auch dann sinnvoll, wenn man sie in Zusammenhang mit bestimmten logisch abgeschlossen Programmabschnitten bringt, wie es hier der Fall ist. Das Unterprogramm ab Zeile 300 erfüllt die Funktion "Bildschirm löschen", das ab Zeile 200 die Funktion "Meßwert zeichnen".

```
200 CALL IGX(GX)
210 IF GX=159 THEN GOSUB 300
220 S%=FNT(EY) : CALL IGV(S%)
230 CALL IG1
240 S%=1 : CALL IGV(S%)
250 CALL IG0
260 S%=FNT(EY)+1 : CALL IGV(S%)
270 D%=90 : CALL IGR(D%)
280 S%=1 : CALL IGV(S%)
290 D%=90 : CALL IGL(D%)
295 RETURN

300 SCREEN 1 : CALL IGE
310 CALL IG0
320 S%=82 : CALL IGZ(S%)
330 D%=90 : CALL IGL(D%)
340 S%=159 : CALL IGV(S%)
350 D%=90 : CALL IGR(D%)
360 RETURN
```

Das Unterprogramm "Meßwert zeichnen" läßt die Grafik-Schildkröte um den Temperaturwert bei abgeschaltetem Stift nach oben fahren, schaltet den Stift ein und läßt die Schildkröte noch eins vorgehen. Damit erscheint auf dem Bildschirm ein Punkt, der um so viele Bildpunkte von der Unterkante entfernt ist, wie der Temperatur entspricht. Die Grafik-Schildkröte wird dann wieder an den unteren Bildschirmrand zurückgezogen, nach rechts gedreht und in die nächste Spalte gestellt. Dann wird sie wieder nach links gedreht. Anschließend ist die Grafik-Schildkröte bereit für den nächsten Aufruf. Wenn die Bildschirmseite gefüllt ist, und auch vor dem ersten Verwenden muß jedoch der Bildschirm gelöscht werden. Hier-

Die Celsius-Skala der Temperatur (°C) ist nach dem Schmelzpunkt und dem Siedepunkt des Wassers ausgerichtet. Der Temperaturbereich dazwischen wurde in 100 gleiche Abschnitte, je ein Grad Celsius, eingeteilt. Die Fahrenheit-Skala (°F) benutzt andere Orientierungspunkte. Als 100°F wurde die Bluttemperatur des Menschen festgesetzt; als 0°F die tiefste Temperatur, die zu jener Zeit mit einem Salz-Wasser-Gemisch erzielt werden konnte. Die Kelvin-Skala (K - ohne Gradzeichen!) ist jüngeren Datums. Nachdem festgestellt wurde, daß es einen absoluten Tiefstpunkt der Temperatur gibt, -273,15 °C, wurde dieser als Null Kelvin bezeichnet. Die Temperaturschritte sind die gleichen wie bei der Celsius-Skala.
Die Umrechnungsformeln für die entsprechenden Temperaturskalen sind:

0 K = -273,15 °C

32...212 °F = 0...100 °C
oder - mathematisch exakt -

C = 5/9 (F - 32)

wobei C = Temperatur in °C und F = Temperatur in °F bedeutet.

zu dient das Unterprogramm ab Zeile 300. Nach Einschalten der Grafik wird die Schildkröte in die linke untere Ecke des Bildschirms rangiert. Übrigens: die Zahlenwerte 82 und 159 betreffen die Bildschirmabmessungen. Sie stellen die Fahrstrecken der Grafik-Schildkröte dar, um von der Bildmitte in die linke untere Ecke zu gelangen. Das Programm wird mit der Ctrl-Break-Taste angehalten. Damit bleibt jedoch der Grafik-Bildschirm eingeschaltet. Geben Sie deshalb im Direktmodus folgende Kommandos ein:

```
CALL IGA
SCREEN 0
CALL IW80
```

Die Temperaturanzeige läßt sich noch verdeutlichen. Bei unseren Experimenten können wir sicherlich auf den Temperaturbereich von 0°C bis 20°C verzichten und den Bereich zwischen 20°C und 40°C auf die Bildschirmhöhe spreizen. Dazu müssen Sie folgende Zeilen ändern:

```
220 S%=(FNT(EY)-20)*6 : CALL IGV(S%)
260 S%=(FNT(EY)-20)*6+1 : CALL
        IGZ(S%)
```

Wir werden unser Programm jetzt noch international anpassen, damit wir auch unserem englischen Freund mitteilen können, wieviel Grad Fahrenheit bei uns herrschen. Wissenschaftlich Interessierten liefern wir auch gleich noch die Angabe in Kelvin mit.
Wir erweitern unser Programm, so daß nach jeder Messung die Temperatur gleichzeitig in °C, Kelvin und °F auf dem Bildschirm angezeigt wird:

```
42 PRINT"Temperatur= ";FNT(EY)
       +273.15;"Kelvin"
44 PRINT"Temperatur= ";FNT(EY)
       *1.8+32;"Grad Fahrenheit"
```

Nach Start mit RUN wird wieder die aktuelle Temperatur, nun gleich dreifach, angezeigt.

7.2. Steuerung der Wärmezufuhr: Heizungsregelung

Im letzten Kapitel haben wir gesehen, wie man mit dem Heißleiter Temperaturen messen und mit dem Computer anzeigen kann. Der Computer kann aber noch mehr: Man kann ihn auch zur Steuerung der Wärmezufuhr benutzen. Die Änderungen der Temperatur sollen dabei möglichst gering bleiben. In der Praxis ist dies nichts anderes als eine Heizungsregelung. Damit lernen wir auch gleichzeitig das verstehen, was die Experten als Regelkreis bezeichnen.

Für den Versuch bauen wir das Modell Ofen aus der Bauanleitung zusammen. Das Modell verbinden wir mit dem Interface - zuvor bitte genau prüfen, ob alles richtig verdrahtet ist und nicht etwa ein Kurzschluß vorliegt.

Sehen wir uns das Modell etwas genauer an. Es besteht aus einer Lampe mit dem darüber liegenden NTC-Widerstand. Das Lämpchen dient hier als "Brenner" für die Heizung. Bitte wundern Sie sich nicht, denn solch eine Lampe strahlt nicht nur Licht aus, sondern auch Wärme. Wer's nicht glaubt, kann ja einmal eine Glühbirne in der Stehlampe zuhause anfassen, die einige Zeit eingeschaltet war. Aber vorsichtig, bitte!

Der NTC-Widerstand dient als Temperaturfühler für die vom Brenner erzeugte Wärme. Die Temperatur soll auf einem bestimmten Wert konstant gehalten werden. Dies erreichen wir über einen Regelkreis: wir geben eine Solltemperatur vor und messen die vorhandene Isttemperatur am Brenner. Erreicht die Brennertemperatur den Sollwert - das meldet der Heißleiter -, soll der Brenner ausschalten. Wird die Temperatur nach Abkühlung unterschritten, soll er wieder einschalten.

Dies wollen wir jetzt ausprobieren. Die Lampe ist am Interface mit dem Motoranschluß M3 verbunden. Eingeschaltet wird sie - wie wir in einem früheren Versuch schon gelernt haben - mit dem Befehl CALL I3L bzw. CALL I3R, ausgeschaltet mit CALL I3A. Der Heißleiter ist mit dem Analogeingang EY verbunden und wird also mit CALL IEY(EY) abgefragt. Geben Sie Folgendes ein:

```
LOAD"INIT"
10 CALL IIN
12 CLS
15 DEF FNT(X)=160-30*LOG(X)
30 LOCATE 22,1
40 CALL IEY(EY)
50 TE=FNT(EY)
60 PRINT TE
```

Diese Programmzeilen kennen wir schon; mit Zeile 10 wird das Interface in den Grundzustand gebracht (initialisiert). Die Umrechnungsfunktion in °C, die Sie natürlich wieder durch Ihre bessere, individuelle ersetzen sollten, wird in Zeile 15 definiert. Die Zeilen 40 bis 60 zeigen uns die Temperatur am Heißleiter an, die in der Variablen TE gespeichert wird.
Jetzt geben wir eine Temperatur vor, bei der der Brenner abschalten soll (Sollwert), beispielsweise 30 °C:

20 TS=30

Natürlich müssen wir jetzt den tatsächlichen Temperaturwert (Istwert) nach jeder Messung mit dem Sollwert vergleichen. Geben Sie dazu noch ein:

70 IF TE<TS THEN CALL I3R
80 IF TE>TS THEN CALL I3A
90 GOTO 30

Wenn die Isttemperatur TE kleiner als die Solltemperatur TS ist, wird der Brenner eingeschaltet (CALL I3R). Dies geschieht in Zeile 70. Hat der Istwert den Sollwert überschritten, wird der Brenner mit CALL I3A ausgeschaltet (Zeile 80). Danach springt das Programm wieder nach Zeile 30 - es folgt eine erneute Temperaturmessung.
Und nun zur Praxis: Starten Sie das Programm mit RUN. Die Lampe schaltet ein, was ja richtig ist, denn zunächst ist die Temperatur am NTC-Widerstand kleiner als 30 °C (Sollwert). Jetzt tut sich scheinbar nichts mehr. Es kann höchstens sein, daß die Lampe blinkt. Stören Sie sich nicht daran, es liegt daran, daß manchmal die Berechnungen etwas Zeit brauchen und das Interface zwischendurch die Ausgänge abschaltet (s. auch die Beschreibung der Schutzschaltung in Kap. 4). Zunächst einmal muß der Brenner den Heißleiter aufheizen, und das braucht seine Zeit. Irgendwann schaltet die Lampe aus: die Solltemperatur ist erreicht. (Wenn sich nichts tun sollte, ist vielleicht der NTC-Widerstand zu weit von der Lampe entfernt.)
Der Brenner heizt nun nicht mehr, so daß der Meßfühler abkühlt. Nach kurzer Zeit schaltet der Brenner wieder ein, und der Vorgang beginnt erneut.
Dieses Ein- und Ausschalten wiederholt sich nun laufend: der Regler schaltet zwischen zwei Punkten den Brenner: beim Überschreiten der Solltemperatur aus und beim Unterschreiten wieder ein. Man nennt

einen solchen Regler auch ganz folgerichtig einen Zweipunktregler. Sein Verhalten erkennt man am besten aus der Regelkurve in Bild 7.2. Die Regelkurven können Sie sich selbst auch auf Ihren Bildschirm holen, indem Sie die Unterprogramme aus Kapitel 7.1 zur Anzeige der Temperatur (ab Zeile 200) und zum Löschen des Bildschirms (ab Zeile 300) hinzufügen. Die Unterprogramme werden folgendermaßen aufgerufen:

28 GOSUB 300

65 GOSUB 200

Die Isttemperatur schwankt zickzackförmig um die Solltemperatur. Die Zeit zwischen "Brenner ein" und "Brenner aus" nennt man Hysterese. Diesen Effekt wollen wir noch etwas genauer untersuchen. Nach Anhalten des Programms mit der Ctrl-Break-Taste ändern Sie folgende Zeilen ab:

25 OT=TS+2
26 UT=TS-2
70 IF TE<UT THEN CALL I3R
80 IF TE>OT THEN CALL I3A

und starten das Programm wieder mit RUN. Was stellen wir fest? Richtig! Die Zeit zwischen "EIN" und "AUS", das Hystereseintervall ist größer geworden, da jetzt erst über 32 °C (OT) ausgeschaltet und unter 28 °C (UT) wieder eingeschaltet wird. Merken Sie sich nun die Anzahl der Schaltintervalle z.B. in 5 min. Jetzt erhöhen wir die Solltemperatur auf 35 °C:

20 TS=35

und starten wieder mit RUN. Wieviel Schaltimpulse zählen Sie diesmal in derselben Zeit? Es sind mehr als vorhin, und warum?
Der Heißleiter kühlt sich bei gleicher Umgebungstemperatur schneller ab als vorher. Versuchen Sie es doch einmal mit 10 °C Solltemperatur (20 TS=10). Aber warten Sie nicht zulange. Oder sitzen Sie gerade in einem solch kühlen Raum?
Sie sehen, hier kann man noch viel experimentieren. Das Modell hat uns gezeigt, wie man durch Steuerung der Wärmeerzeugung die Temperatur regeln kann. Wir verstehen jetzt, welche Aufgabe bei unserer Heizung zuhause das Raumthermostat hat und warum die Heizung bei kaltem Wetter öfter anspringt.
Auch zu diesem Modell finden Sie wieder ein Musterprogramm auf der Diskette, mit

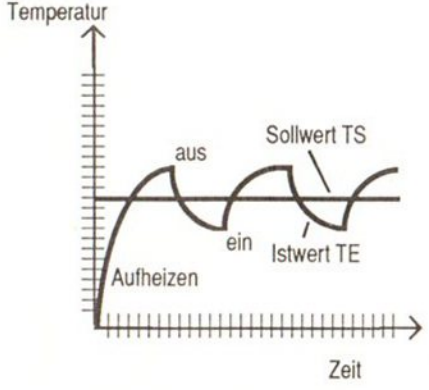


Bild 7.2: Regelkurve eines Zweipunktreglers

Regler, wie wir sie hier in unserem Experiment kennengelernt haben, werden in einer Vielzahl von technischen Prozessen eingesetzt. Dabei handelt es sich beileibe nicht nur um die Temperatur. Das kann auch genausogut die Ausgangsspannung eines Netzteils, die Benzinzufuhr zum Autovergaser, der Druck in einer Dampfmaschine (der historisch erste technische Regler) oder die Aktivität eines Kernreaktors sein. Und es geht noch weiter: auch biologische Prozesse unterliegen einem Regelmechanismus, damit "die Bäume nicht in den Himmel wachsen". Selbst auf gesellschaftliche Phänomene kann man die Formeln der Regeltechnik anwenden.

dem der Regelvorgang verständlich dargestellt wird. Sein Name ist OFEN.BAS. Vergleichen Sie auch hier die Zeilen des Hauptprogramms mit unseren Programmzeilen.

7.3. Steuerung der Kühlung: Gebläse

Statt über die Wärmezufuhr kann man natürlich die Temperatur auch über die Kühlung regeln. Was man jeweils macht, hängt von Soll- und Isttemperatur und im Normalfall von der Lufttemperatur ab. Das bedeutet aber auch, daß es Regler gibt, die sowohl heizen als auch kühlen müssen. Beispiel: Klimaanlagen; im Winter arbeiten sie als Heizung, im Sommer als Kühlung.
Wir wollen uns jetzt mit der Temperaturregelung durch Steuerung der Kühlung befassen und lassen dazu unsere Heizung, die Glühlampe, in Betrieb. Kühlen kann man - wenn es sich um höhere Temperaturen handelt - sehr gut mit ganz gewöhnlicher Luft. Und damit das schneller und wirkungsvoller funktioniert, unterstützt man die Luftzufuhr mit einem sogenannten Kühlgebläse.
Dieses Prinzip wird oftmals dann benutzt, wenn sich die Wärmequelle nicht abschalten läßt. So kann man z.B. einen Automotor während der Fahrt auch nicht einfach abschalten, wenn er zu warm wird. Oder wollen Sie alle paar hundert Meter anhalten und etliche Minuten warten, bis Sie mit abgekühltem Motor wieder weiterfahren dürfen?
Mit einem Gebläse läßt sich jedoch die

Temperatur bei laufendem Motor regeln. Wie das geht, soll der folgende Versuch zeigen. Wir bauen dazu das Modell Gebläse der Bauanleitung auf. Über der Lampe als Wärmequelle befindet sich wieder der NTC-Widerstand für die Temperaturmessung. Davor ist ein Kühlgebläse angebracht. Verbinden Sie das Modell mit dem Interface und prüfen zunächst durch Einzeltests, ob alles richtig verdrahtet ist. Geben Sie bitte ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL I1R
```

Nach Programmstart muß der Lüfter ca. ½ Sekunde laufen. Geben Sie jetzt ein:

```
20 CALL I1A
30 CALL I3R
```

Diesmal muß die Lampe kurz aufleuchten. Und nun der letzte Test:

```
20 CALL I3A
30 CALL IEY(EY)
40 PRINT EY
```

Nach dem Start des Programms wird ein Wert von etwa 100 auf dem Bildschirm angezeigt. Wenn das alles der Fall ist, können wir mit dem Versuch beginnen.
Die Regelung soll folgendermaßen ablaufen: Die Lampe als Heizquelle brennt dauernd. Der Heißleiter mißt die Temperatur und schaltet das Gebläse ein, wenn die Solltemperatur überschritten wird. Geben Sie zunächst ein:

```
LOAD"INIT"
10 CALL IIN
15 DEF FNT(X)=160-30*LOG(X)
40 CALL I3L
50 CALL IEY(EY)
60 TE=FNT(EY)
90 GOTO 40
```

Damit sind die ersten beiden Bedingungen erfüllt. Die Lampe brennt dauernd (Zeile 40), und die Temperatur wird gemessen (Zeile 50 -60). Sie ist in der Variablen TE abgespeichert. Nun kommt der Soll-/Istwert-Vergleich für den Regelkreis:

```
20 OT=30 : UT=25
70 IF TE>OT THEN CALL I1R
80 IF TE<UT THEN CALL I1A
```

Starten Sie nun das Programm mit RUN.

Die Lampe brennt und wärmt den Heißleiter auf. Das braucht zunächst seine Zeit. Wird die obere Grenztemperatur (OT) überschritten, schaltet der Lüfter ein (Zeile 70). Er kühlt den Heißleiter ab, bis die untere Grenztemperatur (UT) unterschritten wird. Hier schaltet das Gebläse wieder aus, und der Kreislauf beginnt von neuem: erwärmen - Lüfter ein - abkühlen - Lüfter aus - erwärmen usw.
Die Temperaturen, bei denen das Gebläse ein- und ausschaltet, können wir uns auch anzeigen lassen:

```
30 CLS
84 LOCATE 22,1
85 PRINT "Isttemperatur=";TE;"Grad
       Celsius"
```

Auch bei diesem Versuch lassen sich Hystereseverhalten und Schalthäufigkeit bei unterschiedlichen Solltemperaturen wie beim Modell Ofen grafisch darstellen. Versuchen Sie diese Programmänderungen selbst einmal!
Wir wollen uns nun mit einem anderen Effekt befassen, den wir vom Auto her kennen. Sie haben sicher schon einmal gelesen:
"Vorsicht, Lüfter läuft auch bei ausgeschalteter Zündung !".
Der Automotor wird also auch gekühlt, wenn er abgestellt wurde und seine Temperatur noch zu hoch ist. Das soll unser Modell nun auch machen. Dazu geben wir folgendes ein:

```
90 A$=INKEY$
100 IF A$="" THEN GOTO 50
110 CALL I3A
120 CALL I1R
130 CALL IEY(EY)
140 TE=FNT(EY)
150 LOCATE 22,1
160 PRINT "Isttemperatur=";TE;"Grad
       Celsius"
170 IF TE>20 THEN GOTO 130
180 CALL I1A
190 END
```

Nach dem Starten mit RUN läuft das Programm zunächst wie vorher. Abschalten läßt sich der "Motor" (Lampe) nun durch Drücken einer beliebigen Taste (Zeile 90 bis 100). Die Lampe erlischt, und der Lüfter läuft noch solange weiter, bis die Temperatur unter 20 °C gesunken ist. Danach endet das Programm. Wenn Sie den Temperaturverlauf mit der Schildkröte mitprotokolliert

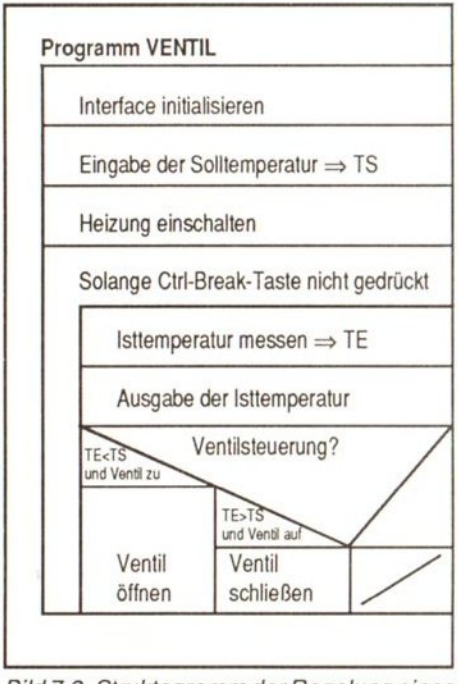


Bild 7.3. Struktogramm der Regelung eines Thermostatventils

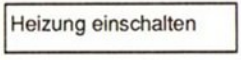
hatten, sollten Sie nicht vergessen, die Grafik abzuschalten:

185 CALL IGA : SCREEN 0 : CALL IW80

Die Temperaturabfrage sowie der Soll-/Istwert-Vergleich ist mit den Zeilen 40 - 80 identisch.
Das soll für diesen Versuch reichen. Natürlich läßt sich auch an dem Programm noch einiges verbessern. So können Sie z.B. den Wärmeerzeuger schrittweise aufheizen usw. Ein fertiges Programm zur Darstellung der Temperaturregelung durch Kühlung gibt es wieder auf der Diskette unter dem Namen GEBLAESE.BAS.

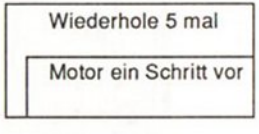
Man kann die Temperatur auch durch die Steuerung des Wärmeflusses regeln. Was man darunter versteht, läßt sich am besten anhand eines Thermostatventils eines Heizkörpers erklären. Der Heizkörper ist an einen Heizkreis angeschlossen, durch den laufend warmes Wasser gepumpt wird. Wieviel Wasser (= Wärmemenge) durch den Heizkörper fließen soll, bestimmt das Thermostatventil. Hier stellt man die gewünschte Temperatur ein. Das Ventil ist solange geöffnet, bis diese Temperatur erreicht ist. Dann schließt es; es kann kein Wasser nachfließen, bis die Temperatur wieder unter den Sollwert abgesunken ist. Dieses Prinzip wollen wir wieder durch einen Versuch kennenlernen und bauen dazu das Modell Ventil aus der Bauanleitung auf. Das Wasser ersetzen wir durch Luft und das Ventil durch einen Schieber. Über dem Heizelement, der Lampe, ist wieder der Temperaturfühler angebracht. Die Lampe ist am Ausgang M3, der NTC-Widerstand am Eingang EY des Interfaces angeschlossen. Neu ist der Schieber, der wie ein Ventil wirkt und den Wärmefluß von der Lampe zum Heißleiter unterbrechen soll. Er wird durch eine Bauplatte realisiert, die durch einen Motor (Ausgang M1) gedreht wird. Da der Motor im Schrittsteuerprinzip angetrieben wird, ist noch ein Schal-

Unter einem Struktogramm versteht man eine zeichnerische Darstellung des Programms. Das Struktogramm wird von oben nach unten gelesen. Treffen Sie ein Rechteck an, so wird die darin beschriebene Aktion ausgeführt:



Eine Verzweigung wird durch ein auf der Spitze stehendes Dreieck angegeben. Darunter folgen für die verschiedenen Wege nebeneinanderliegende Rechtecke:

Schleifen erhalten am Rand einen Balken. Entweder oben oder unten oder auch gar in der Mitte steht die Bedingung für die Wiederholung der Schleife, je nachdem ob am Anfang, am Ende oder in der Mitte die Prüfung auf das Ende der Schleife erfolgt:



ter (E2) an seiner Welle angebracht.
Für dieses Modell wollen wir jetzt ein Steuerprogramm erstellen. Den Ablauf sehen wir uns in Bild 7.3 an, einem sog. Struktogramm. Die Programmierung umfangreicherer Aufgaben sollte immer mit einem Struktogramm beginnen, um später Fehler besser zu finden und Ergänzungen einfacher einbauen zu können.
Unser Programm läuft nun folgendermassen ab: Man gibt eine Solltemperatur vor, die das Modell erreichen und halten muß (z.B. 36°C). Der Brenner wird eingeschaltet, der Schieber geschlossen und die Isttemperatur gemessen. Ist sie kleiner als der Sollwert, wird der Schieber geöffnet. Er bleibt solange offen, bis die Solltemperatur überschritten wird; dann schließt er.
Bevor Sie nun das folgende Programm abtippen, versuchen Sie doch einmal, das Struktogramm des Programms zu verstehen. Sie haben ja schon bei den bisherigen Experimenten Erfahrungen gesammelt.
Hat's geklappt? Prima! Sehen wir uns aber nun das Programm an:

```
LOAD"INIT"
10 CALL IIN
12 CLS
15 DEF FNT(X)=160-30*LOG(X)
20 INPUT "Solltemperatur:";TS
30 OT=TS+1 : UT=TS-1
40 S=0 : CALL I3R
50 CALL IEY(EY) : TE=FNT(EY)
60 PRINT "Temperatur=";TE
70 IF TE<UT AND S=0 THEN
       CALL I1V:S=1
80 IF TE>OT AND S=1 THEN
       CALL I1V:S=0
90 GOTO 50
```

Die Eingabe der Solltemperatur erfolgt in Zeile 20. Wählen Sie den Wert nicht zu niedrig, da die Lampe ganz schön heizt.
Zeile 30 legt die Grenzwerte für "Schieber öffnen" und "Schieber schließen" fest. Die Variable S in Zeile 35 hält die Schieberstellung fest (0=geschlossen). Nach Temperaturmessung erfolgt der Soll-/Istwert-Vergleich. Ist die Isttemperatur TE kleiner als die untere Sollwertgrenze und (AND) der Schieber geschlossen, öffnet der Schieber (Zeile 70). Das merken wir uns mit S=1. In Zeile 80 ist es genau umgekehrt: Ist TE>OT und (AND) der Schieber offen (S=1), wird er geschlossen. Das Programm läuft in einer Schleife; es fängt wieder bei Zeile 50 an.
Auch zu diesem Versuch gibt es wieder ein fertiges Programm, das den Namen VENTIL.BAS trägt.

Nach einer Richtlinie des VDI (VDI steht für Verband Deutscher Ingenieure, und die müssen es ja wissen) handelt es sich bei Robotern um "universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegung hinsichtlich Bewegungsfolge und -wegen bzw. -winkeln frei programmierbar und gegebenenfalls sensorgeführt sind".

Bis hierher haben wir eine ganze Reihe von Experimenten gemacht, die eigentlich nur ein einziges Ziel hatten: Der Computer sollte seine Umwelt erkennen können und seine Erkenntnisse wiederum zum Steuern einsetzen. So lernte der Computer sehen - mit Hilfe des Fotowiderstandes - und fühlen - mit Hilfe des NTC-Widerstandes - und er lernte eine Bewegung zu steuern - mit Hilfe des Motors.
Mit diesen Fähigkeiten hat unser Computer schon eine ganze Menge von dem Gehirn eines Roboters. Was ist eigentlich ein Roboter?
Ein Roboter ist eine Maschine oder Automat, der sich fast wie ein menschlicher Arm bewegen kann und solche Arbeiten, wie Greifen, Stapeln, Schweißen usw., ausführen kann. Was er in welcher Reihenfolge tun soll, wird ihm per Programm beigebracht. Und im Programm steht auch, ob er dabei auch Meßdaten, wie Helligkeit oder Wärme, berücksichtigen muß.
Was es mit den Bewegungsachsen, -wegen und -winkeln auf sich hat, wollen wir mit Hilfe unseres Modells kennenlernen. Dazu bauen wir das Robotermodell aus der Bauanleitung zunächst einmal auf. Das Gebilde ist ein sog. Schweißroboter. Die Schweißzange vorn realisieren wir durch ein Lämpchen. Sie brauchen jetzt natürlich keine Eisenteile bereitzulegen, geschweißt wird hier nicht. Mit dem Schweißroboter wollen wir nur die Bewegungsmöglichkeiten und die Programmierung eines Roboters kennenlernen. Und den Schweißroboter haben wir uns deshalb ausgedacht, weil er in der Produktion von Autos eine so wichtige Rolle spielt.
Wenn Sie den Roboter nun mit dem Interface an den Computer angeschlossen haben, versuchen Sie zunächst einmal, ihn zu bewegen, bevor wir auf die Bewegungsachsen zu sprechen kommen. Der Roboter muß sich in Grundstellung befinden: Schweißarm eingefahren und nach vorn gerichtet. Lösen Sie hierzu den Motor aus dem Getriebeeingriff und stellen Sie den Roboterarm richtig ein. Rasten Sie den Motor wieder in seine korrekte Stellung ein. Geben Sie dann ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL I1L
```

Nach Start des Programms dreht sich der gesamte Aufbau des Roboters um einige Grad. Mit der Zeile:

Industrieroboter haben meist sechs Achsen. Drei Hauptbewegungsachsen dienen dazu, den Greifarm in die richtige Position zu fahren. Daß dazu gerade drei Achsen notwendig sind, liegt daran, daß der Raum dreidimensional ist (Länge, Höhe und Breite). Drei Orientierungsachsen des Roboters sind im "Handgelenk" untergebracht. Sie dienen dazu, das Werkzeug oder das Werkstück richtig auszurichten (Drehen, Kippen, Wenden). Das Betätigen des Werkzeugs (Schweißzange, Schraubendreher usw.) oder des Greifers zählt bei den Achsen nicht mit.

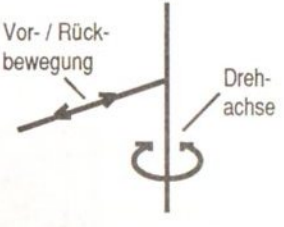


Bild 8.1: Bewegungsachsen unseres Schweißroboters.

20 CALL I1R

und erneutem Programmstart dreht er sich wieder zurück. Eine präzisere Drehung des Roboters erhalten Sie wieder mit der Verwendung der Schrittkommandos:

CALL I1V bzw. **CALL I1Z**

Wenn Sie

20 CALL I2V

eingeben und das Programm starten, wird sich die Spindel am Roboterarm kurz drehen. Stellen Sie die Spindel wieder in die Grundstellung (Schweißarm ganz eingezogen). Mit

20 FOR I=1 TO 12
30 CALL I2V
40 NEXT

und RUN fährt der Arm ganz aus. Ändern Sie in Zeile 30 den Befehl in CALL I2Z, dann fährt er wieder zurück. Aber Vorsicht, wenn die Spindel zu Beginn nicht ordnungsgemäß in Grundstellung gebracht wurde; in den Endstellungen kann sich die Mechanik leicht verklemmen. Machen Sie weitere Versuche, um die "Reichweite" des Roboters zu erforschen. Mit

20 FOR I=1 TO 10
30 CALL I1V
40 NEXT

und RUN dreht er sich um 90°. Ein Schritt entspricht somit 9°. Achten Sie dabei darauf, daß sich die Anschlußkabel des Roboters nicht verklemmen oder verheddern. Mit dem Kommando CALL I1Z in Zeile 30 dreht sich der Roboter wieder zurück.
Unser Roboter hat also zwei Bewegungsachsen: eine Drehung um die senkrechte Achse und eine Vor- und Zurückbewegung des Armes. Verglichen mit unserem Körper wäre das eine Drehung in der Hüfte und ein Vorstrecken des Armes.
Moderne Roboter besitzen natürlich noch wesentlich mehr Achsen. Sie können z.B. den Arm auf- und abbewegen oder den Greifer drehen, um beim Automobilbau in jede Ecke einer Karosserie zu gelangen.
Schematisch dargestellt kann sich unser Roboter wie in Bild 8.1 bewegen.
Daraus wollen wir nun den erreichbaren Raum des Roboters ableiten. Wir gehen dabei von einer direkten Bewegung ohne

Umgehung von Hindernissen aus. Versuchen Sie selbst, die entsprechende Fläche auf einem Blatt Papier zu skizzieren. Bewegen Sie den Roboter, wie oben beschrieben, wenn Ihnen noch etwas unklar ist.
Wie wir sehen, kann der Roboter auf eine ringförmige Fläche - ähnlich einer großen Unterlegscheibe - zugreifen. Er kann darauf mit zwei Bewegungsachsen jeden Punkt erreichen (Bild 8.2).
Man sagt auch, die Ausbreitung ist zweidimensional (Breite x Tiefe). Für industrielle Anwendungen werden meist alle drei Raumdimensionen gefordert, also auch die Höhe. Drei Achsen können Sie mit einem anderen fischertechnik-Modell, dem Trainingsroboter, bewegen.
Wir wollen uns nun hier mit der Programmierung unseres Roboters befassen, nachdem wir einiges über die Robotergeometrie gelernt haben. Die einzelnen Bewegungsschritte lassen sich zu einem Programm zusammenfassen. Und der Roboter wird dann eine Arbeit planmäßig ausführen - genau so, wie wir es ihm sagen.

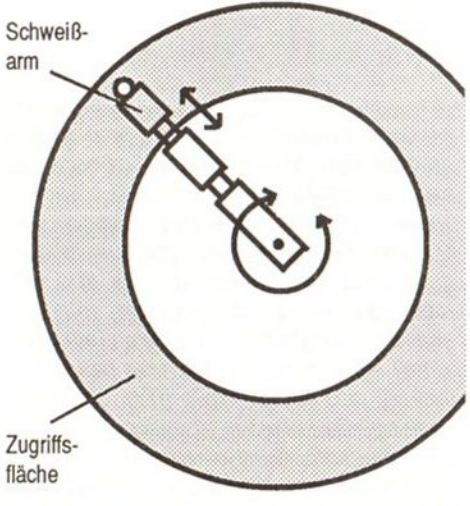


Bild 8.2: Zugriffsfläche des Schweißroboters.

Für den Schweißroboter wollen wir nun ein Steuerprogramm schreiben. Zunächst bringen wir ihn in Grundstellung, d.h. der Schweißarm ist eingefahren, und der Aufsatz zeigt nach vorn in Längsrichtung des Rahmens. Genau positionieren läßt er sich, wie wir im vorigen Abschnitt gesehen haben, mit den Befehlen

CALL I1V bzw. **CALL I1Z**
für die Drehachse und

CALL I2V bzw. **CALL I2Z**
für die Armbewegung.

Unser Roboter soll an einem Punkt A schweißen, dann in Grundstellung zurückfahren, dort eine Zeit warten und wieder zum Punkt A fahren, wo er erneut schweissen soll. Dieser Bewegungsablauf ist in Bild 8.3 übersichtlich dargestellt.
Der Roboter muß also folgende Aktionen nacheinander ausführen:

1. Drehung 45° nach rechts
2. Arm ausfahren
3. Schweißen
4. Arm einfahren
5. Drehung 45° nach links
6. Pause

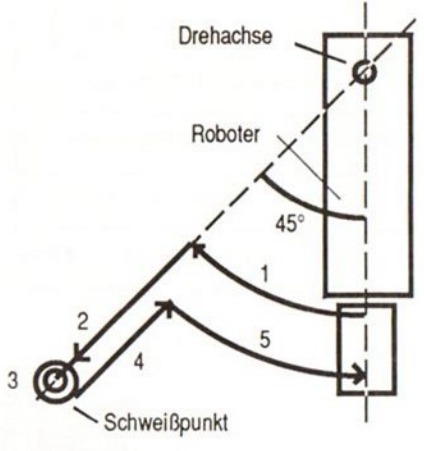


Bild 8.3: Bewegungsablauf beim Schweissen.

Die Drehrichtung "rechts" entspricht einer Drehung im Uhrzeigersinn, wenn man von oben auf das Modell sieht. Diese sechs Schritte realisieren wir im einzelnen wie folgt:
Zunächst erstellen wir wieder ein Struktogramm für das Programm (Bild 8.4).
Man erkennt, daß der Programmablauf linear von oben nach unten erfolgt; danach beginnt der Vorgang wieder von vorn. Für jeden Bewegungsteil ist ein Programmabschnitt zuständig. Man nennt diese Methode auch "lineare Programmierung" eines Roboters. Zu dem Struktogramm wird nun das entsprechende Programm erstellt.
Versuchen Sie es zunächst selbst, bevor Sie weiterlesen.

```
LOAD"INIT"
10 CALL IIN
20 REM Arm rechts
30 FOR I=1 TO 5
40 CALL I1Z
50 NEXT I
60 REM Arm vor
70 FOR I=1 TO 12
80 CALL I2V
90 NEXT I
100 REM Schweißen
110 FOR I=1 TO 500
120 CALL I3R
130 NEXT I
140 CALL I3A
150 REM Arm zurück
160 FOR I=1 TO 12
170 CALL I2Z
180 NEXT I
190 REM Arm links
200 FOR I=1 TO 5
210 CALL I1V
220 NEXT I
230 REM Pause
240 FOR I=1 TO 2000
250 NEXT I
260 GOTO 30
```

Haben Sie es geschafft? So sollte das Programm aussehen. Sie sehen, es hat jede Menge FOR...NEXT-Schleifen; für jede Armbewegung ist eine solche Schleife notwendig, da die Bewegungen jeweils aus Einzelschritten bestehen.
So hat z.B. die erste Schleife für die Armdrehung rechts um 45° fünf Durchläufe, d.h. es wird fünfmal der Befehl CALL I1Z ausgegeben. Dabei dreht sich der Arm jedesmal um 9°. Auch das Schweißen und die Wartepause am Ende des Durchlaufs sind mit FOR...NEXT-Schleifen aufgebaut.

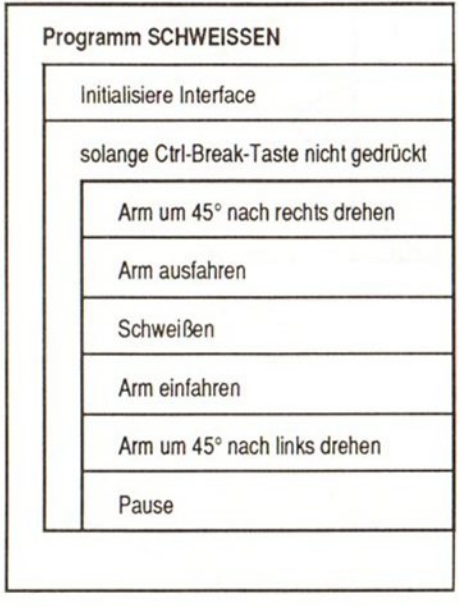


Bild 8.4: Struktogramm zum Programm "Schweißen".

Hier dienen sie aber dazu, jeweils eine bestimmte Zeit verstreichen zu lassen.
Wenn Sie das Programm eingegeben haben und es mit RUN starten, wird der Roboter seine Arbeit nach dem Programm genauso ausführen, wie wir es ihm vorgeschrieben haben. Anhalten läßt er sich mit der Ctrl-Break-Taste.
Unser Programm läuft zwar einwandfrei, hat aber einen Nachteil: es läßt sich nur für diesen einen Arbeitsvorgang gebrauchen. Bei Änderungen, z.B. Drehung nach links oder Schweißen an zwei Punkten, muß es komplett neu geschrieben werden. Bei kleinen Programmen ist das nicht allzu schwierig, bei großen macht dies aber schon eine Menge Arbeit. Daß es auch anders geht, zeigt das nächste Kapitel.

Jede Fabrik, in der ein Roboter für irgendwelche Arbeiten eingesetzt wird, muß sicher ab und zu das Programm für den Roboter ändern, wenn ein neues Werkstück gefertigt werden soll oder sich der Arbeitsablauf ändert. Nehmen wir nur das Beispiel Autoindustrie: jedes Jahr kommt ein neues Modell auf den Markt, für das immer wieder derselbe Schweißroboter eingesetzt wird. Ein Programm, das vielleicht mehrere hunderttausend Mark kostet, jedesmal neu zu kaufen oder zu entwickeln, ist natürlich viel zu teuer. Dafür gibt es eine billigere Lösung: die Tabellenprogrammierung des Roboters.
Die Befehle für den Bewegungsablauf des Roboters werden in einer Tabelle im Computerspeicher abgelegt und nacheinander aufgerufen. Für andere Aufgaben wird dann nur die Tabelle neu erstellt.
Dies wollen wir jetzt auch mit unserem Schweißroboter ausprobieren. Wir nehmen denselben Arbeitsablauf wie im vorigen Kapitel und erstellen dazu ein Tabellenprogramm. Überlegen wir uns, wie wir die Tabelle gestalten. Jede Aktion des Roboters versehen wir mit einem Kennbuchstaben:

V - Schweißarm vorwärts

Die BASIC-Funktion VAL(...) ist ganz nützlich um Zahlenwerte aus einer Zeichenkette herauszuziehen. Sie beginnt am Anfang der Zeichenkette und berücksichtigt alle Zeichen, die zu einem Zahlenwert beitragen (z.B. Ziffern und Dezimalpunkt). Die Auswertung endet bei dem ersten Zeichen, das nicht zu einem Zahlenwert gehört.
Die BASIC-Funktion RIGHT$(...) schneidet einen Teil der Zeichenkette aus. Sie beginnt beim rechten Ende und nimmt soviele Zeichen, wie in dem zweiten Argument angegeben sind.
In unserem Fall wird die Roboteraktion also durch das rechte Ende des Tabelleneintrags, das dazugehörige Maß durch das linke Ende des Tabelleneintrags bestimmt. Dies erlaubt interessante Möglichkeiten der Kommentierung der Tabelle, wobei allerdings keine Kommas oder Leerzeichen erlaubt sind, z.B.:

12Schritte_bis_zum_Objekt_V

Z - Schweißarm zurück
R - Roboter nach rechts drehen
L - Roboter nach links drehen
S - schweißen
P - Pause
E - Ende des Programms

Zu allen Aktionen außer dem Programmende müssen wir dann noch Maßzahlen angeben, z.B. um wieviele Schritte der Schweißarm vorgeschoben werden soll oder wie lange geschweißt werden soll. Der Bequemlichkeit halber setzen wir die Maßzahl vor die Kennziffer der Aktion; so läßt sich die Tabelle nachher leichter per Programm auswerten. Der Bewegungsablauf des vorigen Kapitels stellt sich wie folgt dar:

5R (Roboter um fünf Schritte nach rechts)
12V (Schweißarm um zwölf Schritte vor)
500S (500 Schleifendurchläufe lang schweißen)
12Z (Schweißarm um zwölf Schritte zurück)
5L (Roboter um fünf Schritte nach links)
2000P (2000 Schleifendurchläufe lang pausieren)
E (Ende des Programms)

Die Tabelle im Computer wird in Form von DATA-Zeilen geführt. Bitte eingeben:

LOAD"INIT"
900 DATA 5R,12V,500S,12Z,5L,2000P,E

Die Tabellenwerte, z.B. 5R sind nicht mit Kommandos zu verwechseln. Welche Kommandos benötigt werden, muß das Programm erst noch aus den Tabellenwerten ermitteln. Wir können die Tabellenwerte "5R", "12V" usw. nacheinander lesen und ausführen lassen. Dies erfolgt mit:

20 RESTORE
30 READ A$

In der Variablen A$ steht dann der Befehl, den wir weiterverarbeiten können. Wir trennen im ersten Schritt die Maßzahl ab; dies erfolgt durch die Funktion VAL(...). Die Aktion ermitteln wir als den am weitesten rechts stehenden Buchstaben des Befehls mit der Funktion RIGHT$(...). Probieren wir das Ganze einmal aus; geben Sie ein:

10 CALL IIN
20 RESTORE
30 READ A$
40 W=VAL(A$)

```
50 K$=RIGHT$(A$,1)
60 IF K$="V" THEN GOTO 200
70 IF K$="Z" THEN GOTO 300
80 IF K$="R" THEN GOTO 400
90 IF K$="L" THEN GOTO 500
100 IF K$="S" THEN GOTO 600
110 IF K$="P" THEN GOTO 700
120 IF K$="E" THEN END
130 GOTO 30
```

Dieser erste Teil des Programms zerlegt den Befehl in seine Bestandteile und verzweigt gemäß der gewünschten Aktion in die nachfolgenden Programmteile.

```
200 REM Schweißarm vor
210 FOR I=1 TO W
220 CALL I2V
230 NEXT I
240 GOTO 30
300 REM Schweißarm zurück
310 FOR I=1 TO W
320 CALL I2Z
330 NEXT I
340 GOTO 30
400 REM Roboter nach rechts
410 FOR I=1 TO W
420 CALL I1Z
430 NEXT I
440 GOTO 30
500 REM Roboter nach links
510 FOR I=1 TO W
520 CALL I1V
530 NEXT I
540 GOTO 30
600 REM Schweißzange ein
610 FOR I=1 TO W
620 CALL I3R
630 NEXT I
640 CALL I3A
650 GOTO 30
700 REM Pause
710 FOR I=1 TO W
720 NEXT I
730 GOTO 30
```

Starten Sie das Programm mit RUN. Der Roboter durchläuft einmal den vorigen Bewegungsablauf.
Probieren Sie nun eigene Arbeitsabläufe aus - Sie brauchen nur die Tabelle entsprechend zu ändern. Geben Sie z.B. ein

```
900 DATA 6R,12V,200S,12Z,12L
910 DATA 1000P,12V,200S,12Z,6R,E
```

Der Roboter macht (fast) alles mit, was Sie ihm vorschreiben. Sie sollten allerdings darauf achten, daß der Bewegungsablauf

immer in der Grundstellung endet. Sie ersparen sich damit, ihn vor jedem Programmstart eigens wieder in die Grundstellung zu bringen. Wenn Sie diese Regel bei der Erstellung der DATA-Zeile(n) befolgen, können Sie den Bewegungsablauf auch vom Programm beliebig oft wiederholen lassen:

```
120 IF K$="E" THEN GOTO 20
```

Jetzt zeigt sich der Nutzen der RESTORE-Anweisung. Sie bewirkt, daß das nächste READ-Kommando wieder auf den ersten Tabellenwert in der ersten DATA-Zeile zugreift.
Das vorliegende Programm sollten Sie auf Diskette speichern oder im Computer geladen lassen, denn im nächsten Kapitel wird es noch ausgebaut.
Die Programmierungsart mit Bewegungstabellen wird auch bei professionellen Robotern angewandt; sie macht den Roboter universell einsetzbar. Auf Diskette befindet sich wieder ein etwas erweitertes Programm zu diesem Thema:
ROBOTTAB.BAS.

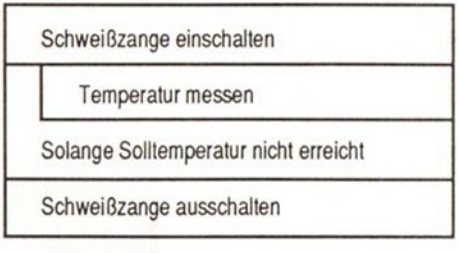


Bild 8.5: Struktogramm zum Programm "Schweißen mit Sensor".

Wer sich mit der Schweißtechnik auskennt, weiß, daß unterschiedliche Materialien verschiedene Schweißmethoden erfordern. Unter anderem ist die Temperatur der Schweißnaht wichtig für die spätere Haltbarkeit der Verbindung. Unser Roboter soll diese Prüfung auch durchführen; d.h. er soll feststellen, wann die Schweißstelle die richtige Temperatur hat, bevor er sich zur nächsten begibt. Genau dasselbe macht ein moderner Industrieroboter auch.
Dazu bauen wir zunächst die Variante des Schweißroboters aus der Bauanleitung auf. Im Prinzip sieht der Roboter genauso aus wie zuvor, nur besitzt er jetzt an der "Schweißzange" einen Temperatursensor. Diesen NTC-Widerstand kennen wir bereits aus früheren Versuchen - unser Roboter lernt jetzt also Wärme fühlen und bezieht die gemessene Temperatur in den Schweißvorgang mit ein. Einen Fühler bezeichnet man ganz allgemein auch als Sensor, so daß wir unseren Schweißroboter ohne Übertreibung als sensorgeführt bezeichnen können.
Dazu müssen wir jetzt die Temperatur mit in das Programm einbauen, denn die Schweißdauer soll ja jetzt vom Signal des Heißleiters abhängen. Wenn er die richtige

Neben Temperaturfühlern benutzt man in der Robotik auch noch optische Sensoren, um bestimmte Punkte zu finden, oder Meßfühler für Materialdickenprüfung und dgl. Damit lassen sich die Roboterarme millimetergenau führen und Arbeitsabläufe exakt einstellen. Ganz moderne Roboter sind sogar mit einer Videokamera ausgestattet. Die Bildauswertung erlaubt dem Roboter z.B. auch dann sicher zuzugreifen, wenn Teile in wahlloser Ausrichtung auf einem Förderband gebracht werden. Oder die Videokamera ist mit einem Infrarotfilter für Wärmestrahlung ausgestattet. Dann kann der Roboter die Qualität seiner Schweißnaht sogar "sehen".

Temperatur meldet, wird der Schweißvorgang abgebrochen.
In der Bewegungstabelle geben wir jetzt als Maßzahl die Temperatur des Heißleiters ein, die später erreicht werden soll, z.B. 35S. Diese Temperatur entspricht natürlich nicht der Schweißtemperatur einer richtigen Schweißzange; diese liegt wesentlich höher, über 2800 °C.
Wenn der Roboter einen Schweißvorgang durchführen muß, wird zunächst die Schweißzange eingeschaltet. Laufend wird jetzt die Temperatur an der Schweißzange gemessen. Wenn der Sollwert erreicht ist, wird der Schweißvorgang beendet.
Beginnen wir wieder mit dem Struktogramm (Bild 8.5). Ein Vergleich mit dem Programm des vorigen Kapitels zeigt, daß lediglich der Programmabschnitt, der das Schweißen steuert, ausgetauscht wird.
Versuchen Sie das Programm wieder selbst zu schreiben, bevor wir es besprechen.

15 DEF FNT(X)=160-30*LOG(X)

Dies ist die bekannte Umrechnung des Analogwertes in °C gemäß Kapitel 7. Dort steht auch beschrieben, wie Sie sich durch Kalibrierung des NTCeine genauere Temperaturanzeige des NTC verschaffen.

600 REM Schweißen (mit Sensor)
610 CALL I3R
620 CALL IEY(EY)
630 T=FNT(EY)
640 IF T<=W THEN GOTO 620
650 CALL I3A
660 GOTO 30

Selbstverständlich muß auch die DATA-Zeile an das neue Programm angepaßt werden. Anstelle der Zahl der Schleifendurchläufe für die Schweißdauer erscheint jetzt die Temperaturangabe in °C:

900 DATA 5R,12V,35S,12Z,5L,2000P,E

Geben Sie die Zeilen ein und starten das Programm mit RUN. Der Roboter wird jetzt die Schweißlänge von der Temperatur abhängig machen. Probieren Sie auch andere Temperaturen.
Wir haben mit diesem Versuch einen sensorgesteuerten Roboter kennengelernt, der auf Wärme reagiert.
Die Roboterbewegung mit Sensorführung können Sie auch wieder mit dem Musterprogramm auf Diskette studieren. Sein Name ist ROBOTSNS.BAS.

Eine andere Variante des Programms ist ROBOTGRA.BAS. In diesem Programm wird die Grafik-Schildkröte dazu benutzt, um die Bewegungen des Roboters am Bildschirm darzustellen. Die Schildkröte zeichnet dabei nicht, sie dient lediglich als Cursor, um die Schweißpunkte anzuzeigen. Als Hintergrund wird eine Autokarosserie eingeblendet. Studieren Sie dieses Programm; es zeigt Ihnen, wie einfach eine Begleitgrafik eingebaut werden kann.

9.1 Bewegung der Schidkröte: Zwei rechts, zwei links

Wir nennen das Fahrzeug "Schildkröte". Der Begriff kommt vom Englischen und heißt dort "Turtle". Es gibt dafür auch noch andere Namen, "Igel" beispielsweise - ihre Bedeutung ist aber immer dieselbe. Man hat das Wort "Schildkröte" oder "Turtle" gewählt, weil manche Modellroboter mit einem Schreibstift versehen sind und eine Linie entlang ihrer Fahrspur ziehen - ähnlich wie eine Schildkröte mit ihrem Schwanz im Sand. Die Programmiersprache LOGO und die Grafik-Schildkröte stammen übrigens auch von einem fahrbaren Modellroboter ab.

Wenn bisher von Robotern die Rede war, dann haben wir darunter stationäre Arbeitsautomaten verstanden. Sie standen auf einem Grundrahmen und hatten lediglich einen beweglichen Arm, mit dem sie ihre nähere Umgebung erreichen konnten. Etwas Neues werden Sie jetzt kennenlernen: den Fahrroboter.

Wie der Name schon sagt, kann er umherfahren und an verschiedenen Orten arbeiten. Typische Fahrroboter sind die sog. Flurförderfahrzeuge, die Lasten ohne Führer hin- und hertransportieren können. Wie diese Fahrzeuge - oder besser Roboter - gesteuert werden und was sie alles können, zeigt uns das nächste Modell, das wir nach der Bauanleitung zusammenbauen.

Bevor wir dieses Fahrzeug in Betrieb nehmen, sehen wir es uns erst einmal genau an. Es besitzt auf der linken und rechten Seite jeweils unabhängig voneinander angetriebene Räder. Sie werden von zwei Motoren im Schrittsteuerprinzip angetrieben, was man an den beiden Mikroschaltern auf der Rückseite des Modells erkennt. Das Gleichgewicht wird durch ein drittes Rad im Heck erreicht, das sich frei bewegen kann und nicht angetrieben wird. Vorn hat das Fahrzeug noch eine Stoßstange, hinter der ebenfalls ein Schalter sitzt. Beim Drücken auf die Stoßstange macht er sich durch Klicken bemerkbar. Außerdem ist über der Achse noch ein optischer Sensor angebracht.

Damit die Schildkröte in den folgenden Experimenten exakt läuft, sollten Sie sich Mühe bei der Ausrichtung der Mechanik geben. Die Schildkröte ist dann am besten justiert, wenn sie bei den folgend beschriebenen Kommandos besonders schnell läuft.

Verbinden Sie jetzt das Kabel der Schildkröte mit dem Interface. Wenn Sie den Interfacetreiber und BASIC geladen haben, geben Sie Folgendes ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL I1V
```

Wenn das Programm gestartet wird, dreht sich der in Fahrtrichtung rechte Motor kurz, die Schildkröte insgesamt dreht sich um wenige Grad gegen den Uhrzeigersinn (von oben betrachtet). Mit der Programmänderung:

```
20 CALL I2V
```

und nochmaligem Programmstart passiert

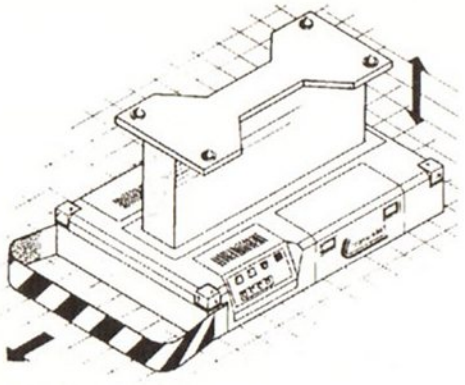


Bild 9.1: Ein fahrerloses Transportfahrzeug mit Hubtisch

Fahrbare Roboter werden auch als FTS (= fahrerlose Transportsysteme) oder englisch AGV (= automatically guided vehicles) bezeichnet. In der Produktion bieten sie eine höhere Flexibilität als Förderbänder. Wenn sie Lasten transportieren, müssen sie nur diejenigen Stationen anfahren, wo die Lasten aufgenommen oder abgegeben werden. Die Aufträge erhalten sie per Funk oder anderen Methoden von einem Prozeßrechner. Manche FTS sehen wie Hubtische, andere wie Gabelstapler aus. Wieder andere ziehen wie eine Lokomotive eine Schar von Anhängern. Wie Roboter aus den Science-Fiction-Geschichten sehen sie aber alle nicht aus.

dasselbe mit dem linken Motor. Die Schildkröte dreht sich nach rechts. Lassen wir sie im Kreis fahren:

```
20 FOR I=1 TO 150
30 CALL I2V
40 NEXT I
```

Geben Sie die Zeilen ein und starten das Programm mit RUN. Die Schildkröte dreht sich im Uhrzeigersinn um das rechte Rad. Achten Sie dabei auf das Anschlußkabel, damit es sich nicht verklemmt, und stellen Sie die Schildkröte nach Drehungen immer wieder in die Ausgangsstellung zurück. Das Kabel zum Interface wäre sonst nach ein paar Versuchen aufgewickelt. Bei der späteren Programmierung müssen Sie darauf besonders achten.
Geradeaus-Bewegungen sind nur möglich, wenn beide Motoren gleichzeitig angesteuert werden. Wenn Sie

```
30 CALL I1V : CALL I2V
```

eingeben, sehen Sie, daß sich jeweils zuerst das rechte und dann das linke Rad dreht. Die Schildkröte bewegt sich zickzackförmig vorwärts. Wie es eleganter geht, zeigen wir im nächsten Kapitel.

9.2 Codierung der Fahrroute: Wegweisungen

Bei unseren ersten Bewegungsversuchen mit der Schildkröte haben wir gesehen, daß wir sie mit Hilfe von zwei unabhängig voneinander angetriebenen Rädern vor-, zurückfahren und sich drehen lassen können. Sie bewegt sich also wie ein Kettenfahrzeug, und das gilt natürlich auch für die Geradeausfahrt. Dazu müssen beide Motoren gleichzeitig angesteuert werden. Da dies mit den bisherigen Kommandos nicht möglich ist, führen wir neue Befehle ein. Geben Sie ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 S%=1 : CALL ITV(S%,TS,E5)
```

Starten Sie das Programm und die Schildkröte fährt vorwärts. Das Kommando CALL ITI diente dazu, die Schildkröte zu initialisieren. Was das Initialisieren bei der Schildkröte bewirkt, erfahren Sie in Kapitel 10. Vorerst gehen wir davon aus, daß das Initialisieren vor dem Gebrauch der neuen Schildkrötenbefehle erforderlich ist. Das nächste Kommando bewegt die Schildkröte mit beiden Motoren gleichzeitig einen Schritt vorwärts. Mit:

Fassen wir die Schildkrötenbefehle zusammen:

CALL ITI

Die Schildkröte wird initialisiert.

CALL ITV(S%,TS,E5)

*Die Schildkröte fährt um S%*5 mm vorwärts.*

CALL ITZ(S%,TS)

*Die Schildkröte fährt um S%*5 mm zurück. (S% = 0 .... 32767)*

CALL ITR(D%,TS)

Die Schildkröte dreht sich um D% Grad nach rechts.

CALL ITL(D%,TS)

Die Schildkröte dreht sich um D% Grad nach links. (D% = 0 .... 355, durch 5 teilbar)

Welchen Nutzen die Parameter TS und E5 haben, erfahren Sie in Kapitel 10.

30 S%=1 : CALL ITZ(S%,TS)

bewegt sie sich nach dem Programmstart zurück. Eine längere Strecke legt sie mit:

30 S%=20 : CALL ITV(S%,TS,E5)

zurück. Nun fährt sie genau 10 cm vor. Damit kennen wir auch schon die Schrittweite für ein Vorwärtskommando CALL ITV: Es werden 20 Schritte an beiden Motoren ausgeführt. 10 cm Strecke geteilt durch 20 ergibt eine Fahrstrecke von 5 mm pro Schritt. Für die Rückwärtsbewegung gilt natürlich das Gleiche. Geben Sie

30 S%=20 : CALL ITZ(S%,TS)

ein, und die Schildkröte fährt wieder 10 cm zurück.

Durch Ansteuerung beider Motoren gleichzeitig ist auch eine Drehung der Schildkröte um die eigene Achse möglich. Dabei muß sich ein Rad vor und das andere zurückdrehen. Auch dafür gibt's zwei neue Befehle. Geben Sie

30 D%=5 : CALL ITR(D%,TS)

ein. Die Schildkröte dreht sich um einige Grad im Uhrzeigersinn (von oben auf die Schildkröte gesehen). Mit:

30 D%=5 : CALL ITL(D%,TS)

dreht sie sich wieder zurück. Nun wollen wir natürlich noch wissen, um wieviel Grad sie sich bei einem Befehl dreht. Dies ermitteln wir durch einen größeren Drehwinkel:

30 D%=90 : CALL ITR(D%,TS)

Bevor Sie die Taste Return am Ende der Zeile drücken, merken Sie sich die Achsenlinie (Kabel evt. anheben!). Die Schildkröte dreht sich um einen rechten Winkel. Unsere Vermutung bestätigt sich: das Drehmaß der Schildkröte wird direkt in Winkelgraden eingegeben. Probieren Sie jetzt mal aus:

30 D%=37 : CALL ITR(D%,TS)

Die Schildkröte reagiert überhaupt nicht, stattdessen wird auf dem Bildschirm eine Fehlermeldung angezeigt, die besagt, daß die Schildkröte solche Schritte nicht ausführen kann. Der Grund: Wenn der rechte Motor um einen Schritt zurückläuft und der linke Motor um einen Schritt vorläuft, dreht sich die Schildkröte um 5°. Bei größerer

Zahl von Drehschritten wird die Gradzahl immer ein Vielfaches von 5° sein. Das Kommando prüft die Maßzahl somit auf Durchführbarkeit. Ein ähnlicher Fall:

```
30 D%=400 : CALL ITR(D%,TS)
```

Auch dieser Befehl führt zu einer Fehlermeldung, denn eine Drehung um mehr als 360° (Volldrehung) führt zu nichts außer einem verdrehten Anschlußkabel. Das Kommando läßt deshalb höchstens Drehwinkel von 355° zu. In obigen Fall hätten wir also besser:

```
30 D%=40 : CALL ITR(D%,TS)
```

angegeben.
Zur Probe versuchen Sie, folgende Schildkrötenbewegung hintereinander ablaufen zu lassen: 5 cm vorwärts, 90° Rechtsdrehung, 8 cm zurück. Vergleichen Sie Ihr Programm mit diesem:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 S%=10 : CALL ITV(S%,TS,E5)
40 D%=90 : CALL ITR(D%,TS)
50 S%=16 : CALL ITZ(S%,TS)
```

Machen Sie sich weiter mit der Schildkrötensteuerung vertraut - Sie werden sehen, daß man mit den vier Elementarkommandos auch die kompliziertesten Wege beschreiben kann. Wird die Strecke allerdings zu umfangreich, sind also viele Drehungen nötig und Teilstücke zu durchfahren, ist das bis jetzt angewandte Programmierverfahren nicht empfehlenswert. Wir wollen im nächsten Kapitel eine andere Möglichkeit zeigen, Steuerprogramme für die Schildkröte zu schreiben. Wir kennen sie übrigens schon von der Roboterprogrammierung her.

9.3 Routenplanung mit der Schildkröte: Planspiele

Eine Schildkröte ist ein Fahrroboter und soll deshalb auch größere Strecken mit mehreren Richtungsänderungen zurücklegen können. Stellen Sie sich ein Flurförderfahrzeug vor, das in einer großen Halle Material von einer Ecke in eine andere transportiert und dabei kreuz und quer durch die Halle fahren muß. Das Programm dafür ist natürlich entsprechend umfangreich.

Solch ein Programm wollen wir jetzt auch für unsere Schildkröte erstellen. Wir wenden dafür die numerische Routenplanung an. Wir kennen diese Art der Programmierung bereits von unserem Roboter: in einer Tabelle werden die einzelnen Bewegungsschritte zusammengestellt, die die Schildkröte nacheinander ausführt. Der Vorteil dieses Verfahrens ist wieder die universelle Anwendbarkeit des Programms. Man braucht nur die Tabelle zu ändern und schon fährt die Schildkröte einen anderen Weg. Gut, daß wir die Methode uns schon beim Schweißroboter angeschaut haben. Das neue Programm ist sogar einfacher als jenes des Schweißroboters. Als Aktionen haben wir die vier Schildkrötenbewegungen **R**echts, **L**inks, **V**orwärts und **R**ückwärts. Probieren wir's aus!

LOAD"INIT"

```
10 CALL IIN
20 CALL ITI
30 RESTORE
40 READ A$
50 W%=VAL(A$)
60 K$=RIGHT$(A$,1)
70 IF K$="V" THEN GOTO 200
80 IF K$="Z" THEN GOTO 300
90 IF K$="R" THEN GOTO 400
100 IF K$="L" THEN GOTO 500
110 IF K$="E" THEN END
120 GOTO 40
200 REM Schildkröte vor
220 CALL ITV(W%,TS,E5)
230 GOTO 40
300 REM Schildkröte zurück
320 CALL ITZ(W%,TS)
330 GOTO 40
400 REM Schildkröte rechts drehen
420 CALL ITR(W%,TS)
430 GOTO 40
500 REM Schildkröte links drehen
520 CALL ITL(W%,TS)
530 GOTO 40
```

Die Bahn der Schildkröte wird wieder in DATA-Zeilen codiert, z.B.:

```
900 DATA 20V,60R,20V,60R,20V,60R,
      20V,60R,20V,60R,20V,300L,E
```

Wenn Sie einen Monochrom-Schirm benutzen, geben Sie folgende zusätzlichen Kommandos ein:

29 CALL IHA
110 IF K$="E" THEN CALL IHE : CALL IGA : SCREEN 0 : CALL IW80 : END

Die Bedeutung der zusätzlichen Kommandos ist in dem Anhangkapitel A1.4 erläutert.

Probieren Sie das Programm aus. Welche geometrische Figur beschreibt die Schildkröte? Warum steht an vorletzter Stelle in der Tabelle das Kommando 300L und nicht 60R? Sie können die DATA-Zeile austauschen und eigene Bahnen codieren. Ihrer Phantasie sind dabei keine Grenzen gesetzt. Ein ähnliches Programm befindet sich auf der Diskette; sein Name ist ROUTENUM.BAS.

Wie wär's denn mit einer schönen Darstellung der Schildkrötenroute auf dem Bildschirm? Wozu haben wir eine Grafik-Schildkröte, die fast den gleichen Befehle gehorcht? Wenn zu dem Kommando CALL ITV das Kommando CALL IGV gestellt wird, macht die echte Schildkröte die Schritte in Vorwärtsrichtung und die Grafik-Schildkröte saust auf dem Bildschirm entsprechend vor. Die Grafik-Schildkröte hinterläßt auf dem Bildschirm auch noch ihre Spur, wenn der Grafikstift eingeschaltet war. Genauso entsprechen sich das Rückwärts- und die Drehkommandos. Wenn also alle Kommandos gleichermaßen an die echte und die Grafik-Schildkröte gehen, wird die Route der Schildkröte auf dem Bildschirm aufgezeichnet. Das Programm wird um die folgenden Zeilen ergänzt:

25 SCREEN 1 : CALL IGE
210 CALL IGV(W%)
310 CALL IGZ(W%)
410 CALL IGR(W%)
510 CALL IGL(W%)

Die Zeile 110 muß geändert werden, um beim Programmende die Grafik wieder abzuschalten:

110 IF K$="E" THEN CALL IGA : SCREEN 0 : CALL IW80 : END

Die Programme auf der Diskette benutzen Hintergrundbilder, die mit CALL IGLOAD geladen werden (s. Kap. 6). Dies steht Ihnen ebenso frei:

26 F$="TURTLE"
27 CALL IGLOAD(F$)
28 CALL IGC

Nach Einfügen dieser Zeilen bewegt sich die Schildkröte auf einem Rasterfeld, das Ihnen die Orientierung erleichtert. Beachten Sie aber, daß beim Ausführen dieses Programms die fischertechnik Diskette in das Diskettenlaufwerk eingelegt sein muß, damit das Bild geladen werden kann.

Man nennt dies ein Teach-In-Verfahren, zu deutsch: Lehrverfahren, da der Roboter seinen Bewegungsablauf gewissermaßen erlernt. Der große Vorteil des Teach-In-Verfahrens: der Roboter-Instruktor sieht sofort, was der Roboter macht und muß sich dies nicht aus Tabellen vorstellen. Oder hätten Sie sofort den DATA-Zeilen der vorigen Kapitel angesehen, was die Roboter im einzelnen machen?

9.4 Teach-In Verfahren: Lernfähig

Modernen Robotern bringt man ihren Bewegungsablauf durch Ausprobieren bei. Schrittweise werden sie von Position zu Position gebracht. Den Ablauf merkt sich der Roboter (bzw. sein Steuercomputer) und macht daraus eine Bewegungstabelle, nach der er dann arbeitet.
Wir wollen dies jetzt auch mit unserer Schildkröte ausprobieren. Dazu nehmen wir wieder die Schildkröte und schließen sie am Interface an. Kontrollieren Sie, ob alles richtig gesteckt und der Interfacetreiber sowie BASIC geladen sind. Geben Sie dann ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 S%=1 : CALL ITV(S%,TS,E5)
```

Nach dem Start des Programms muß sich die Schildkrötet einen Schritt vorbewegen. Wenn nicht, kontrollieren Sie bitte alles noch einmal nach.
Bevor wir mit der Routenplanung nach dem Teach-In-Verfahren beginnen, wollen wir uns zunächst eine feste Fahrunterlage schaffen, auf der sich die Schildkröte leicht bewegen kann. Außerdem lassen sich darauf mit Bleistift oder Klebeband Wege und Markierungen für die spätere Fahrstrecke darstellen.
Schneiden Sie sich aus Sperrholz oder dickem Karton eine ca. 50 cm x 50 cm große Platte aus. Darauf zeichnen Sie mit weichem Bleistift die geplante Fahrstrecke - aber dünn, damit Sie sie auch wieder ausradieren können!
Die Schildkröte soll vom Ausgangspunkt ein Viereck befahren: zeichnen Sie also ein Quadrat mit einer Kantenlänge von 10 cm mitten auf die Platte, und setzen Sie die Schildkröte auf einen Eckpunkt. Dieser sollte unter der Achse der Schildkröte liegen. Bild 9.2 zeigt die genaue Startposition.
Jetzt benötigen wir ein Programm, mit dem die Schildkröte den Weg "erlernt" und ihn später abfährt. Wir befahren die Route mit der Schildkröte, indem wir am Computer jeweils eine Taste für die gewünschte Richtung drücken. Dafür benutzen wir die Cursortasten, die Folgendes bewirken sollen:

Cursor hoch:	Schildkröte 1 Schritt vor
Cursor runter:	Schildkröte 1 Schritt zurück
Cursor rechts:	Drehung um 5° nach rechts
Cursor links:	Drehung um 5° nach links

Das Programm für die Cursortastenabfrage sieht folgendermaßen aus:

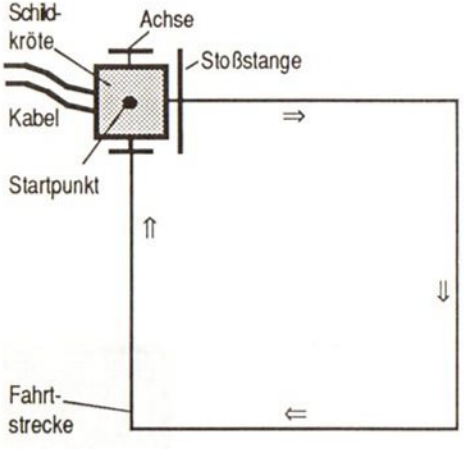


Bild 9.2: Fahrstrecke der Schildkröte.

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
40 AUF$=CHR$(0)+CHR$(72)
50 AB$=CHR$(0)+CHR$(80)
60 RE$=CHR$(0)+CHR$(77)
70 LI$=CHR$(0)+CHR$(75)
100 K$=INKEY$
110 IF K$="" THEN GOTO 100
130 IF K$<>AUF$ THEN GOTO 170
140 S%=1 : CALL ITV(S%,TS,E5)
160 GOTO 100
170 IF K$<>AB$ THEN GOTO 210
180 S%=1 : CALL ITZ(S%,TS)
200 GOTO 100
210 IF K$<>RE$ THEN GOTO 250
220 D%=5 : CALL ITR(D%,TS)
240 GOTO 100
250 IF K$<>LI$ THEN GOTO 290
260 D%=5 : CALL ITL(D%,TS)
280 GOTO 100
290 IF K$<>"E" THEN GOTO 100
320 END
```

In den Zeilen 40 bis 70 werden die Zeichencodes für die Cursortasten festgelegt.
Die Tastaturabfrage Zeile 100 und 110 kennen wir schon: das Programm wartet solange, bis eine Taste gedrückt wird. Der Tastencode steht dann in der Variablen K$. Welche Taste gedrückt wurde, wird in Zeile 130 bis 290 abgefragt. Hier stehen die Codes für die vier Cursortasten. Je nach Cursortaste (Richtung) wird der entsprechende Befehl ausgeführt. Drückt man "E", wird das Programm abgebrochen. Ansonsten läuft es in einer Schleife: nach jeder Aktion springt es wieder zum Anfang.
Starten Sie das Programm nun mit RUN. Wenn Sie eine Cursortaste drücken, bewegt sich die Schildkröte in die entsprechende Richtung; andere Tasten außer "E" reagieren nicht. Die Schildkröte bewegt sich solange, bis Sie die Cursortaste wieder loslassen. Spielen Sie ruhig etwas mit dem Programm, bis Sie die Schildkröte ganz sicher über die Platte bewegen können.
Damit wäre der erste Schritt der Teach-In-Programmierung getan: die schrittweise Bewegung der Schildkröte über die gewünschte Route. Jetzt muß sich der Computer den Weg noch merken. Dazu lassen wir ihn einfach wieder eine Tabelle aufstellen, genauso wie bei der Programmierung des Roboters. Wir legen ein Feld A$(...) an, in das die Bewegungsbefehle geschrieben werden. Geben Sie Folgendes ein:

```
30 DIM A$(500)
80 I=1
```

I ist der Zähler für die Plätze im Feld A$. In jeden Platz des Feldes A$ kommt nun ein Kommando, genauso wie wir es bislang in die DATA-Zeilen geschrieben haben. Das Kommando muß aufgrund der Steuerbewegungen der Schildkröte konstruiert werden. Überlegen wir uns ein Beispiel:
Angenommen Sie fahren die Schildkröte ein Stück vor, haben z.B. fünfmal die Taste Cursor hoch gedrückt. Programmtechnisch wäre es jetzt am einfachsten, das Programm würde in fünf aufeinanderfolgende Plätze des Feldes A$ jeweils "1V" einschreiben. Dies wäre aber Platzverschwendung und würde auch den Programmablauf verlangsamen. Wir sammeln daher gleiche Bewegungen erst einmal auf, um dann in dem genannten Beispiel an einen Platz des Feldes A$ die Aktion "5V" zu schreiben. Dies macht unser Programm etwas komplizierter, lohnt aber die Mühe.
Ab Zeile 90 wird das Programm entsprechend erweitert:
Mit der Variablen W% wird die Anzahl gleicher Schritte gezählt. Außerdem wird noch eine Kopie des Cursorcodes angelegt. Sollte der aktuelle Cursorcode irgendwann nicht mehr mit dieser Kopie übereinstimmen, ist die Folge gleicher Kommandos beendet und es muß in die Tabelle A$ geschrieben werden. Dies erledigt das Unterprogramm ab Zeile 900. Dort werden die vereinbarten Kommandos aus der Schrittzahl und dem Kennbuchstaben zusammengesetzt. Der Schrittzähler wird wieder zurückgesetzt und außerdem eine neue Kopie des Cursorcodes angelegt. Der Zeiger in das Feld A$ wird auch erhöht.

```
90 W%=0
110 IF K$<>AUF$ AND K$<>AB$ AND
      K$<>RE$ AND K$<>LI$ AND
      K$<>"E" AND K$<> "e" THEN
      GOTO 100
120 IF L$<>K$ THEN GOSUB 900
150 W%=W%+1
190 W%=W%+1
230 W%=W%+5
270 W%=W%+5
300 REM Teach-In Modus beendet
310 A$(I)="E"
900 IF L$="" THEN L$=K$ : RETURN
910 IF L$=AUF$ THEN CODE$="V"
920 IF L$=AB$ THEN CODE$="Z"
930 IF L$=RE$ THEN CODE$="R"
940 IF L$=LI$ THEN CODE$="L"
950 A$(I)=STR$(W%)+CODE$
960 I=I+1
970 W%=0
980 L$=K$
```

Wir hätten bei der Numerierung der Feldplätze auch bei I=0 anfangen können. Dies wurde nicht getan, weil eines der Programme auf Diskette I=0 als Spezialfall behandelt und wir aber die Felder einheitlich benutzen wollten.

In Zeile 950 erscheint der Ausdruck STR$(W%).
Die Funktion STR$ dient dazu, einen Zahlenwert wie hier die Anzahl der Schritte in der Variablen W% in eine Zeichenkette umzuwandeln, d.h. in diesem Fall in eine Folge von Dezimalziffern.

990 RETURN

Starten Sie das Progamm jetzt mit RUN und lassen die Schildkröte fünf Schritte vorwärts fahren. Bislang merken Sie keinen Unterschied zu dem vorigen Programm. Nach "E" geben Sie ein:

PRINT A$(1)

Auf dem Bildschirm erscheint

5V

Dies ist der erste erzeugte Befehl. Jetzt wollen wir den erlernten Weg selbständig von der Schildkröte abfahren lassen. Dazu erweitern wir unser Programm um ein ähnliches Programmstück wie bei der Ausführung der DATA-Zeilen (s. voriges Kapitel). Das Lesen der DATA-Zeilen wird jetzt durch ein Lesen des Feldes A$ ersetzt. Ausserdem wurde der Programmteil etwas kürzer formuliert. Die zusätzlichen Zeilen werden anstelle des END-Kommandos eingeschoben:

```
320 REM Ausführteil
330 I=1
340 W%=VAL(A$(I))
350 K$=RIGHT$(A$(I),1)
360 IF K$="V" THEN CALL
      ITV(W%,TS,E5)
370 IF K$="Z" THEN CALL ITZ(W%,TS)
380 IF K$="R" THEN CALL ITR(W%,TS)
390 IF K$="L" THEN CALL ITL(W%,TS)
400 IF K$="E" THEN END
410 I=I+1
420 GOTO 340
```

Geben Sie die Programmzeilen ein. In der Zeile 330 wird der Zähler I wieder auf den Anfang des Feldes A$ gesetzt. Danach wird jeder Befehl gelesen und ausgeführt (Zeile 340-420).
Starten Sie nun das Programm mit RUN. Wir sind jetzt im Eingabe- oder Lernteil. Fahren Sie den Weg (das Viereck) mit der Schildkröte ab, indem Sie die entsprechenden Cursortasten drücken. Wenn Sie zwischendurch vom Weg abkommen und neu anfangen wollen, drücken Sie die Ctrl-Break-Taste, setzen die Schildkröte wieder auf den Ausgangspunkt und beginnen das Programm erneut mit RUN.
Am Ende steht die Schildkröte wieder auf dem Startpunkt. Geben Sie jetzt "E" ein. Damit lassen wir sie das Erlernte ausführen und schon fährt sie los - genau auf dem

Weg, den wir ihr beigebracht haben. Wenn Sie die gleiche Bahn nochmals sehen wollen, geben Sie

```
GOTO 320
```

ein - nicht RUN, denn dann würde das Feld A$ wieder gelöscht werden. Durch die Kreisfahrten der Schildkröte wickelt sich das Anschlußkabel immer mehr auf. Vor jedem Start sollte man deshalb die Schildkröte zurückdrehen, damit sich das Kabel frei bewegen kann.
Probieren Sie nun neue Wege aus oder erweitern das Programm. Sie können z.B. Fehleingaben rückgängig machen, wenn Sie vom Weg abgekommen sind oder den Weg rückwärts durchfahren oder die Route auf dem Bildschirm gleichzeitig darstellen. Speichern Sie aber zunächst das Programm, denn es wird im folgenden Kapitel ausgebaut. Ein komfortables Programm mit Bildschirmanzeige finden Sie auf Diskette unter dem Namen ROUTEACH.BAS. Es kann zudem die Routen auf der Diskette abspeichern und wieder laden, auf dem Drucker oder dem Bildschirm ausgeben.

In Kapitel 6.3 haben wir bereits die Bildschirmgrafik kennengelernt. Mit einem Zeichenstift - der Grafik-Schildkröte - konnten wir einzelne Linien und Punkte und auch ganze Figuren auf den Grafikschirm zeichnen. Diese Bildschirmgrafik wollen wir jetzt für die Routenplanung der Schildkröte einsetzen. Am Bildschirm wird die gewünschte Fahrspur mit der Grafik-Schildkröte erstellt, nach der die Schildkröte dann fahren soll. Welche Vorteile bringt das? Für uns keinen; wir können uns Zeit beim Experimentieren lassen und schon in der Lehrphase die Schildkröte benutzen. Anders in der Industrie. Die laufende Produktion mit Robotern müßte für die Lehrphase unterbrochen werden. Daher bedeutet es schon einen gewaltigen Vorteil, wenn die Bewegung des Roboters schon am Bildschirm einstudiert werden kann.
Wenige Änderungen des Teach-In-Programms des letzten Kapitels genügen, um vom Teach-In-Programm zum CAD-Programm zu kommen. Es genügt, die Schildkrötenbefehle während der Lehrphase von der echten Schildkröte auf die Grafik-Schildkröte umzulenken. Vorher muß natürlich noch die Bildschirmgrafik eingeschaltet werden. Bringen wir folgende Änderungen in dem Programm an:

Dieses System finden wir in der Industrie unter dem Namen CAD-Programmierung (CAD = Computer Aided Design). Übrigens keine leichte Aufgabe für das Computersystem: dem Roboterprogrammierer muß ja ein realistischer Eindruck wie bei einer Fernsehaufzeichnung geboten werden, damit er den Bildschirm-Roboter zuverlässig führt. Bei Detailproblemen muß er mit seiner Bildschirmanzeige näher heranfahren können. Auch die Umgebung des Roboters muß auf dem Bildschirm dargestellt werden. Es wäre verhängnisvoll, wenn ein Roboter eine andere Maschine streifen würde, bloß weil sie während der Programmierung auf dem Bildschirm nicht zu sehen war.
Da haben wir es mit der Schildkröte schon leichter.

Wenn Sie einen Monochrom-Schirm benutzen, tauchen wieder zwei zusätzliche Kommandos auf:

325 CALL IHA
400 IF K$="E" THEN CALL IHE : CALL IGA : SCREEN 0 : CALL IW80 : END

(s. Anhang A1.4)

72 SCREEN 1 : CALL IGE
140 CALL IGV(S%)
180 CALL IGZ(S%)
220 CALL IGR(D%)
260 CALL IGL(D%)
400 IF K$="E" THEN CALL IGA : SCREEN 0 : CALL IW80 : END

Probieren Sie das Programm aus. Das Lehren der Route geht am Bildschirm nun sogar wesentlich flüssiger. Bei Betätigen der Taste E setzt sich dann die richtige Schildkröte in Bewegung.
Verbessern Sie Ihr Programm, indem Sie sich die Planquadrate der Schildkrötenwelt auf den Bildschirm holen. So können Sie leichter den Bewegungsspielraum abschätzen.

73 F$="TURTLE"
74 CALL IGLOAD(F$)
75 CALL IGC

Wer will, kann die Grafik-Schildkröte vor dem Wiederhollauf auf die Ausgangsposition zurücksetzen, die Stiftfarbe umschalten und zusätzlich zu der Schildkröte die Route auf dem Bildschirm malen. So können Sie verfolgen, wo sich die Schildkröte jeweils befindet. Ein anderer Vorschlag: Erweitern Sie das Programm so, daß Sie zunächst auf dem Bildschirm die Lage von Hindernissen aufzeichnen. Konstruieren Sie dann mit dem CAD-Verfahren den Weg zwischen den Hindernissen hindurch. Wenn Sie keinen Fehler gemacht haben, sollte auch die echte Schildkröte zwischen den echten Hindernissen hindurchfinden.
Wir haben in diesem Kapitel gesehen, wie man mit Hilfe des Computers am Bildschirm Zeichnungen - hier die Fahrroute - erstellen kann, die dann später ausgeführt werden. Auf Diskette finden Sie auch zu diesem Thema ein fertiges Programm unter dem Namen ROUTEDIT.BAS. Es ist mit dem hier entwickelten Programm nicht sehr eng verwandt. Nach Konstruktion der Route am Bildschirm kann nicht sofort in den Ausführbetrieb übergewechselt werden. Vielmehr muß die Route auf einer Diskette abgespeichert werden. Ausgeführt wird sie mit dem Programm ROUTEACH.BAS, das ja Routen von der Diskette laden kann. Dafür entschädigt Sie ROUTEDIT.BAS mit einer Vielzahl von Fähigkeiten: nicht nur, daß Sie die Route konstruieren können; Sie können auch noch nachträglich Änderungen anbringen, Bahnstücke von Diskette an jeder beliebigen Stelle einfügen, Kommandos löschen usw. Nebenbei lernen Sie durch Studium dieses Programms, wie ein Editor funktioniert.

10 Die Schildkröte bekommt Fühler
10.1 Sensor für Hindernisse: Stoßstange

Industrielle Fahrroboter haben riesige Not-Stop-Bügel, die in erster Linie für den Personenschutz vorgesehen sind. Zur Orientierung benutzen sie andere Sensoren, z.B. ein Echolot auf der Basis von Ultraschall.

Daß 0 und 1 gegenüber unseren früheren Experimenten gerade vertauscht ist, hat seinen Grund in der Verkabelung des Tasters. Schauen Sie genau hin: Im Gegensatz zu anderen Anwendungen wird diesmal Kontakt 1 und 2 verwendet. Klar, bei dem gegebenen Einbau hätte in Kontakt 3 kein Stecker eingesteckt werden können. Ein weiterer Grund geht tiefer. In der industriellen Praxis werden alle Sicherheitsüberwachungen an den öffnenden Kontakt eines Tasters angeschlossen. Sollte durch einen Unfall die Leitung zum Taster beschädigt oder abgerissen werden, reagiert die Elektronik genauso, wie wenn der Taster gedrückt würde, also meist mit der Abschaltung der Anlage.

Die Programmierung der Schildkröte sah bisher so aus: Vorgabe der Route per Tabelle oder im Teach-In-Verfahren und anschließend Fahrt nach dieser Tabelle. Stellen Sie sich einen Fahrroboter in einer grossen Halle vor, der z.B. Pakete hin- und hertransportiert. Sein Weg ist natürlich auch vorgeschrieben. Plötzlich fällt ein Paket aus dem Regal genau in seinen Fahrweg. Was nun? Der Roboter kann es rammen und beiseite schieben oder darüber fahren. Besser wäre es natürlich, wenn er das Paket irgendwie erkennen würde und das Hindernis umgehen könnte. Dazu braucht er einen Sensor, einen Fühler, der z.B. auf Druck reagiert.
Unsere Schildkröte hat dafür einen Sensor: die Stoßstange. Sie ist vorne vor den Rädern angebracht und betätigt einen Mikroschalter, wenn man sie nach hinten drückt. Der Schalter ist am Eingang E5 des Interface angeschlossen. Seine Funktion läßt sich mit folgendem Programm überprüfen:

```
LOAD"INIT"
10 CALL IIN
20 CLS
30 LOCATE 1,1
40 CALL IDE(E1,E2,E3,E4,E5,E6,E7,E8)
50 PRINT E5
60 GOTO 30
```

Geben Sie die Zeilen ein, und starten Sie das Programm mit RUN. Am Bildschirm erscheint jetzt eine "1". Drücken Sie auf die Stoßstange, wechselt die Anzeige auf "0". Stoßstange frei bedeutet also: E5=1, Stoßstange vor Hindernis: E5=0.
Der Digitaleingang, an dem der Schalter liegt, wird mit CALL IDE (Zeile 40) abgefragt. Mit der Ctrl-Break-Taste halten Sie das Programm an.
Die Funktion der Stoßstange wollen wir jetzt bei fahrender Schildkröte testen. Sie soll sich solange vorwärts bewegen, bis sie an ein Hindernis stößt.
Legen Sie vor die Schildkröte in ca. 10 cm Abstand ein Buch als Hindernis. Geben Sie ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 S%=200 : CALL ITV(S%,TS,E5)
```

Die Schildkröte fährt vorwärts und stoppt, wenn die Stoßstange an das Buch stößt. Das Ganze geschah ohne zusätzliche Abfrage, denn das Kommando CALL ITV prüft schon selbst die Betätigung des Tasters. Die Null bzw. Eins des Tasters wird auch von dem Kommando CALL ITV ständig in die Variable E5 der Parameterliste ge-

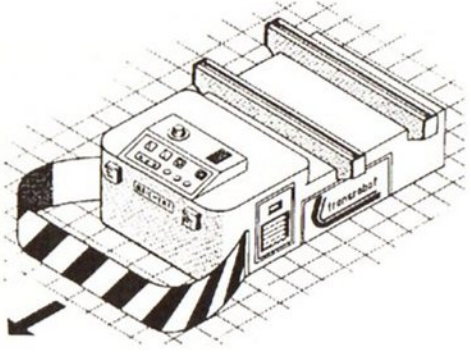


Bild 10.1: Industrieroboter mit Not-Stop-Bügel

schrieben. Sie können daher folgende Ergänzung vornehmen:

40 PRINT E5

Starten Sie das Programm mit RUN. Die Schildkröte fährt los. Sobald die Schildkröte an ein Hindernis stößt, bleibt sie stehen, und der Bildschirm zeigt eine Null. Wenn aber die Schildkröte unbehindert fahren konnte, bleibt sie nach einem Meter stehen und der Bildschirm zeigt eine Eins. Durch Abfragen von E5 kann also ermittelt werden, ob die geforderte Strecke ordnungsgemäß gefahren wurde oder ein Hindernis im Weg lag. Die Abfrage der Stoßstange wird nur von dem Kommando CALL ITV vorgenommen, denn die Schildkröte hat nur vorne eine Stoßstange. Insbesondere muß es der Schildkröte möglich sein, auch bei gedrückter Stoßstange mit dem Kommando CALL ITZ zurückzufahren, um sich wieder vom Hindernis zu entfernen.
Wie weit ist die Schildkröte gekommen, bis sie anstieß? Die Variable TS zeigt dies an. Sie enthält nach Beendigung des Kommandos CALL ITV die tatsächlich gefahrene Schrittzahl. Wurde der Weg in voller Länge gefahren, so enthält sie natürlich gerade den Zahlenwert, der im CALL ITV-Kommando angegeben war. Auch mit diesem Hilfsmittel kann festgestellt werden, ob die Schildkröte angestoßen ist.
Damit ist unsere Schildkröte selbständiger geworden; sie lernt, ihre Umwelt zu erkennen. Lassen wir sie ihre Welt, ihren Bewegungsraum auf der Platte, auf der sie fährt, ertasten. Dazu bauen wir "Mauern" aus Büchern um sie, so daß in der Mitte eine Fläche von ca. 40 cm x 40 cm übrigbleibt. In jeder Richtung soll die Schildkröte jetzt bis zur Mauer fahren und anhalten. Damit wir sie nicht jedesmal drehen müssen, soll sie anschließend von selbst in die nächste Richtung fahren.

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 S%=200 : CALL ITV(S%,TS,E5)
40 IF E5=1 THEN GOTO 30
50 S%=10 : CALL ITZ(S%,TS)
60 D%=90 : CALL ITR(D%,TS)
70 GOTO 30
```

Setzen Sie die Schildkröte parallel zu einer Wand und starten das Programm mit RUN. Sie wird jetzt nacheinander bis zu jeder Wand fahren, dort anhalten, sich um 90°

Fassen wir die verschiedenen Rückmeldungen der Schildkröte zusammen:

CALL ITX(TX)

Setzt in die Variable TX die derzeitige X-Position der Schildkröte.

CALL ITY(TY)

Setzt in die Variable TY die derzeitige Y-Position der Schildkröte.

CALL ITK(TK)

Setzt in die Variable TK den derzeitigen Kurs der Schildkröte.

CALL ITV(S%,TS,E5)

CALL ITZ(S%,TS)

CALL ITR(D%,TS)

CALL ITL(D%,TS)

Außer daß die Schildkröte bewegt wird, setzen die Kommandos in die Variable TS die Zahl der durchgeführten Schritte der Schildkröte. Das Kommando CALL ITV schreibt zusätzlich in die Variable E5 den Schaltzustand der Stoßstange ein.

drehen und weiterfahren. Vor der Drehbewegung muß sie etwas zurücksetzen, damit sie mit den Rädern nicht anstößt. Mit der Ctrl-Break-Taste läßt sie sich anhalten. Wenn wir die Schritte zwischen zwei gegenüberliegenden Wänden abfragen, wissen wir auch, wie groß der Raum ist. Geben Sie

```
25 FOR K=1 TO 4
45 IF K=3 THEN SB=TS
46 IF K=4 THEN SL=TS
70 NEXT K
80 PRINT "Breite:";SB*5+80;" mm"
90 PRINT "Länge:";SL*5+80;" mm"
```

ein, setzen die Schildkröte parallel zur Querwand mit Fahrtrichtung nach rechts und starten mit RUN. Sie wird jetzt alle vier Wände anfahren und stoppen. Dies erreichen wir mit der FOR...NEXT-Schleife in Zeile 25 (vier Durchläufe). Die Übertragung der Schrittzahl für die Breite erfolgt bei der Fahrt von rechts nach links, also bei K=3 (Zeile 70). Die Länge wird bei der Fahrt von vorn nach hinten gemessen, also bei K=4. Beide Werte werden in Millimeter umgerechnet, der Abstand zwischen Radachse und Stoßstange (40 mm) zweimal hinzugezählt und dann am Bildschirm angezeigt.

Sie werden feststellen, daß die Genauigkeit, mit der die Schildkröte ihre Welt auslotet, sehr empfindlich davon abhängt, ob sie auch wirklich exakt parallel zu den Begrenzungen ihrer Welt fährt. Wir können dies verbessern, wenn wir auch schon bei der ersten und der zweiten Kante die Position der Schildkröte aufzeichnen. Hierfür stehen uns noch weitere Schildkrötenkommandos zur Verfügung:

CALL ITX(TX)

speichert die derzeitige X-Position der Schildkröte in die Variable TX. Ähnliches gilt für

CALL ITY(TY)

Dieses Kommando setzt die Y-Position in die Variable TY. Auch der derzeitige Kurs der Schildkröte kann ermittelt werden:

CALL ITK(TK)

setzt den Kurs in die Variable TK. Die Positionsangaben beziehen sich immer auf Position und Kurs der Schildkröte zu dem Zeitpunkt, wo das CALL ITI-Kommando gegeben wurde. Es kann daher durchaus

sinnvoll sein, das CALL ITI-Kommando mehr als einmal im Programm zu benutzen, wenn man sich eine neue Ausgangslage wählen will.
Mit diesen Kommandos schreiben wir obiges Programm um:

```
43 IF K=1 THEN CALL ITY(OBEN%)
44 IF K=2 THEN CALL ITX(RECHTS%)
45 IF K=3 THEN CALL ITY(UNTEN%)
46 IF K=4 THEN CALL ITX(LINKS%)
80 PRINT "Breite:";(RECHTS%-
       LINKS%) *5+80;" mm"
90 PRINT "Länge:";(OBEN%-UNTEN%)
       *5+80;" mm"
```

Mit diesem Programm und mit Hilfe des Sensors "Stoßstange" kann die Schildkröte ihre Welt schon ganz munter untersuchen. Sie können natürlich das Programm noch erweitern, indem Sie die Welt der Schildkröte auf dem Bildschirm grafisch anzeigen.
Zur Demonstration gibt es auf Diskette ein Programm, das WELT.BAS heißt. Mit ihm können Sie ebenfalls die Schildkrötenwelt erforschen.

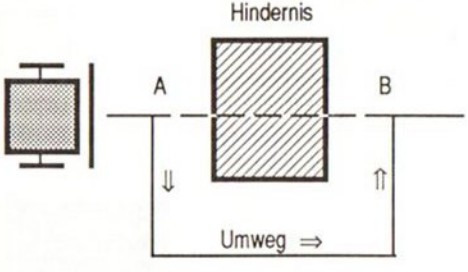


Bild 10.2: Fahrt um ein Hindernis.

10.2 Umfahren von Hindernissen: Achtung! Kollision

Den Tastsinn der Schildkröte haben wir im letzten Kapitel kennengelernt. Mit der Stoßstange als Fühler hat sie ihre Umwelt erforscht und die Grenzen ihres Bewegungsraumes erkannt. Jetzt wollen wir diesen Tastsinn dazu benutzen, daß die Schildkröte ein Hindernis erkennt und ihm ausweicht.
Wir benutzen wieder die Schildkröte mit der Stoßstange. Setzen Sie die Schildkröte auf die Fahrplatte aus Sperrholz oder Pappe und bauen ca. 10 cm vor ihr ein Hindernis (Buch) auf. Es sollte zunächst nicht breiter als die Schildkröte selbst sein. Sehen wir uns die Aufgabe in Bild 10.2 an.
Die Schildkröte soll von A nach B fahren und trifft auf ein Hindernis. Sie soll es rechts umfahren und dahinter wieder auf die ursprüngliche Route gelangen.
Das Programm beginnt mit der Fahrt bis zum Hindernis:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 S%=200 : CALL ITV(S%,TS,E5)
40 IF E5=1 THEN GOTO 30
```

Geben Sie die Zeilen ein und starten das Programm mit RUN. Die Schildkröte bleibt

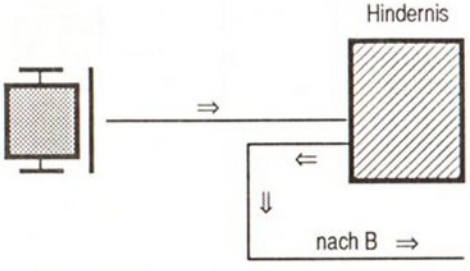


Bild 10.3: Ausweichen der Schildkröte nach rechts.

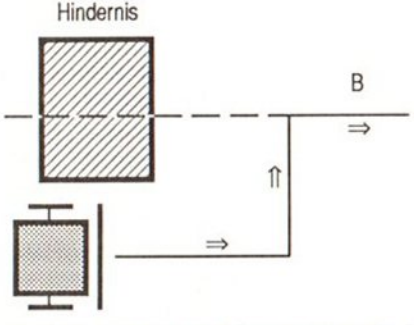


Bild 10.4: Rückfahrt der Schildkröte nach links.

am Hindernis stehen (E5=0). Zum Ausweichen muß sie nun etwas zurückfahren und sich um 90° nach rechts drehen. Jetzt fährt sie mindestens 12 cm vor (eine Schildkrötenbreite) und dreht sich wieder um 90° nach links. Hier setzt sie ihren Weg fort, um zum Ziel nach B zu kommen. Bild 10.3 zeigt das Umfahrungsmanöver.
Ergänzen wir unser Programm entsprechend:

```
200 REM Hindernis ausweichen
210 S%=10 : CALL ITZ(S%,TS)
220 D%=90 : CALL ITR(D%,TS)
230 S%=24 : CALL ITV(S%,TS,E5)
240 D%=90 : CALL ITL(D%,TS)
250 S%=34 : CALL ITV(S%,TS,E5)
```

Setzen Sie die Schildkröte zurück und starten das Programm mit RUN. Nach Zeile 240 steht sie neben dem Hindernis. Die Schildkröte setzt ihren Weg nach vorne fort. Als Maß für diesen Weg haben wir 34 gewählt; damit geht die Schildkröte die zehn Schritte wieder vor, die sie vor dem Schwenken zurücksetzte und dann nochmal um eine Schildkrötenbreite weiter.
Ist das Hindernis breiter als eine Schildkrötenbreite (12 cm), so steht die Schildkröte vor dem Hindernis und kann nicht weiter.

Wir wollen das Programm so schreiben, daß die Schildkröte ein Hindernis beliebiger Breite abtasten kann und dann ihren Weg am Hindernis vorbei fortsetzt. Da wir diese Aufgabe noch öfter brauchen werden, machen wir ein Unterprogramm daraus. Die Zeilennummern sind mit Bedacht schon passend gewählt worden.

```
50 GOSUB 200
190 END
260 IF TS<34 THEN GOTO 210
270 RETURN
```

Was jetzt noch fehlt, ist eine automatische Abtastung der Hindernislänge. Dazu benutzen wir natürlich auch wieder die Stoßstange und das gleiche Unterprogramm.

```
60 D%=90 : CALL ITL(D%,TS)
70 S%=34 : CALL ITV(S%,TS,E5)
80 IF E5=0 THEN GOSUB 200
```

Jetzt fehlt nur noch der Weg hinter dem Hindernis zurück auf die ursprüngliche Route. Bild 10.4 zeigt diesen Ablauf.
Wir ergänzen unser Programm:

```
90 CALL ITX(TX)
100 IF TX<0 THEN S%=ABS(TX)
```

In den USA, Japan und Europa werden ganze Wettbewerbe für selbstgebaute Schildkröten veranstaltet. Dabei geht es darum, daß ein solcher fahrender Roboter möglichst schnell durch ein Labyrinth findet.

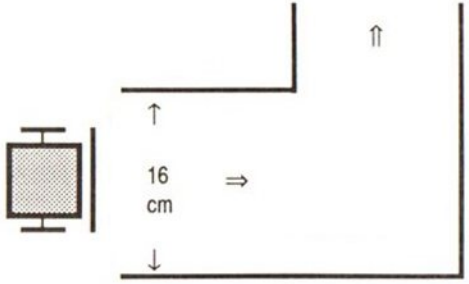


Bild 10.5: Labyrinth mit Abbiegung links.

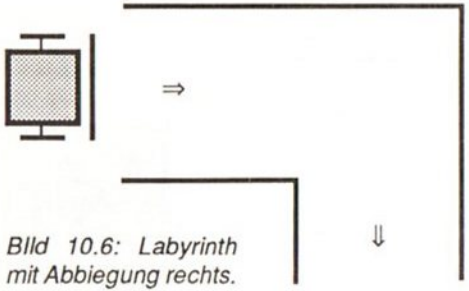


Bild 10.6: Labyrinth mit Abbiegung rechts.

```
: CALL ITZ(S%,TS)
110 IF TX>0 THEN CALL ITV(TX,TS,E5)
120 D%=90 : CALL ITR(D%,TS)
130 S%=10 : CALL ITV(S%,TS,E5)
140 PRINT "Hindernis umrundet."
```

In den Zeilen 90 bis 110 benutzen wir das Kommando CALL ITX um die Position zu ermitteln und auf jeden Fall auf die Ausgangsposition X=0 zurückzufinden. Anschließend wird auf den Originalkurs eingeschwenkt und geradeaus weitergefahren. Setzen Sie die Schildkröte nun wieder vor das Hindernis und starten das Programm mit RUN. Sie wird jetzt an jedem Hindernis vorbeikommen, egal wie lang es ist. Natürlich läßt sich noch einiges an dem Programm ausbauen - Sie können es z.B. so erweitern, daß die Schildkröte auch links am Hindernis vorbeifährt. Beachten Sie auch noch folgenden Grenzfall: Wenn die Schildkröte auf ihren Originalkurs zurückkehrt, kann es sein, daß sie so eng an der Rückseite des Hindernisses entlangfährt, daß sie nicht genügend Platz zum Drehen hat. Verbessern Sie das Programm, um auch diesen Fall zu berücksichtigen.
Ein fertiges Programm mit Benutzerführung finden Sie auf Diskette unter dem Namen HINDERNS.BAS.

10.3 Ertasten des Weges: Im Labyrinth

Für den folgenden Versuch benutzen wir wieder die Stoßstange unserer Schildkröte. Wir wollen die Möglichkeit, daß die Schildkröte Hindernisse erkennen und ihnen ausweichen kann, so ausnutzen, daß sie durch ein Labyrinth findet. Bevor wir das Labyrinth auf der Fahrplatte aufbauen, muß zunächst die erforderliche Straßenbreite für die Schildkröte ermittelt werden. Setzen Sie die Schildkröte mitten auf die Platte und lassen Sie sie einmal um ihre eigene Achse drehen:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 D%=180
40 CALL ITR(D%,TS) : CALL ITR(D%,TS)
```

Markieren Sie mit einem Bleistift den Platzbedarf bei der Drehung, die später in dem Labyrinth auch möglich sein muß. Es werden ca.16 cm Straßenbreite benötigt. Nun bauen wir die erste Labyrinthstrecke nach Bild 10.5 auf.
Sie besteht aus einer geraden Strecke, die nach ca. 20 cm nach links abbiegt. Begrenzt wird der Weg durch seitliche Wände, die wir z.B. durch Holzleisten (10mm x

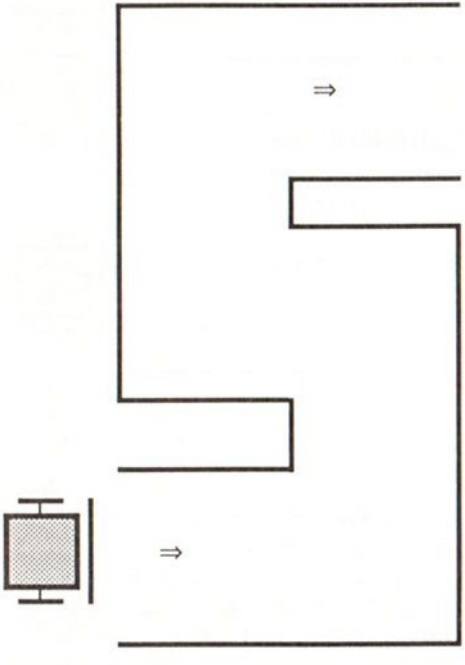


Bild 10.7: Einfaches Labyrinth.

10mm Querschnitt) herstellen können. Diese befestigen wir mit doppelseitigem Klebeband auf der Platte entlang der vorher aufgezeichneten Straßengrenzen. Ein- und Ausfahrt lassen wir offen.
Unser Programm soll in der Lage sein, die Schildkröte durch das Labyrinth bis zum Ausgang zu führen. Zunächst lassen wir sie vom Eingang aus vorwärts fahren, bis sie auf ein Hindernis stößt: die Querwand. Geben Sie ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 GOSUB 200
190 END

200 REM vor bis zur Wand
220 S%=200 : CALL ITV(S%,TS,E5)
240 IF E5=1 THEN GOTO 220
250 S%=8 : CALL ITZ(S%,TS)
260 RETURN
```

Die Vorwärtsbewegung mit Erkennung des Hindernisses legen wir in einem Unterprogramm ab (Zeile 200-260), da wir diesen Bewegungsteil später öfters benötigen. Mit Zeile 30 sprechen wir das Unterprogramm an. Bei einem Hindernis fährt die Schildkröte sofort acht Schritte zurück (Zeile 250), damit sie Platz für die anschließende Drehung hat.
Starten Sie das Programm mit RUN; die Schildkröte muß bis zur Wand vorfahren und zurücksetzen. Nun soll sie nach links oder rechts fahren. Wir gehen davon aus, daß sie die Richtung noch nicht kennt. Also lassen wir sie den Weg rechts und anschließend links prüfen, ob er frei ist. Natürlich kann man es auch in der anderen Reihenfolge machen. Wir erweitern unser Programm:

```
40 D%=90 : CALL ITR(D%,TS)
60 GOSUB 200
80 D%=180 : CALL ITL(D%,TS)
90 GOSUB 200
```

Setzen Sie die Schildkröte wieder an den Eingang und starten das Programm. An der Querwand dreht sie sich um 90° nach rechts (Zeile 40) und fährt vor. Dafür benutzen wir wieder das Unterprogramm ab Zeile 200. Ist der Weg frei, fährt sie ungehindert weiter. Bei einem Hindernis hält sie an und setzt zurück. Nun versucht sie die andere Richtung. Dazu dreht sie sich um 180° (Zeile 80) und fährt weiter vorwärts (GOSUB 200). Da hier der Ausgang unse-

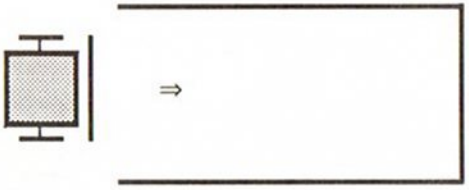


Bild 10.8: Sackgasse.

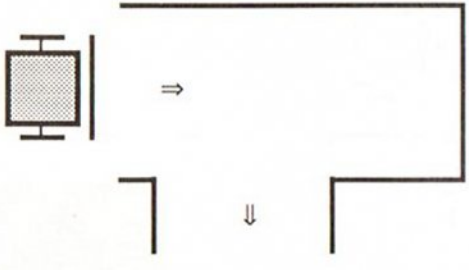


Bild 10.9: Sackgasse mit seitlichem Ausgang.

res Labyrinths ist, wird sie ungehindert weiterfahren. Anhalten läßt sie sich mit der Ctrl-Break+F10-Taste.
Prüfen wir, ob die Schildkröte auch den Ausgang rechts findet. Ändern Sie dazu die Strecke nach Bild 10.6 ab. Setzen Sie die Schildkröte an den Eingang, und starten Sie mit RUN.
Wir wollen nun sehen, ob wir mit diesem kleinen Programm auch durch ein längeres Labyrinth mit mehreren Abbiegungen hintereinander fahren können. Dazu bauen wir den Weg nach Bild 10.7 auf.
Damit das Programm bei jeder neuen Abbiegung auch wieder von vorn anfängt, müssen wir noch ergänzen:

```
70 IF T>20 THEN GOTO 40
100 IF T>20 THEN GOTO 40
210 T=0
230 T=T+TS
```

Im Unterprogramm werden die Vorwärtsschritte mit Hilfe des Zählers TS in T aufaddiert. Fährt die Schildkröte nach einer Rechtsdrehung ungehindert weiter (Zeile 60) und stößt bei der nächsten Abbiegung gegen die Querwand, würde sie eine Drehung um 180° machen (Zeile 80). Da sie aber bis dahin mehr als 20 Schritte gemacht hat, springt das Programm wieder zum Anfang (Zeile 70). Die gleiche Abfrage findet sich auch nach dem zweiten Unterprogrammaufruf (Zeile 100).
Geben Sie die Zeilen ein und starten Sie das Programm mit RUN. An jeder Abbiegung prüft die Schildkröte beide Richtungen und entscheidet sich immer für die freie. So findet sie nach kurzer Zeit den Ausgang. Halten Sie sie mit der Ctrl-Break+F10-Taste an.
Mit diesem einfachen Labyrinth wollen wir uns aber nicht begnügen. Neben dem richtigen Weg gibt es auch immer einen, der in eine Sackgasse führt. Unsere Schildkröte soll natürlich auch aus einer Sackgasse herausfinden. Bauen Sie zunächst die Strecke nach Bild 10.8 auf.
Nach Programmstart wird die Schildkröte bis zur Querwand am Ende fahren und sich nach rechts drehen. Hier findet sie keinen Ausgang, also dreht sie sich zur anderen Seite. Auch hier geht's nicht weiter: sie hängt fest! Erweitern Sie das Programm wie folgt:

```
40 W%=T-8
50 D%=90 : CALL ITR(D%,TS)
110 REM zurück aus der Sackgasse
120 D%=90 : CALL ITR(D%,TS)
```

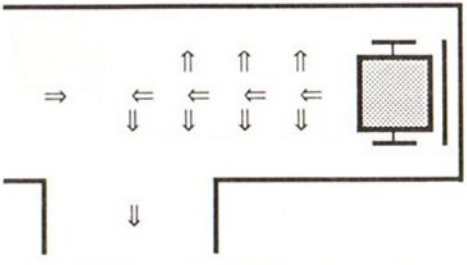


Bild 10.10: Ausfahrt aus der Sackgasse.

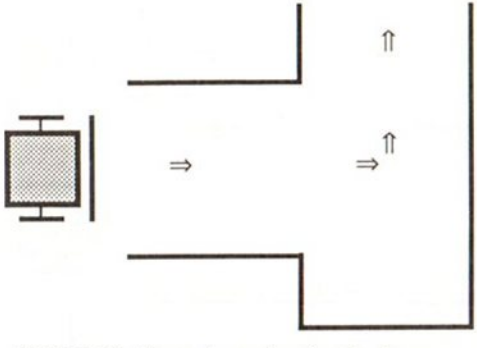


Bild 10.11: Abzweigung in eine Sackgasse.

130 CALL ITZ(W%,TS)

Die Schildkröte fährt zur linken Wand (Zeile 90). Stößt sie dagegen, hat sie bis dahin weniger als 20 Schritte gemacht. Das Programm geht bei Zeile 110 weiter. Hier dreht sie sich wieder in Fahrtrichtung und fährt den Weg aus der Sackgasse zurück. Die Schrittzahl hat sie sich vorher in W% gemerkt (Zeile 40).
Gehen wir noch einen Schritt weiter: der Ausgang aus der Sackgasse ist irgendwo seitlich, wie Bild 10.9 zeigt.
Auch diesen Weg soll die Schildkröte finden. Dazu sehen wir uns erst einmal an, wo die Schildkröte am Ende der Sackgasse (Bild 10.10) steht, bevor sie zurückfährt (Zeile 130).
Sie soll nicht direkt zurückfahren, sondern abwechselnd links und rechts prüfen, ob ein Ausgang vorhanden ist. Dies macht sie bis zum Sackgassenanfang - sie geht also höchstens W% Schritte zurück. Geben Sie die nötigen Programmzeilen ein:

```
130 S%=10 : CALL ITZ(S%,TS)
140 W%=W%-10
150 IF W%>20 THEN GOTO 50
160 CALL ITZ(W%,TS)
```

Die Schildkröte fährt 10 Schritte zurück und zieht diese von der Gesamtschrittzahl W% ab (Zeile 140). Jetzt prüft sie, ob hier ein Ausgang ist - aber nur, wenn sie sich noch nicht wieder am Eingang der Sackgasse befindet (W%<20).
Den Versuch, einen Ausgang zu finden, kennen wir schon: wir tun einfach so, als stände die Schildkröte vor einer Abbiegung (Programm ab Zeile 50). Sind beide Seiten zu, fährt sie weiter zurück wie in einer Sackgasse (ab Zeile 110).
Bauen Sie nun den Weg nach Bild 10.10 auf und starten die Schildkröte am Eingang. Sie findet den Ausgang aus der Sackgasse. Sollte sie dabei an den Wänden anstoßen, verbreitern Sie den Weg etwas oder verringern die Rückschritte z.B. auf 5 (Zeile 140). Dann tastet die Schildkröte die Wände in kleineren Abständen ab.
Nun fehlt noch eine letzte Labyrinthmöglichkeit: wenn an einem Abzweig der Weg in die eine Richtung in eine Sackgasse führt, der in der anderen weiterführt. Bild 10.11 zeigt den Verlauf.
Wenn die Schildkröte von links kommt, wird sie bis zur Querwand fahren (Zeile 30) und zurücksetzen. Liegt die Ausfahrt rechts, wird sie diese sofort finden, da sie diese zuerst prüft (Zeile 50). Bei einer Sackgasse

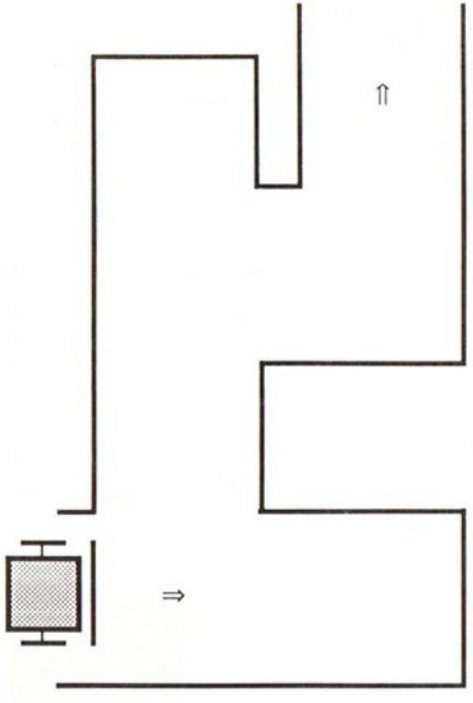


Bild 10.12: Schwieriges Labyrinth.

rechts sucht die Schildkröte rückwärts nach einem Ausweg (Zeile 130-160), bis sie wieder am Eingang der Gasse steht. Wir lassen sie nun einfach eine Kehrtwendung von 180° machen, damit sie auf dem freien Weg weiterfahren kann (Zeile 30).

```
170 D%=180 : CALL ITL(D%,TS)
180 GOTO 30
```

Geben Sie die Zeilen ein und testen die Schildkröte, ob sie sich richtig verhält. Bauen Sie die Strecke mit Hilfe der Holzleisten auf und probieren alle Möglichkeiten aus.

Zum Schluß wollen wir ein richtiges Labyrinth aufbauen, um alle Suchmöglichkeiten, die wir kennengelernt haben, von unserer Schildkröte ausführen zu lassen. Erstellen Sie sich z.B. ein Labyrinth nach Bild 10.12, das schon einige Schwierigkeiten aufweist. Schicken Sie die Schildkröte dort hinein. Wenn nichts schiefgeht, wird sie irgendwann am Ausgang ankommen. Ein fertiges Programm zum Durchfahren eines Labyrinths finden Sie wieder auf Diskette (LABYRINT.BAS).

Probieren Sie auch andere Strecken aus. Vielleicht - oder wahrscheinlich - ergeben sich doch noch Probleme, die die Schildkröte mit unserem Programm noch nicht lösen kann. Es kann z.B. passieren, daß die Schildkröte zwar wieder aus dem Labyrinth herausfindet, aber nicht zu dem Ausgang, den wir uns überlegt haben. Bild 10.13 zeigt ein solches Labyrinth, wo die Ausfahrt nach rechts übersehen wird, weil die Schildköte mit dem jetzigen Programm nach einem Ausgang nur sucht, wenn sie geradeaus nicht mehr weiter kann.

Noch schlimmer: Studieren Sie einmal, wie sich die Schildkröte bei dem Labyrinth aus Bild 10.14 verhalten würde.

Richtig - in diesem Labyrinth ist die Schildkröte gefangen und fährt immer im Kreis herum. Den Ausgang, der zur Freiheit führen würde, prüft die Schildkröte nicht, da sie vorzugsweise immer nach rechts abbiegt. Die Vorzugsrichtung "rechts" mit "links" zu tauschen bringt aber auch keine allgemeingültige Lösung. Die Schildkröte würde dann in einem gespiegelten Labyrinth nach Bild 10.13 festhängen.

Zum Erfolg führt die sogenannte "Rechte-Hand-Regel". Sie besagt, daß man immer wieder aus einem Labyrinth herausfindet, wenn man bei **jeder** möglichen Abzweigung nach Möglichkeit rechts abbiegt. Anschaulich: Beim Betreten des Labyrinths

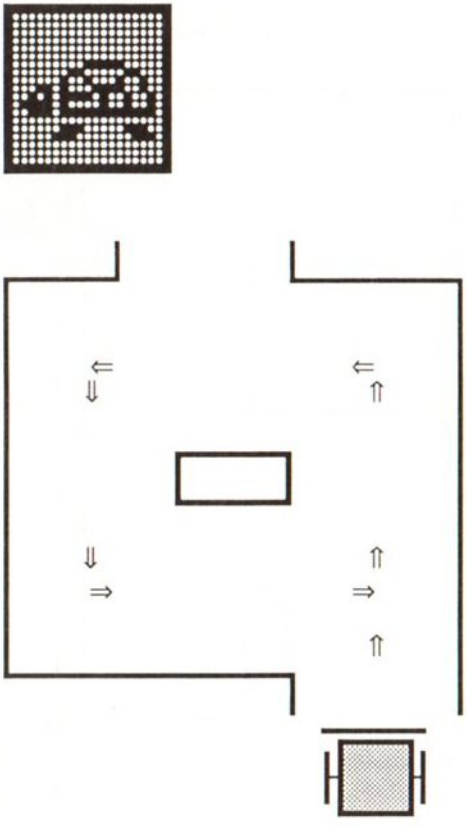


Bild 10.13: Bei diesem Labyrinth findet die Schildkröte nicht den oberen Ausgang.

berühren Sie mit der rechten Hand die rechts liegende Wand und laufen an dieser Wand entlang. Früher oder später werden Sie wieder aus dem Labyrinth herauskommen, vielleicht nicht bei dem erwarteten Ausgang oder auch nicht auf dem kürzesten Wege, aber Sie kommen heraus. Auch diese Strategie kann mit der Schildkröte programmiert werden. Da die Schildkröte keinen Fühler an der rechten Flanke besitzt, muß sie in regelmäßigen Abständen sich nach rechts wenden und fühlen, ob die Wand noch vorhanden ist. Wenn ja, geht es wieder zurück auf die Bahn, Linksschwenk und weiter. Wenn sich rechts eine Öffnung ergeben hat, geht es dort weiter. Das ständige Tasten macht die Bewegung der Schildkröte zwar langsam, aber dafür findet sie jetzt aus jedem Labyrinth. Und noch etwas: das Programm ist verblüffend kurz. Da wir vom bisherigen Programm nichts verwenden können, speichern und löschen wir es und geben neu ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 S%=10 : CALL ITV(S%,TS,E5)
40 IF E5=0 THEN GOTO 70
50 D%=90 : CALL ITR(D%,TS)
60 GOTO 30
70 REM angestoßen
80 CALL ITZ(TS,TS)
90 D%=90 : CALL ITL(D%,TS)
100 GOTO 30
```

Den Zahlenwert 10 in Zeile 30 müssen Sie vielleicht etwas anpassen. Ist er zu groß, trifft die Schildkröte vielleicht nicht jede Abzweigung nach rechts; das Problem hatten wir vorher schon beim Rückzug aus einer Sackgasse betrachtet. Zu klein darf der Wert aber auch nicht werden. Wenn die Schildkröte sich in einem geraden Gang befindet, muß gewährleistet sein, daß sie innerhalb der Zahl der Schritte in Zeile 30 auch tatsächlich die rechte Wand findet.
In Zeile 80 erscheint die Variable TS zweimal in der Parameterliste des Kommandos CALL ITZ. Der erste Eintrag ist die Anzahl der Schritte. Der zweite Eintrag ist der Rückgabewert der erfolgreich zurückgelegten Schritte.
Es steht Ihnen frei, das Programm mit einer Darstellung des Wegs der Schildkröte auf dem Bildschirm auszustatten. Das Programm STRATEGY.BAS auf der Diskette macht dies ebenfalls.
Bleibt noch die Frage wie die Schildkröte auf dem kürzesten Wege durch ein Laby-

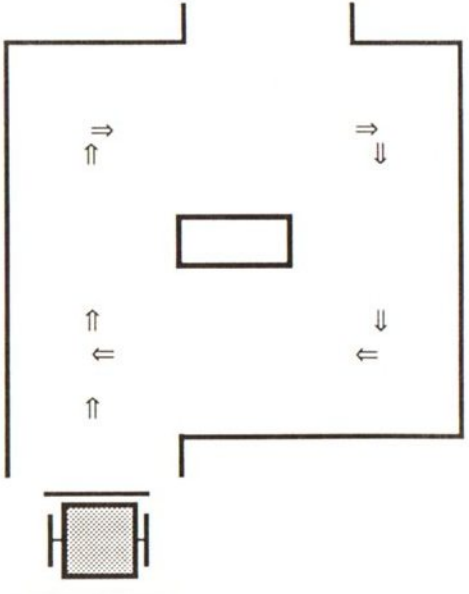


Bild 10.14: In diesem Labyrinth verirrt sich die Schildkröte.

rinth findet. Bei den eingangs angesprochenen Wettbewerben kommt es natürlich auf die Fahrzeit der Schildkröten an. Mit einfachen Programmen geht es allerdings hier nicht weiter. Wir wollen nur den Gedankengang kurz andeuten. Bei einem ersten Durchgang durch das Labyrinth tastet die Schildkröte möglichst viele Gänge ab. Daraus wird wiederum eine "Landkarte" konstruiert. Mit den mathematischen Methoden der Graphentheorie sucht nun das Programm auf der Landkarte die kürzeste Verbindung zwischen Start und Ziel heraus. Im zweiten Durchgang fährt die Schildkröte dann auf diesem Weg möglichst schnell die Strecke ab. Die Schildkröten sind übrigens meist mit berührungslosen Abstandsfühler, einer Art Echolot, ausgestattet, mit denen sie auch bei hoher Geschwindigkeit auf der Bahn bleiben, Hindernisse im voraus erkennen und seitliche Abgänge ermitteln, ohne sich ihnen hinwenden zu müssen. Sinnvoll wären seitliche Sensoren sicherlich auch für unsere Schildkröte. Vielleicht läßt sich die Schildkröte mit Tastern und weiteren Bauteilen aus Ihren anderen fischertechnik-Baukästen erweitern?

Solche Schildkröten sind keineswegs nur eine Spielerei. Industriefirmen und Universitäten haben schon Schildkröten gebaut und untersucht. Ziel dieser Forschungen ist die Entwicklung selbständiger Fahrautomaten in der Industrie und beim Katastrophenschutz. Vielleicht wird aber daraus auch einmal ein Haushaltsroboter, der alleine staubsaugen, rasenmähen und andere Arbeiten verrichten kann.

10.4 Sensor für Licht: Hell und dunkel

Neben dem Tastsensor "Stoßstange" besitzt die Schildkröte noch einen optischen Sensor. Dieses Auge, ein Fotowiderstand, ist über der Achse der Schildkröte angebracht. Sie finden den Fotowiderstand versteckt hinter einer Blende, die das Sichtfeld des optischen Sensors begrenzen soll. Dadurch sieht die Schildkröte gerade soviel wie nötig und wird nicht von seitlichen Helligkeitsunterschieden beeinflußt.
Der Fotowiderstand ist am Analogeingang EX des Interfaces angeschlossen (orangefarbenes Kabel); das Meßprinzip hatten wir bereits in früheren Versuchen kennengelernt. Ob die Lichtmessung funktioniert, können wir mit den Befehlen

```
LOAD"INIT"
10 CALL IIN
40 CALL IEX(EX)
50 PRINT EX
```

überprüfen. Wenn Licht auf den Fotowiderstand trifft, wird eine kleine Zahl (30...100) am Bildschirm angezeigt. Halten Sie die Sensoröffnung zu, so daß kein Licht einfällt, liegt der Wert ziemlich hoch, bei ca. 300.
Wir wollen die Lichtmessung jetzt mit der Schildkrötenbewegung verbinden: bei Licht soll die Schildkröte losfahren, bei Dunkelheit stoppen. Das Programm dazu sieht so aus:

```
20 CALL ITI
30 S%=1
50 IF EX<128 THEN CALL ITV(S%,TS,E5)
60 GOTO 40
```

Geben Sie die Zeilen ein und starten das Programm mit RUN. Die Schildkröte fährt los, wenn der Raum hell beleuchtet ist. Halten Sie einen Gegenstand oder Finger vor den Lichtsensor, bleibt die Schildkröte stehen. Im Programm wertet Zeile 50 den Helligkeitsunterschied aus; die Schaltschwelle liegt bei einem Wert von 128. Durch Anpassen dieser Zahl können Sie auch mit dem Lichtschalter in Ihrem Zimmer die Schildkröte in Gang setzen. Dabei sollte aber nicht allzuviel Tageslicht auf den Sensor treffen.
Mit Licht läßt sich auch die Fahrtrichtung unserer Schildkröte steuern. Machen wir sie zunächst einmal lichtscheu, d.h. sie soll sich vom Licht abwenden. Ändern Sie das Programm:

```
30 D%=5
50 IF EX<128 THEN CALL ITR(D%,TS)
```

In der Praxis benutzt man ebenfalls Licht zur Steuerung von Fahrrobotern. Mit einem Lichtstrahl lassen sich Wege markieren oder Hindernisse erkennen. Damit der Roboter aber nicht zu empfindlich auf Tageslicht oder andere Lichtquellen reagiert, wird unsichtbares Infrarotlicht verwendet, das der Empfänger aus allem anderen Licht herausfiltert. Infrarotlicht folgt am langwelligen Ende des Lichtspektrums auf das sichtbare Licht.

Stellen Sie die Schildkröte so, daß sie ins Licht schaut, z.B. zum Fenster und starten mit RUN. Sie dreht sich solange, bis es ihr dunkel genug ist. Umgekehrt geht's auch. Drücken Sie die Ctrl-Break-Taste, und geben Sie ein:

```
50 IF EX>128 THEN CALL ITL(D%,TS)
```

Nach RUN dreht sie sich aus dem Dunklen wieder zum Licht.
Jetzt steuern wir die Schildkröte mit einer beweglichen Lichtquelle. Sie soll einer Taschenlampe folgen, die wir vor ihr herführen.
Ergänzen Sie nun das Programm wie folgt:

```
50 IF EX>128 THEN GOTO 90
60 S%=1 : CALL ITV(S%,TS,E5)
70 GOTO 40
90 L=EX
100 D%=5 : CALL ITR(D%,TS)
110 CALL IEX(EX)
130 IF EX<L THEN GOTO 40
140 D%=10 : CALL ITL(D%,TS)
150 CALL IEX(EX)
170 IF EX<L THEN GOTO 40
180 D%=10 : CALL ITR(D%,TS)
190 GOTO 110
```

und starten es mit RUN. Richten Sie den Lichtstrahl aus ca. 20 cm Abstand direkt auf den Sensor. Die Schildkröte sucht jetzt die Lampe; sie dreht sich nach links und rechts. Erkennt sie den Lichtstrahl, fährt sie vorwärts auf ihn zu (Zeile 60). Wir haben die Schaltschwelle auf 128 gelegt (Meßwert bei den genannten Bedingungen), damit die Steuerung nicht zu empfindlich wird.
Die Schildkröte bewegt sich solange nach vorn, bis die Lichtmessung einen Wert über 128 ergibt, es also dunkler wird. Das Programm merkt sich die letzte Lichtstärke (Zeile 90) und prüft zunächst nach rechts (Zeile 100-130), ob es hier heller ist. Wenn das der Fall ist, also die Taschenlampe jetzt aus dieser Richtung strahlt, fährt sie dorthin weiter. Ansonsten dreht sie sich zurück und prüft die Lichtstärke in der anderen Richtung (Zeile 140-170). In dieser Suchschleife bleibt die Schildkröte solange, bis wieder genügend Licht auf den Sensor fällt und sie vorwärts fahren kann. Anhalten läßt sie sich mit der Ctrl-Break-Taste.
Damit die Schildkröte auch erkennt, wann die Taschenlampe ausgeschaltet wird, ergänzen wir das Programm:

```
80 N=0
120 N=N+1
```

```
160 N=N+1
190 IF N<5 THEN GOTO 110
```

Die Variable N zählt die Suchvorgänge nach Licht. War die Suche in beiden Richtungen zusammen fünfmal erfolglos, wird das Programm beendet.
Wir haben in diesem Kapitel gesehen, wie die Schildkröte mit Hilfe des Lichtsensors gesteuert werden kann. Er läßt sich natürlich noch weit vielfältiger einsetzen. Das kommt in den nächsten Abschnitten.

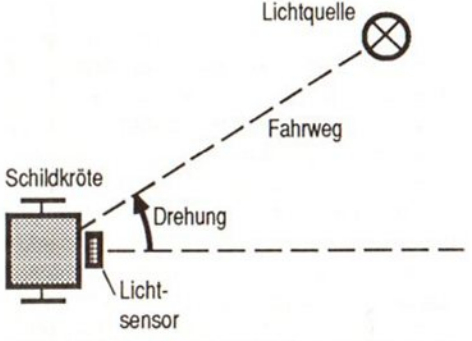


Bild 10.15: Die Schildkröte sucht nach Licht.

10.5 Suchen nach Licht: Augen auf!

Im letzten Kapitel haben wir gesehen, wie man den optischen Sensor zur Steuerung der Schildkröte benutzen kann. Mit einem einfachen Programm war es bereits möglich, daß die Schildkröte einer beweglichen Lichtquelle folgt. Dabei wurde nur zwischen Hell und Dunkel unterschieden.
Wir wollen jetzt die Lichtmessung exakter durchführen und die Schildkröte nach Licht suchen lassen. Sie soll sich ein Abbild der Helligkeitsverteilung in ihrer Umgebung machen und dabei die hellste Lichtquelle finden und anfahren können. Bild 10.15 zeigt den Ablauf.
Zunächst lassen wir die Schildkröte ihre Umgebungshelligkeit messen. Sie soll sich um 360° drehen und nach jedem Drehschritt die Helligkeit messen und ausdrukken. Geben Sie folgendes ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL ITI
30 H=1000
40 R%=0
50 FOR W=0 TO 355 STEP 5
60 CALL IEX(EX)
70 IF EX<H THEN H=EX : R%=W
80 PRINT "Winkel ";W;
       " : Helligkeit ";EX
```

```
90 D%=5 : CALL ITR(D%,TS)
100 NEXT W
```

Die Winkelrichtung der Schildkröte mit der bislang größten Helligkeit wird in der Variablen R% festgehalten, der Meßwert für die Lichtstärke in H.
Starten Sie nun das Programm mit RUN. Die Schildkröte dreht sich rechts herum und mißt bei jedem Drehschritt die Lichtstärke (Zeile 60). Außerdem werden Winkelrichtung W und der Wert EX auf dem Bildschirm angezeigt (Zeile 80). Hat die Schildkröte sich um 360° gedreht, ist das Programm zu Ende.
Jetzt haben wir ein "Lichtbild" der Schildkrötenumgebung. Die kleinsten Zahlenwerte entsprechen den größten Lichtstärken. Die Richtung mit der größten Lichtintensität steht in der Variablen R%. Fügen Sie die folgende Zeile hinzu:

```
110 PRINT "Größte Helligkeit in
      Richtung ";R%
```

In diese Richtung soll die Schildkröte nun fahren. Dafür drehen wir sie zunächst wieder in die Anfangsstellung (Bild 10.15) zurück mit:

```
120 D%=180
130 CALL ITL(D%,TS) :CALL ITL(D%,TS)
```

Starten Sie das Programm mit RUN. Die Schildkröte dreht sich wieder um 360° und mißt die Lichtstärken. Nachdem die Richtung mit der größten Helligkeit auf dem Bildschirm angezeigt wurde, dreht sich die Schildkröte wieder in ihre Grundstellung.
Nun drehen wir die Schildkröte in die ermittelte Richtung mit

```
140 CALL ITR(R%,TS)
```

Mit RUN führt sie den Befehl aus, nachdem sie erneut die Helligkeit gemessen hat. Die Schildkröte fährt jetzt erst in Grundstellung (0°) und dann in die hellste Richtung. Dort kann sie auch direkt bei der Rückdrehung stehen bleiben. Fassen Sie daher die Zeilen 120 bis 140 zusammen in:

```
120 D%=360-R%
130 IF D%<360 THEN CALL ITL(D%,TS)
```

Nun findet die Schildkröte die richtige Richtung schneller. Die IF-Anweisung in Zeile 130 vermeidet eine Fehlermeldung, die bei gleichmäßiger Beleuchtung auftreten wür-

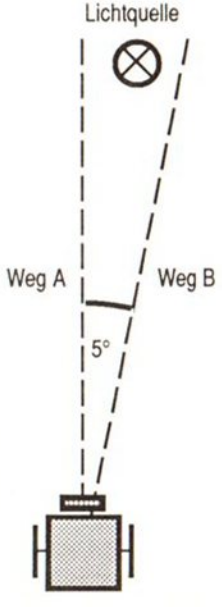


Bild 10.16: Winkelauflösung beim Messen.

de (D%=360). Mit der nächsten Zeile:

140 S%=200 : CALL ITV(S%,TS,E5)

kann man die Schildkröte zur Lichtquelle fahren lassen.
Wenn Sie den Versuch auf Ihrem Tisch zu Hause ausführen, werden Sie feststellen, daß sich die Schildkröte immer zum Fenster dreht, weil dort (am Tage) das meiste Licht ist. Dabei ist es übrigens unerheblich, in welche Richtung die Schildkröte vor dem Programmstart zeigt.
Leuchten Sie mit einer Taschenlampe aus kurzer Entfernung auf die Schildkröte und starten das Programm. Sie wird darauf zufahren. Anhalten läßt sie sich mit der Ctrl-Break+F10-Taste oder durch Antippen der Stoßstange.
Wie wir in einem früheren Kapitel gelernt haben, ist der kleinste Drehschritt der Schildkröte 5°. Wenn sich die Lichtquelle in einiger Entfernung zur Schildkröte befindet, kann es sein, daß sie mit unserem Programm daran vorbeifährt. Die Auflösung ist zu grob, wie Bild 10.16 verdeutlicht: Auf beiden Wegen (A und B) kann sie die Lichtquelle nicht erreichen. Wir müssen also eine oder mehrere Richtungskorrekturen auf dem Weg zur Lichtquelle durchführen. Die Schildkröte fährt dann jeweils ein Stück vor, ermittelt in einem bestimmten Winkel erneut die Richtung mit der größten Helligkeit und setzt ihren Weg dorthin fort. Bild 10.17 zeigt diesen Vorgang.
Erweitern wir das Programm:

```
140 S%=20 : CALL ITV(S%,TS,E5)
150 D%=15 : CALL ITL(D%,TS)
160 H=1000
170 R%=0
180 FOR W=0 TO 30 STEP 5
190 CALL IEX(EX)
200 IF EX<H THEN H=EX : R%=W
210 D%=5 : CALL ITR(D%,TS)
220 NEXT W
230 D%=30-R% : CALL ITL(D%,TS)
240 GOTO 140
```

Nach RUN wird die Schildkröte jetzt die Kurskorrektur in Richtung Lampe vornehmen. Die Einzelschritte sind in Bild 10.18 dargestellt:

1: 20 Schritte vor,
2: 15° nach links drehen (Meßanfang),
3: 30° nach rechts drehen und Richtung mit größter Helligkeit merken,
4: In diese Richtung zurückdrehen,

Bild 10.17: Richtungskorrektur bei der Anfahrt auf die Lichtquelle.

5: Weiterfahren.

Der Merkvorgang für die Richtung mit der größten Helligkeit entspricht dem in Zeile 30 bis 100. Sie können diesen Programmteil auch getrennt testen. Überspringen Sie den ersten Teil des Programms, indem Sie nach Initialisieren des Interface die Zeile:

25 GOTO 150

einfügen.
Richten Sie eine Taschenlampe aus einiger Entfernung auf die Schildkröte und setzen diese nicht genau in Richtung Lampe. Sie wird nach einigen Zwischenmessungen und -korrekturen exakt in Richtung Lichtquelle fahren.
Damit die Schildkröte an der Lampe, die auf der Fahrplatte steht, anhält, müssen wir nochmals auf den Sensor "Stoßstange" zurückgreifen. Mit

145 IF E5=0 THEN END

fährt sie nur noch bis zum "Hindernis" Lampe. Wenn die Stoßstange betätigt wird (E5=0), erfolgt in Zeile 145 ein Programmstop.
Damit wollen wir die Programmierung zu diesem Versuch abschließen. Natürlich gibt es noch einige Verbesserungen: man kann den Weg der Schildkröte am Bildschirm anzeigen oder das Helligkeitsbild auf einem Radarschirm darstellen, wie wir es bei einem der ersten Versuche mit dem Fotowiderstand kennengelernt haben.
Zu diesem Versuch gibt es auch wieder ein fertiges Programm auf Diskette. Sein Name ist SUCHER.BAS. Dort wird am Bildschirm Aufbau und Ablauf des Versuchs "Schildkröte sucht nach Licht" demonstriert.

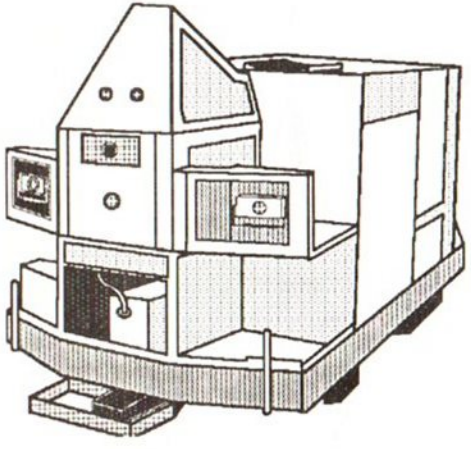


Bild 10.18: Industrieller Fahrroboter mit optischen und Ultraschall-Sensoren. Die Stoßstange und die Verkleidung sind teilweise entfernt. Die Lenkung dient nur dem manuellen Eingriff.

Bisher war der optische Sensor an der Schildkröte so angebracht, daß er Lichtstrahlen von vorn wahrnehmen konnte. Jetzt wird er als Lesekopf benutzt und tastet optisch eine Fläche auf der Fahrbahn vor der Schildkröte ab.

Zunächst bauen wir das entsprechende Modell nach der Bauanleitung um. Die Schildkröte besitzt jetzt keine breite Stoßstange mehr, sondern an der Vorderseite den Lesekopf, in dem sich der Fotowiderstand befindet. Er tastet das reflektierte Licht der Lampe von der Fahrbahn ab. Er sollte mindestens 3 mm Abstand von der Fahrbahn haben, damit das Licht unter den Sensor treffen kann. Justieren Sie den Lesekopf durch Verschieben der Bausteine entsprechend ein. Der Taster, der sich ursprünglich hinter der Stoßstange befunden hatte, ist jetzt an der Vorderfront des Lesekopfes angebracht und kann behelfsmäßig noch Hindernisse erkennen.

Prüfen wir zunächst wieder, ob das Modell richtig funktioniert. Für die folgenden Versuche benötigen wir unbedingt eine weiße Unterlage (z.B. Karton), auf der die Schildkröte fährt. Eine dunkle Fahrbahn reflektiert nicht genügend Licht, und das führt zu Fehlmessungen. Setzen Sie die Schildkröte nun auf die Unterlage und schließen Sie sie am Interface an. Die Lampe des Lesekopfes schließen Sie vorerst noch nicht an der 28-poligen Steckbuchse an. Nehmen Sie vielmehr ein 44 cm langes Kabel, und verbinden Sie die Lampe direkt mit den Steckern des Netzgeräts am Interface (in die Querlöcher einstecken). So leuchtet die Lampe dauernd. Geben Sie ein:

```
LOAD"INIT"
10 CALL IIN
20 CALL IEX(EX)
30 PRINT EX
```

Nach Programmstart erscheint eine Zahl, die der Helligkeit der Fahrbahn vor der Schildkröte entspricht. CALL IEX liest den Wert ein, der mit PRINT EX angezeigt wird. Setzen Sie die Schildkröte auf eine dunkle Fläche, so wird nach erneutem Start des Programms der Anzeigewert wesentlich größer sein. Eine helle Fläche ergibt also eine kleinere Zahl als eine dunkle.

Dies wollen wir jetzt für die Steuerung der Schildkröte ausnutzen. Sie soll einer dunklen Spur auf der Fahrbahn folgen. Um besten Kontrast zu erzielen, verwenden Sie mattschwarzen Klebestreifen (z.B. PVC-Isolierband) als dunkle Spur. Damit läßt sich dann ein Weg markieren, den die

Bild 10.19: Fahrspur für die Schildkröte.

Schildkröte fahren muß. Dieses Prinzip finden wir ebenso bei industriellen Fahrrobotern; z.T. wird hier allerdings auch mit Induktionsleitern im Boden gearbeitet; der Fahrroboter wird also gewissermaßen elektromagnetisch gelenkt.
Kleben Sie nun eine gerade Bahn von ca. 20 cm Länge auf die Unterlage. Auf dieser Bahn soll die Schildkröte fahren (Bild 10.19). Das Programm dafür sieht so aus:

```
10 CALL IIN
20 FOR I=1 TO 200
30 CALL I3R
40 NEXT I
50 CALL ITI
70 CALL IEX(EX)
80 H=EX
90 S%=1 : CALL ITV(S%,TS,E5)
100 CALL IEX(EX)
110 IF EX>H-20 THEN GOTO 90
```

Geben Sie die Zeilen ein. Verbinden Sie die Lampe des Lesekopfes nun wieder mit dem Ausgang M3 der 28-poligen Buchse. Setzen Sie die Schildkröte links auf die Bahn, so daß der Lesekopf auf den Klebestreifen zeigt, und starten Sie das Programm mit RUN.
Die Schildkröte fährt vor, bis der Streifen zu Ende ist. Die Fahrbahnlinie erkennt sie durch laufendes Messen des reflektierten Lichtes vom Boden. Dazu hat sie am Start die Lichtstärke der Bahn gemessen (Zeile 70) und sich diese gemerkt, also in H abgespeichert. Nach jedem Vorwärtsschritt wird erneut gemessen (Zeile 100). Dieser Wert EX wird in Zeile 110 mit dem vorherigen H verglichen. Solange EX nicht kleiner als H ist, befindet sich die Schildkröte noch auf der schwarzen Bahn. Sie fährt weiter vorwärts. Ist der Klebestreifen zu Ende, ist die reflektierte Lichtmenge vom hellen Untergrund größer als vorher - EX wird kleiner als H -, und die Schildkröte stoppt. Auch das Programm ist nun beendet.
Zwischen EX und H besteht ein Toleranzbereich von 20 (H-20), da auch die Meßwerte der Bahn etwas schwanken. EX muß somit erst um 20 kleiner werden als H, bevor die Schildkröte anhält. Ohne diese Sicherheit würde sie auf der Spur dauernd stoppen - probieren Sie's aus!
Wenn die Schildkröte nicht genau geradeaus fährt, kann sie zwischendurch die Bahn verlassen und stehen bleiben. Damit das nicht passiert, ergänzen wir das Programm um einen Korrekturteil, der die Schildkröte immer wieder auf den richtigen Weg führt. Geben Sie ein:

Bild 10.20: Richtungskorrektur auf die Spur.

Bild 10.21: Abbiegung nach links bzw. rechts.

```
120 D%=5 : CALL ITR(D%,TS)
130 CALL IEX(EX)
140 IF EX>H-20 THEN GOTO 90
150 D%=10 : CALL ITL(D%,TS)
160 CALL IEX(EX)
170 IF EX>H-20 THEN GOTO 90
```

Setzen Sie die Schildkröte, wie in Bild 10.20 gezeigt, etwas schräg auf die Spur am Fahrbahnanfang, und starten Sie mit RUN. Die Schildkröte wird nach kurzer Strecke die Bahn verlassen. Nun dreht sie nach rechts (Zeile 120) und mißt die Fahrbahnhelligkeit. Ist dieser Wert größer als der vorige - befindet sich dort also die Bahn -, fährt die Schildkröte weiter. Ansonsten dreht sie nach links und prüft mit der gleichen Methode, ob die Bahn dort verläuft (Zeilen 150 bis 170). Wenn nicht, ist das Programm zu Ende, da in diesem Fall auch das Ende der Bahn erreicht ist. Sie werden sehen, daß es der Schildkröte mit dieser Programmergänzung immer gelingt, auf der Bahn zu bleiben. Auch leichte Krümmungen in der Spur kann sie erkennen und verfolgen.

Ans Ende der Bahn bauen wir nun eine Abbiegung; der Einfachheit halber zunächst um 90° nach rechts oder links. Die Schildkröte soll auch hier den richtigen Weg finden. Wenn sie über das Ende der Spur hinausfährt, wird sie zunächst annehmen, sie sei vom Weg abgekommen. Die Schildkröte prüft nach rechts und links (je ein Schritt), ob der Weg dort weitergeht. Nach Bild 10.21 befindet sie sich aber an einer Abbiegung.

Die Schildkröte prüft jetzt in beiden Richtungen, wohin die Spur führt. Dazu fährt sie zunächst 13 Schritte vor, damit ihre Drehachse über der Bahn steht:

```
180 D%=5 : CALL ITR(D%,TS)
190 S%=13 : CALL ITV(S%,TS,E5)
```

Vorher dreht sie sich wieder in die ursprüngliche Fahrrichtung (Zeile 180). Jetzt dreht sie sich um 90° nach rechts:

```
200 D%=90 : CALL ITR(D%,TS)
```

und prüft anhand des Helligkeitsunterschieds, ob sich dort die Bahn befindet:

```
210 CALL IEX(EX)
220 IF EX>H-20 THEN GOTO 90
```

Wenn ja, fährt sie weiter vorwärts (Zeile 90). Ansonsten dreht sie sich in die andere Richtung:

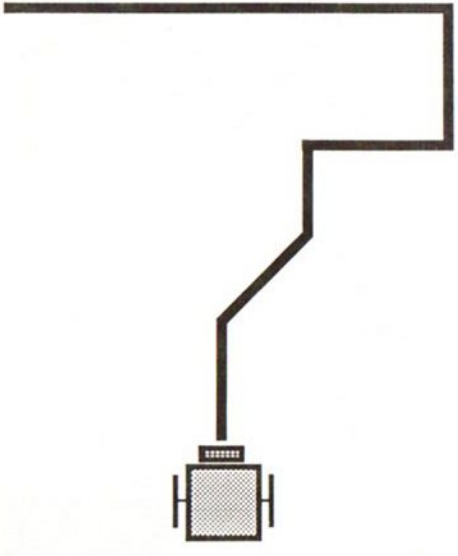


Bild 10.22: Fahrspur mit mehreren Abbiegungen.

```
230 D%=180 : CALL ITL(D%,TS)
```

und testet, ob es dort weitergeht:

```
240 CALL IEX(EX)
250 IF EX>H+20 THEN GOTO 90
```

Wenn sie keinen Weg findet, ist die Bahn zu Ende.

```
260 PRINT "Bahnende!"
270 END
```

Geben Sie die Zeilen ein, setzen die Schildkröte wieder an den Bahnanfang und starten mit RUN. Sie wird die Abbiegung erkennen, egal ob nach links oder rechts. Wenn Sie eine größere Strecke zusammenkleben, achten Sie darauf, daß die Teilstücke zwischen den Abbiegungen mindestens die Länge der Schildkröte haben. Sonst weiß sie nach einer 90°-Drehung nicht weiter.
Natürlich lassen sich auch Abbiegungen mit einem kleineren Winkel erkennen. Die Schildkröte muß dabei nicht nur nach der 90°-Drehung sondern auch bei den Drehschritten dazwischen prüfen, ob sie sich wieder auf der Spur befindet. Ändern Sie das Programm ab:

```
210 W=0
220 D%=5 : CALL ITL(D%,TS)
230 W=W+5
240 CALL IEX(EX)
250 IF EX>H-20 THEN GOTO 90
260 IF W<180 THEN GOTO 220
270 PRINT "Bahnende!"
280 END
```

und erstellen eine Bahn nach Bild 10.22 (oder ähnlich).
Nach dem Start muß die Schildkröte die Spur von Anfang bis Ende durchfahren und jede Abbiegung erkennen. Achten Sie darauf, daß immer ein gleichmäßig helles Licht herrscht. Schon ein Schatten kann die Messung beeinflussen, so daß die Schildkröte unkontrolliert vom Weg abkommt.
Wegführungen dieser Art werden Sie z.B. bei Flurförderfahrzeugen sehr oft finden. Meist werden sie jedoch wegen der höheren Störsicherheit durch Leitungen im Boden hergestellt. Der Roboter wird dann nicht durch Licht, sondern induktiv über ein Kabel gesteuert.
Programme, die Fahrzeugsteuerungen nach Bildaufzeichnungssystemen ermöglichen, werden auch dem Begriff "Künstliche Intelligenz" zugeordnet. Dieser Begriff verursacht derzeit Aufregung.

Bei Volkswagen hat es auch schon Versuche gegeben, ein Fahrzeug mit einer Kamera automatisch fahren zu lassen. Das Bildverarbeitungssystem orientierte sich nur an den Randstreifen und den Mittelstreifen einer Straße. Die Versuche sind tatsächlich gelungen. Das Auto war sogar 120 km/h schnell - unfallfrei. Und Hersteller von Roboterfahrzeugen bieten jetzt schon Seriengeräte für innerbetriebliche Transporte an, das sich ebenfalls an Fahrbahnmarkierungen und Wegbegrenzungen orientieren, wenn sie auch nicht ganz so schnell fahren.

Kann der Computer die Rolle des Menschen einnehmen, indem er intelligent reagiert? Sicherlich nicht, denn zum menschlichen Denken gehört viel mehr als nur Intelligenz - Phantasie, Einfühlungsvermögen, Kreativität ... Der Begriff "Künstliche Intelligenz" und seine Bedeutung ist selbst unter Fachleuten umstritten. Wir wollen hier eine ganz praktische Beschreibung geben: Programme nach Methoden der künstlichen Intelligenz können schneller zu Problemlösungen kommen als durch systematisches Ausprobieren, weil sie in ihrem Arbeitsspeicher frühere Erfahrungen abspeichern und diese bei der Problemlösung mitverwenden.

Unserem Bahnverfolgungsprogramm wollen wir jetzt auch einen Hauch von künstlicher Intelligenz verleihen. Bauen Sie zunächst eine Bahn in Form einer Acht auf (Bild 10.23).

Die Schildkröte sollte mit dem vorliegenden Programm sauber auf der Bahn entlanglaufen. Die Kreuzung sollte sie nicht spüren, wenn diese einigermaßen rechtwinklig ist und die Schildkröte im Kreuzungsbereich ausreichend lange gerade Strecken vorfindet. Durch Kurskorrekturen sauber auf die Gerade gebracht, sollte sie diese Strecken in maximaler Geschwindigkeit durchlaufen.

Anders in der Rechtskurve, hier sind immer wieder Kurskorrekturschritte notwendig, die die Geschwindigkeit herabsetzen. Noch schlimmer in der Linkskurve. Da die Schildkröte immer zuerst nach rechts sucht, dauert diese Kurve noch viel länger. Ein Umstellen des Programms löst das Problem auch nicht; da würden die gleichen Schwierigkeiten in der Rechtskurve auftauchen.

Hier kommt unsere künstliche Intelligenz ins Spiel. Wir führen eine Gedächtnisvariable RI ein. War die letzte Kurskorrektur nach rechts, so wird RI=0 gesetzt, bei links RI=1. Wird wieder eine Kurskorrektur notwendig, soll das Programm bei RI=0 zuerst rechts, bei RI=1 zuerst links probieren.

Löschen Sie zunächst die Zeilen 140 bis 180 und geben Sie folgende Zeilen neu ein:

```
60 RI=0
120 IF RI=0 THEN GOTO 300
130 IF RI=1 THEN GOTO 400
300 D%=5 : CALL ITR(D%,TS)
310 CALL IEX(EX)
320 IF EX>H-20 THEN GOTO 90
330 D%=10 : CALL ITL(D%,TS)
340 CALL IEX(EX)
350 IF EX>H-20 THEN RI=1 : GOTO 90
360 D%=5 : CALL ITR(D%,TS)
```

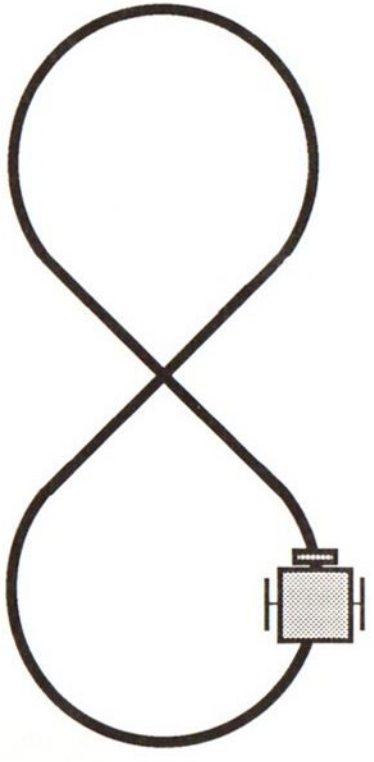


Bild 10.23: Achterschleife für die Schildkröte.

```
370 GOTO 190
400 D%=5 : CALL ITL(D%,TS)
410 CALL IEX(EX)
420 IF EX>H-20 THEN GOTO 90
430 D%=10 : CALL ITR(D%,TS)
440 CALL IEX(EX)
450 IF EX>H-20 THEN RI=0 : GOTO 90
460 D%=5 : CALL ITL(D%,TS)
470 GOTO 190
```

Setzen Sie die Schildkröte mit dem verbesserten Programm auf die Achterschleife. Sie werden feststellen, daß zwar zu Kurvenbeginn die Kurskorrektur noch mit der falschen Seite beginnt, danach hat aber das Programm die neue Situation gelernt und wird die Schildkröte die Kurve zügig durchfahren lassen.
Selbstverständlich können Sie in das Programm noch andere Erfahrungsregeln einbauen. Um beim Übergang von einer Kurve in die andere nicht nach der falschen Seite zu suchen, bauen Sie etwa folgende Vorschrift ein:
Wurden eine Anzahl von Schritten ohne Kurskorrektur durchgeführt, so wird RI gerade umgedreht, also

```
RI=1-RI
```

Mit dieser Strategie werden die Acht und Wellenlinien noch besser durchlaufen.
Auf Diskette finden Sie wieder ein fertiges Programm, mit dem Sie ausführlich die optische Steuerung der Schildkröte nach einer Fahrbahnlinie üben können. Es heißt BAHN.BAS und beinhaltet eine noch verbesserte Helligkeitsjustierung.

10.7 Verkehrsleitsysteme: Auf dem richtigen Weg

Die Schildkröte wurde im letzten Versuch mit einem Lesekopf ausgestattet, mit dem sie einer Spur auf der Fahrbahn folgen konnte. Der Fotowiderstand als Sensor nahm dabei das von der Fahrbahn reflektierte Lampenlicht auf und konnte so erkennen, ob sich die Schildkröte auf der Spur (dunkel) oder daneben (hell) befand. Je nach Meßwert wurde der Weg der Schildkröte korrigiert.

Neben der Steuerung der Schildkröte wollen wir den Lesekopf nun auch zur Aufnahme von Informationen von der Fahrbahn benutzen. Jeder kennt das Prinzip vom Einkauf: Lebensmittel und andere Waren werden in Kaufhäusern mit Etiketten versehen, auf denen sich eine Anzahl von Strichen nebeneinander befinden. An der Kasse wird mit einem Lesestift dieses Etikett abgetastet, worauf Preis, Bezeichnung der Ware usw. angezeigt werden. Die Informationen sind also in diesen Linien - dem Strichcode oder Produktinformationscode - enthalten.

Für den folgenden Versuch benötigen wir das Schildkrötenmodell mit Lesekopf wie zuvor. Der Lesekopf wird so eingestellt, daß er etwa 3...5 mm Abstand von der Fahrbahn hat. Wenn die Schildkröte mit dem Interface verbunden und der Interfacetreiber und BASIC geladen sind, setzen wir sie auf eine weiße Unterlage (Karton). Die Fahrbahn muß hell sein, damit genügend Licht zum Fotowiderstand reflektiert wird.

Ob Lichtstärke, Fahrbahnhelligkeit und Sensorabstand in Ordnung sind, prüfen wir mit folgendem Programm:

```
LOAD"INIT"
10 CALL IIN
15 CALL ITI
20 FOR I=0 TO 200
30 CALL I3R
40 NEXT I
50 CALL IEX(EX)
60 PRINT EX
```

Auf dem Bildschirm erscheint jetzt eine Zahl, die der Helligkeit der Fahrbahn entspricht. Kleben Sie mit schwarzem PVC-Isolierband ein Stück von ca. 15 mm Länge mitten auf die Fahrbahn und setzen die Schildkröte so hin, daß der Lesekopf auf den Streifen zeigt. Geben Sie die beiden Befehle noch einmal ein; jetzt ist die Zahl am Bildschirm wesentlich höher, da die Bahn dunkler ist als der Untergrund.

Ohne daß Sie es merkten, haben Sie bereits eine Information von der Fahrbahn

gelesen: den Zustand "hell" bzw. "dunkel", der als unterschiedlicher Zahlenwert angezeigt wird. Man nennt dies auch eine binäre Information, die nur aus zwei Zuständen besteht. Den beiden Zuständen weisen wir jetzt Zahlen zu: hell = 1, dunkel = 0. Mit diesen Bezeichnungen - auch logische Werte genannt - läßt sich besser weiterarbeiten. Zum Experimentieren geben Sie folgendes Programm ein:

```
70 HE=EX
80 CALL IEX(EX)
90 Z$="binär 1"
100 IF EX>HE+20 THEN Z$="binär 0"
110 PRINT Z$
120 A$=INKEY$
130 CALL I3R
140 IF A$="" THEN GOTO 120
150 GOTO 80
```

Setzen Sie die Schildkröte mit dem Lesekopf auf eine helle Stelle und geben Sie RUN ein. "Hell" wird als "binär 1" erkannt; der entsprechende Meßwert wird in der Variablen HE festgehalten (Zeile 70). Mit Zeile 110 erscheint die Information auf dem Bildschirm. Jetzt wartet das Programm auf einen Tastendruck (Zeilen 120 bis 140) und springt nach Zeile 80. Hier wird erneut gemessen und dieser Wert (EX) dann in Zeile 100 mit dem ersten (HE) verglichen. Liegt er mindestens 20 höher (HE+20) als dieser, befindet sich der Lesekopf auf einer dunklen Fläche. Das entspricht dem Zustand "binär 0". Setzen Sie die Schildkröte auf verschiedene Stellen und messen durch Tastendruck den jeweiligen Zustand bzw. den logischen Wert des Meßpunktes.

Wie das Etikett mit dem Strichcode auf der Dosenmilch soll auch unsere Information aus mehreren Stellen hintereinander bestehen. Man nennt jede Stelle ein "Bit". Diese Bits liest die Schildkröte dann nacheinander ein, wenn sie z.B. von links nach rechts darüber fährt.

Bevor wir aber die Informationsbits auf die Fahrbahn kleben, muß noch etwas zu der Strichbreite gesagt werden. Unsere Schildkröte kann immer nur eins tun: fahren oder messen. Ein Fahrschritt beträgt 5 mm, wie wir in Kapitel 9 gesehen haben; d.h. die einzelnen Bitpunkte in Form von hellen und dunklen Flächen müssen mindestens 5 mm auseinander liegen. Da der Startpunkt beim Messen ebenfalls um mindestens ±2,5 mm differieren kann, müßten wir eine "Strichbreite" von 10 mm wählen, damit der Fotowiderstand mit Sicherheit auf den Strich zeigt.

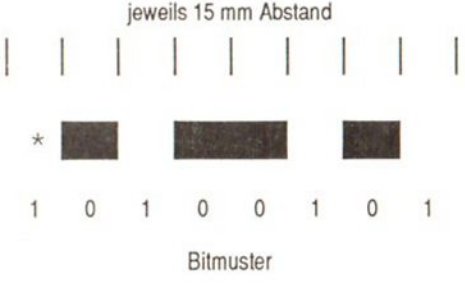


Bild 10.24: Informationscode auf der Fahrspur.

Zuverlässig in die Mitte des jeweiligen Striches trifft die Schildkröte aber erst bei einer Breite von drei Schildkrötenschritten, also 15 mm. Schneiden Sie also von dem schwarzen Isolierband 15 mm lange Streifen ab, um sie später wie in Bild 10.24 auf die Fahrbahn zu kleben.
Wir verändern unser Programm ein wenig. Der Ausdruck wird verkürzt, und die Tastenabfrage am Ende wird ersetzt. Geben Sie ein:

```
80 PRINT "Bitmuster :";
90 CALL IEX(EX)
100 Z$=" 1"
110 IF EX>HE+20 THEN Z$=" 0"
120 PRINT Z$;
130 S%=3 : CALL ITV(S%,TS,E5)
140 GOTO 90
```

Zeile 150 wird gelöscht. Die Schildkröte steht zunächst mit dem Lesekopf auf der Startposition (in Bild 10.24 durch * gekennzeichnet). Wenn Sie das Programm mit RUN starten, wird der Binärwert 1 angezeigt: hell. Anschließend fährt die Schildkröte drei Schritte (15 mm) vor und bleibt auf Bit 1 stehen. Hier wird binär 0 angezeigt, da dieses Feld dunkel ist. Danach geht es weiter. Wenn der letzte Streifen passiert wurde, muß auf dem Bildschirm stehen:

Bitmuster 1 0 1 0 0 1 0 .....

Stoppen Sie die Schildkröte mit der Ctrl-Break-Taste, da sie sonst immer weiter Bits sucht und ausdruckt.
Kleben Sie die Isolierbandstücke in einer anderen Reihenfolge auf und lesen die Information ein. Stimmt sie immer? Wenn nicht, ist die Fahrbahn vielleicht unterschiedlich beleuchtet oder der Lesekopf zu weit davon entfernt.
Nicht befriedigend ist, daß die Schildkröte am Anfang auf der Startposition stehen muß. Sie soll die Information auch bei voller Fahrt lesen können - so wie der Lesestift an der Kasse. Dazu verwenden wir die ersten zwei Streifen, der dunkle gefolgt von einem hellen Streifen als Kennung vor dem Informationsbereich.
Dies wird uns auch erlauben, die Leseempfindlichkeit des Lesekopfes vor jedem Informationsblock, ähnlich einer Aussteuerautomatik beim Kassettenrekorder, für jeden Informationsblock individuell einzustellen.
Erweitern Sie das Programm:

```
80 REM suche Startbit-Kombination
```

Eine solche Kennung wird als Startbit bezeichnet. In der Datenübertragung über Telefon oder andere Leitungen wird auch vor jedem Informationsblock ein Startbit gesendet. Dort genügt eines; wir haben aus Gründen der Betriebssicherheit zwei Startbits.
Nach der Informationsübertragung muß eine Weile nichts kommen, bevor es mit der nächsten Übertragung weiter geht. Diese Weile "nichts" nennt man Stoppbit. Wie man sich leicht überzeugen kann, muß das Stoppbit mindestens solange wie ein Bit sein. In der Datenübertragung werden Stoppbits mit einfacher, anderthalbfacher und doppelter Bitbreite verwendet.

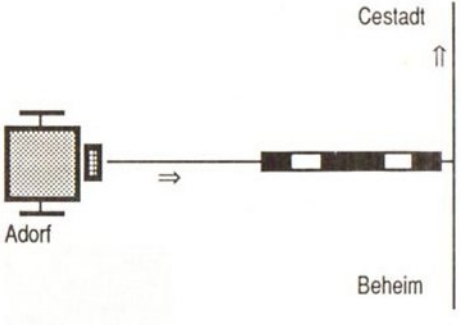


Bild 10.25: Autobahnnetz mit Verkehrsleitsystem.

```
90 MI=HE+15
100 S%=1 : CALL ITV(S%,TS,E5)
110 CALL IEX(EX)
120 IF EX<MI THEN GOTO 100
130 S%=1 : CALL ITV(S%,TS,E5)
140 CALL IEX(EX)
150 HA=EX
160 MI=(HA+HE)/2
170 S%=1 : CALL ITV(S%,TS,E5)
180 CALL IEX(EX)
190 IF EX>=MI THEN GOTO 170
200 REM Startbit identifiziert
210 HE=EX
220 S%=4 : CALL ITV(S%,TS,E5)
230 C$=""
240 FOR I=0 TO 3
250 MI=(HA+HE)/2
260 CALL IEX(EX)
270 IF EX<MI THEN C$=C$+" 1"
      : HE=EX : GOTO 300
280 C$=C$+" 0"
290 HA=EX
300 S%=3 : CALL ITV(S%,TS,E5)
310 NEXT I
320 PRINT "Code: ";C$
```

Das Einlesen der Datenbits erfolgt jetzt automatisch in einer FOR...NEXT-Schleife (Zeile 240 bis 310). Jedes der vier Bits wird mit dem Startbit, den Flächen vor Bit 0 verglichen. Setzen Sie die Schildkröte an den Anfang der Bahn, und starten Sie das Programm mit RUN. Am Ende muß auf dem Bildschirm stehen:

Code: 0 0 1 0

Versuchen Sie auch hier wieder verschiedene Informationscodes. Die Betriebssicherheit sollte durch die zwei Startbits und die Schwellwertautomatik wesentlich verbessert sein.
Was kann man nun mit den eingelesenen Informationen anfangen? Unsere Schildkröte ist ein Fahrroboter, und sie soll deshalb bei ihrer Fahrt mit wichtigen Mitteilungen versorgt werden.
Für die Schildkröte wollen wir ein Verkehrsleitsystem einrichten und dafür unsere Datenübertragung von der Fahrbahn zum Lesekopf benutzen. Zunächst bauen wir ein kleines Autobahnnetz nach Bild 10.25 auf.
Die Schildkröte steht in Adorf und will nach Cestadt. Den Weg dorthin kennt sie nicht. Der Verkehrscomputer kennt die Straßenkarte und die Verkehrslage und teilt der Schildkröte vor der nächsten Abzweigung mit, in welche Richtung sie fahren muß. In

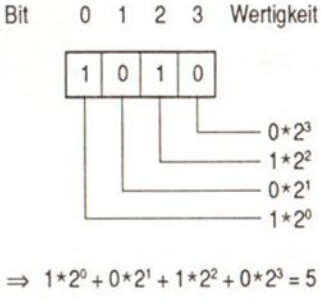


Bild 10.26: Bits im Informationscode.

Solche Systeme nennt man Verkehrslenk- oder -leitsysteme. Darunter versteht man ein Computernetz (z.B. ALI), mit dem man Informationen vom Verkehrsteilnehmer (Startpunkt, Fahrziel usw.) sowie der Fahrstrecke (Verkehrsdichte, Stauungen, Baustellen usw.) sammelt und daraus für den Autofahrer die beste Fahrstrecke ermittelt. Die Verbindung zwischen dem Verkehrscomputer und dem Bordcomputer im Auto wird dabei durch Induktionsschleifen in der Straße und Sensoren im Wagen hergestellt.

Wirklichkeit würden die Daten zur Induktionsschleife in der Straße geschickt - hier übertragen wir sie optisch mit dem Informationsstreifen.
Zunächst wollen wir den Bits im Informationscode eine Bedeutung geben. Bild 10.26 zeigt dies. Mit vier Bits kann man bis 15 zählen. Jede Bitstelle hat eine Wertigkeit. Ist das Bit gesetzt, zählt dieser Wert zur Gesamtsumme. Sind alle Bits gesetzt, erhält man die Zahl 15; ist z.B. nur Bit 2 gesetzt, ergibt das die Zahl 4. Von diesen 16 möglichen Codes belegen wir nur vier:

Bitmuster	Code	Bedeutung
1 0 0 0 0 0	0	geradeaus weiter
1 0 0 0 0 1	1	rechts abbiegen
1 0 0 0 1 0	2	links abbiegen
1 0 0 1 0 0	4	Ende der Fahrt

Wir ergänzen das Programm mit den Fallunterscheidungen für obige vier Codes. Geben Sie diese Zeilen ebenfalls ein und fügen den Informationsstreifen nach Bild 10.25 in die Strecke ein.

```
320 IF C$=" 0 0 0 0" THEN GOTO 100
330 IF C$<>" 0 0 0 1" THEN GOTO 370
340 S%=13 : CALL ITV(S%,TS,E5)
350 D%=90 : CALL ITR(D%,TS)
360 GOTO 100
370 IF C$<>" 0 0 1 0" THEN GOTO 410
380 S%=13 : CALL ITV(S%,TS,E5)
390 D%=90 : CALL ITL(D%,TS)
400 GOTO 100
410 IF C$=" 0 1 0 0" THEN PRINT
        "Programmende" :END
420 PRINT "Ungültigen Code
        gefunden."
430 GOTO 100
```

Beim Abbiegen lassen wir die Schildkröte noch ein Stück vorgehen, damit sie hinter dem Codestreifen dreht. Nach der Drehung erfolgt ein Rücksprung an den Programmanfang, so daß sie den nächsten Codestreifen sucht. Bei "geradeaus weiter" wird gleich zurückgesprungen. Der Code für Bahnende führt nach entsprechender Meldung zum Programmende. Alle anderen Codes werden als ungültige Codes am Bildschirm gemeldet, und die Schildkröte setzt ihren Weg geradeaus fort.
Jetzt setzen Sie die Schildkröte an den Streckenanfang "Adorf" und starten das Programm mit RUN. Sie wird, wenn keine Datenübertragungsfehler auftreten, in Cestadt ankommen. Denken Sie wieder an

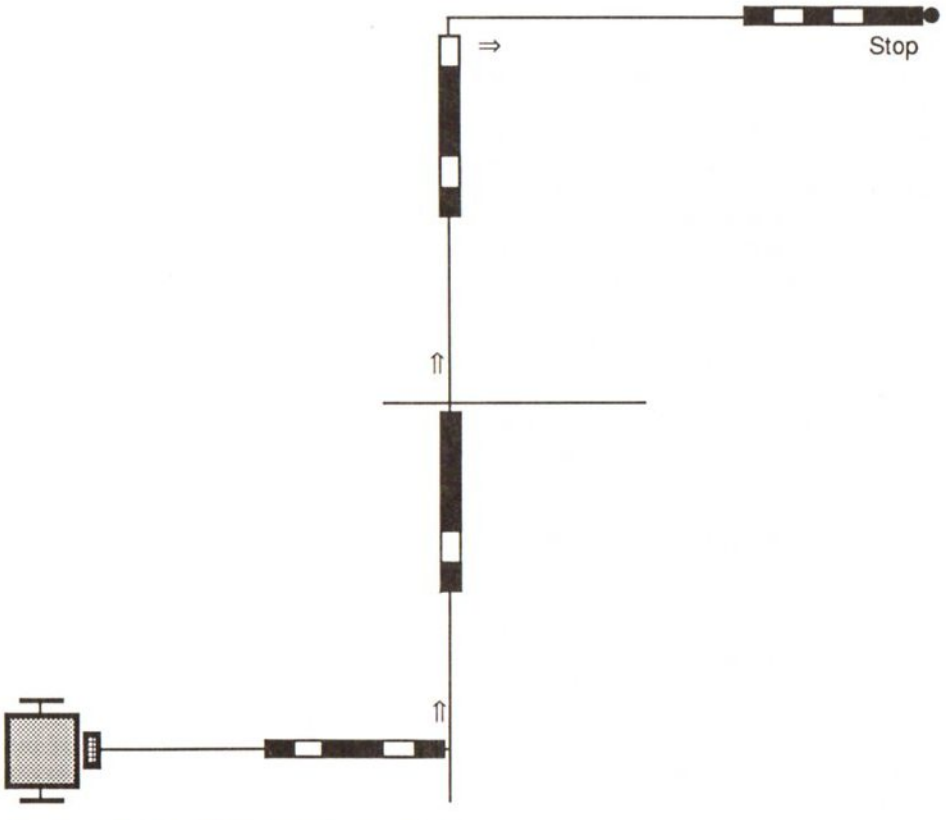


Bild 10.27: Die Schildkröte sollte auch eine längere Strecke meistern.

eine gleichmäßige Beleuchtung und den richtigen Abstand des Lesekopfes von der Fahrbahn!

Es steht Ihnen frei, den noch nicht belegten Codes Bedeutungen zuzuweisen. Schreiben Sie eigene Programmstücke. Hier ein paar Anregungen: Erlauben Sie auch andere Abzweigwinkel, z.B. um 45° nach rechts und links. Flankieren Sie die Informationscodes rechts und links mit Codes, die dem Programm signalisieren, daß die Schildkröte von der Bahn abgekommen ist; wird solch ein Code gefunden, kann eine entsprechende Kurskorrektur durchgeführt werden. Sie können auch Ortsschilder codieren:

Bitmuster	Code	Ortsname
1 0 1 0 0 0	8	Adorf
1 0 1 0 0 1	9	Beheim
1 0 1 0 1 0	10	Cestadt
1 0 1 0 1 1	11	Dehausen usw.

Ein ausführliches Programm zum Thema "Informationcodes" finden Sie auf der Diskette unter dem Namen CODE.BAS. Es zeigt Ihnen die Handhabung der Schildkröte beim Lesen und Verarbeiten von Fahrbahndaten, u.a. in Verkehrsleitsystemen.

Jetzt haben Sie unser ganzes Anleitungsbuch durchgearbeitet und eine Vielzahl von spannenden Versuchen gemacht. Ihr fischertechnik COMPUTING EXPERIMENTAL ist aber nun keineswegs wertlos - im Gegenteil: Sie wissen jetzt schon soviel über Computing und Robotik, daß Sie ganz alleine Experimente durchführen können. Und Sie werden sehen, mit Erfolg. Ein paar Anregungen geben wir Ihnen gerne mit auf den Weg.

Zunächst einmal können Sie mit dem Taster einen Morsetrainer aufbauen; wenn der dann funktioniert, erweitern Sie ihn mit der Lampe und dem Fotowiderstand zu einer optische Datenübertragung, bei der dann mit dem Taster Morsezeichen gesendet werden.

Bauen Sie mal mit der Lampe und dem Fotowiderstand eine Waage auf. Über eine drehbare Kulissenscheibe kann der Fotowiderstand je nach Gewicht unterschiedlich stark abgedeckt werden.

Mit derselben Einrichtung kann man auch eine Regelung aufbauen, bei der eine Platte bei Bewegung immer in der Waagerechten gehalten wird.

Propeller und Lichtschranke sind die Grundelemente für einen Windmesser. Probieren Sie es einmal, ein solches Gerät zu entwickeln. Der Propeller dreht sich frei und wird vom Wind angetrieben. Auf seiner Achse befindet sich ein Flügel, der die Lichtschranke bei Drehung unterbricht. Die Impulse werden gezählt, und daraus soll der Computer die Windgeschwindigkeit errechnen.

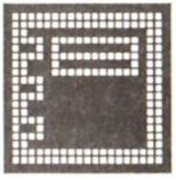
Ebenso lassen sich natürlich auch die vorhandenen Modelle erweitern: So können Sie bei dem Computerauge eine Überkopfbewegung einbauen. Damit läßt sich dann z.B. eine Nachführung für Sonnenkollektoren realisieren.

Dem Roboter könnte man eine dritte Achse spendieren, die den Arm auch nach oben und unten bewegt. Außerdem läßt sich hier ein Greifer durch einen Elektromagneten herstellen.

Zur besseren Hinderniserkennung kann man die Schildkröte mit zusätzlichen seitlichen Fühlern ausrüsten. Wenn man für den optischen Sensor zwei Fotowiderstände nebeneinander benutzt, die an die Eingänge EX und EY angeschlossen werden, lassen sich Richtungsabweichungen durch Differenzmessung besser und schneller erkennen.

Auch die Schildkröte kann mit einem Transportarm ausgerüstet werden. Mit einem Schreibstift versehen, könnte sie sogar

Zeichnungen erstellen. Das Auf und Ab des Schreibstiftes könnte man mit einem Elektromagneten steuern. So entspricht die Schildkröte dann ganz und gar ihrem Bildschirm-Pendant.
Sie sehen, Ihr fischertechnik COMPUTING EXPERIMENTAL bietet Ihnen fast unbegrenzte Möglichkeiten für viele weitere hochinteressante Experimente.



Dieses Kapitel soll Ihnen in erster Linie helfen, die Programme aus den vorangegangen Kapiteln oder von der Diskette nach Ihren eigenen Wünschen zu verändern und auszugestalten. Dazu werden wir eine Reihe von Hinweisen und technischen Detailinformationen geben. Wir erheben damit keinen Anspruch auf Vollständigkeit; Sie werden sicherlich noch eine Menge mehr nützlicher Kniffe entdecken, wenn Sie sich mit Ihrem fischertechnik COMPUTING EXPERIMENTAL intensiv mit Themen wie Messen, Steuern, Regeln, Robotik, Bildschirmgrafik und Bildverarbeitung befassen.

Andererseits soll dieses Kapitel auch nicht so verstanden werden, daß Sie all diese technischen Details beherrschen müßten, bevor Sie mit dem fischertechnik COMPUTING EXPERIMENTAL etwas anfangen können. Dies ist keineswegs so; was in den vorangegangenen Kapiteln dargestellt wurde reicht vollkommen zur Durchführung der Experimente aus.

Aber vielleicht wollten Sie schon immer wissen wie und warum ...

Anhang 1 Technische Informationen

Wichtiger Hinweis!

Die Programme auf der Diskette und in dem vorliegenden Experimentierhandbuch des fischertechnik COMPUTING EXPERIMENTAL wurden auf einem IBM-AT entwickelt und gründlich auf mehreren verschiedenen IBM- und kompatiblen PC getestet.

Es verbleiben in der Software ungelöste Probleme mit dem CALL-Kommando im Direktmodus, wenn GW-BASIC verwendet wird. Die Probleme haben ihre Ursache mutmaßlich in dem GW-BASIC selbst.

Jeder Schreibfehler im CALL-Kommando, eine falsche Zahl von Argumenten oder ein Aufruf nicht-existenter Kommandos kann zu einem Absturz des BASIC oder einem Rücksprung auf MS-DOS-Ebene führen. Auch sollten bei CALL-Kommandos mit Parametern (wie die Schildkrötenkommandos) sämtliche Parameter in der gleichen Zeile definiert werden, wenn das CALL-Kommando im Direktmodus benutzt wird.

Wir empfehlen Ihnen, den Direktmodus aus Sicherheitsgründen gänzlich zu vermeiden und nur den Programmiermodus zu benutzen, ganz so wie die Beispiele im Experimentierhandbuch angelegt sind.

A 1.1 Der Interfacetreiber

Die Interface-Kommandos sind Unterprogrammaufrufe von Programmstücken, die in Maschinensprache abgefaßt sind. Die Sammlung all dieser Unterprogramme wird in den Arbeitsspeicher des Computers geladen, wenn die Datei FISCHER.COM unter MS-DOS aufgerufen wird. Der Installationsteil von FISCHER.COM macht die Unterprogramme resident, d.h. sie bleiben in dem Arbeitsspeicher, bis der Computer abgeschaltet oder neu angefahren wird.

Nachdem BASIC geladen ist, stellen die Anweisungen des Programms INIT.BAS die Verbindung zu den residenten Unterprogrammen her. Zunächst ermittelt das Programm die Anfangsadresse, wobei es auch feststellen kann, ob der Interfacetreiber überhaupt geladen wurde. Danach wird die Einsprungadresse der Kommandos in einem Satz von Ganzzahl-Variablen (=Kommandonamen) gespeichert. Beachten Sie: jegliche Veränderung des Inhalts dieser Variablen wird mit Sicherheit zu einem Absturz des Computers führen. Prüfen Sie deshalb immer wieder, daß Sie keine dieser Variablen, eventuell versehentlich, zu anderen Zwecken verwenden.

Jeglicher Datenaustausch mit den Unterprogrammen erfolgt ebenfalls mit Ganzzahl-Variablen. Ausnahme: CALL IGLOAD

und CALL IGSAVE verwenden Textvariablen zur Speicherung des Dateinamens. Entsprechend der Wahl der Kommando- und Parameternamen legt das INIT-Programm alle Variablen, die mit E, G, I oder T anfangen als Ganzzahl-Variablen fest. Andere Variablen müssen, wenn sie als Parameter von Unterprogrammaufrufen dienen sollen, ausdrücklich als ganzzahlig definiert werden, indem ein Prozentzeichen an den Variablennamen angehängt wird, z.B. S%.

Der Standard-Bildschirm des IBM-PC umfaßt 25 Zeilen mit je 80 Zeichen. Es gibt noch einen Bildschirmmodus mit 40 Zeichen je Zeile, der von den Beispielprogrammen auf der Diskette aufgerufen wird, denn er bietet bessere Lesbarkeit bei Vorführungen.

In den 40-Zeichen-Modus wird mit der BASIC-Anweisung WIDTH 40 umgeschaltet. Diese Anweisung führt jedoch zu einer Fehlermeldung, wenn sie im Zusammenhang mit einem Monochromschirm aufgerufen wird. Deshalb wurde das spezielle Kommando CALL IW40 geschaffen. Auf CGA- und kompatiblen Bildschirmadaptern wirkt es wie die Anweisung WIDTH 40. Auf Monochrom-Adaptern begrenzt es die Bildschirmausgabe auf die 40 Spalten der linken Bildhälfte. Mit dem Kommando CALL IW80 wird wieder zurückgeschaltet.

Wenn die Schildkrötengrafik eingeschaltet wurde, verbleiben noch vier Zeilen für Textdarstellung. Texte in diesen Zeilen erscheinen immer im echten 40-Zeichen-Modus, gleich welcher Bildschirmadapter benutzt wird, denn sie werden im Grafikmodus erzeugt.

Die verbleibenden Zeilen sind die Zeilen 22 bis 25. Dies muß bei Verwendung der LOCATE-Anweisung beachtet werden.

Im Text wird immer von der "Ctrl-Break"-Taste gesprochen, wenn es darum geht, ein BASIC-Programm zu stoppen. Die Taste "Ctrl" kann auf Ihrem Computer als "Steuerung" oder "Strg" bezeichnet sein; die Taste "Break" kann als "Scroll Lock" oder "Bildlauf anhalten" beschriftet sein. Einige Computer verwenden auch die Tastenkombination "Ctrl" und Buchstabe "C" zum Anhalten des BASIC-Programms.

Wenn Sie ein Schrittkommando oder Schildkrötenkommando unterbrechen wollen, müssen Sie zuerst die Ctrl-Break-Taste und dann die Taste F10 drücken. Die Erklärung dafür ist ein wenig kompliziert: Der Tastendruck F10 beendet zwar das Schritt- bzw. Schildkrötenkommando, aber nicht den Programmlauf. Wenn nun als nächstes Kommando wieder ein Schritt- oder Schildkrötenkommando folgt (weil es z.B. in einer Schleife wiederholt wird), so wird der Anschein erweckt, daß der Tastendruck nichts bewirke.

Wenn Sie dagegen vor der F10-Taste Ctrl-Break gedrückt haben, so ist ein Stop-Merkmal in dem BASIC-Interpreter gesetzt. Nach Beendigung des Schritt- bzw. Schildkrötenkommandos (durch F10) wird das Stop-Merkmal aktiv und das Programm endet.

Bei der Verwendung der Cursortasten wird davon ausgegangen, daß **nicht** die Taste Num-Lock gedrückt ist, da sonst nicht die korrekten Zeichencodes zur Cursorabfrage erscheinen. Einige Computer verwenden selbst bei korrekter Stellung des Num-Lock-Schalters nochmals andere Zeichencodes. Schlagen Sie in diesem Fall in dem Handbuch Ihres Computers nach und ändern Sie die Programme entsprechend.

A 1.3 Hochauflösende Bildschirmgrafik

Wenn ein PC mit einer Farbgrafikkarte nach dem CGA-Standard ausgestattet ist, bietet er eine Bildfläche, die aus 640 x 200 Bildpunkten besteht. Im Farbmodus werden je zwei benachbarte Bildpunkte zusammengefaßt; dafür kann man aber jedem dieser zusammengesetzten Bildpunkte eine von vier Farben zuweisen. Die Bildschirmauflösung geht zurück; es stehen nur noch 320 x 200 Bildpunkte zur Verfügung. Die Umschaltung auf den Grafikmodus erfolgt durch die BASIC-Anweisung SCREEN 1. Das Kommando CALL IGE kann erst nach dieser Umschaltung benutzt werden und installiert die Schildkrötengrafik.

Das Zurückschalten in den Textmodus mit 80 x 25 Zeichen erfolgt in drei Schritten: Zuerst wird mit dem Kommando CALL IGA die Schildkrötengrafik beendet, dann wird mit der BASIC-Anweisung SCREEN 0 wieder in den Textmodus geschaltet. Zum Schluß muß, wenn gewünscht, mit dem Kommando CALL IW80 wieder in die 80-Zeichen-Darstellung umgeschaltet werden.

Die Farben des Grafikmodus können unter BASIC aus zwei Paletten ausgewählt werden. Die Palettensteuerung erfolgt nach der Anweisung SCREEN 1 durch die COLOR-Anweisung (nicht zu verwechseln mit der gleichnamigen Anweisung im Textmodus!):

COLOR <Zeichenfarbe>,<Palette>

Durch den Tastendruck Shift-PrtScr können Sie nicht nur den Textschirm sondern auch den Grafikschirm auf dem Drucker ausgeben. Damit letzteres auch funktioniert, müssen Sie vor Starten von BASIC das DOS-Systemprogramm GRAPHICS aufrufen.

Die Grafikausgabe, wie sie in Abschnitt A 1.3 beschrieben wurde, erfordert normalerweise das Vorhandensein eines CGA-Bildschirmadapters (color graphics adapter) oder eines aufwärts kompatiblen Adapters wie EGA (enhanced graphics adapter) oder VGA (video graphics adapter). Andererseits sind die monochromen grafischen Bildschirmadapter von Hercules oder die dazu kompatiblen Adapter weit verbreitet. Sie bieten eine Auflösung von 720 x 348 Bildpunkten ohne Farbdarstellung. Die fischertechnik COMPUTING EXPERIMENTAL Software enthält im Interfacetreiber eine automatisch arbeitende Emulation des CGA-Adapters auf einem Monochromadapter.

Wenn der Interfacetreiber in dem PC einen Monochromadapter feststellt, installiert er ein Stück Maschinenprogramm, das BASIC vorgaukelt, es wäre eine CGA-Karte installiert. Deshalb können Anweisungen wie SCREEN 1 und die damit verknüpften Zeichenkommandos, die normalerweise zu einer Fehlermeldung führen würden, ohne Einschränkung benutzt werden.

Die CGA-Ausgabe wird dabei folgendermaßen in der Bildfläche des Monochrom-Bildschirms untergebracht:

Die 640 Bildpunkte in waagrechter Richtung passen ganz gut in die 720 Bildpunkte des Monochromadapters; hier wird keine Änderung vorgenommen. Von den 200 Bildschirmzeilen wird jede zweite verdoppelt, so daß 300 Bildschirmzeilen entstehen und die Seitenverhältnisse in etwa gewahrt bleiben. Diese Verdopplung geschieht durch ein selbsttätig im Hintergrund laufendes Programm unabhängig von den Ausgabeaktivitäten in festen Zeitabständen. Schneller Bildlauf kann deswegen zu einem Nachleuchten führen.

Bei der Steuerung schneller Prozesse mit dem Interface kann das Verdoppeln zu unerwünschten Verzögerungen führen, insbesondere bei der Steuerung der Schildkröte. Daher stehen zwei Kommandos zum Ein- und Ausschalten der Zeilenverdopplung zur Verfügung:

CALL IHA schaltet die Verdopplung aus und

CALL IHE schaltet sie wieder ein.

Wir empfehlen folgende Vorgehensweise: Belassen Sie das Programm zunächst im Verdopplungsmodus, der standardmäßig eingeschaltet ist, für alle vorbereitenden Arbeiten, insbesondere beim Laden von Hintergrundbildern. Benutzen Sie CALL IHA direkt vor dem Start der Schildkröte. Wenn die Schildkröte ihre Aufgabe gemei-

stert hat, schalten Sie die Verdopplung wieder mit CALL IHE ein, um die Ergebnisse in optimaler Bilddarstellung zu präsentieren.
Auch wenn "Farben" keinen Sinn im Zusammenhang mit Monochromadaptern ergeben, so liefert die Farbsteuerung dennoch unterschiedlich Muster der Bildpunkte (das Bildpunktepaar des CGA-Bildschirms erscheint ja getrennt auf dem Monochrom-Bildschirm). Allerdings können die Farben "blau" und "violett" schlecht unterschieden werden; beide erscheinen grau. Daher sollten Sie diese Farbkombination vermeiden.
Der leere Hintergrundschirm erscheint immer schwarz und kann durch das Kommando CALL IGH nicht beeinflußt werden.
Die Emulation des CGA-Schirms auf dem Monochromschirm kann auch im Zusammenhang mit anderen Softwarepaketen oder BASIC-Programmen benutzt werden.

Das wichtigste Stück Software der Diskette ist der Interfacetreiber FISCHER.COM. Er stellt alle Maschinenprogramme zur Steuerung des Interface, aber auch für die grafische Bildschirmausgabe zur Verfügung.
Das Begrüßungsprogramm FISCHER .BAS ist ein BASIC-Programm. Nach der Ausgabe der Bildschirmmeldungen (evt. mit wichtigen Neuerungen) verzweigt es zu dem Anpassungsprogramm INITBAS .BAS.
Der Schlüssel zu den Maschinenprogrammen des Interfacetreibers wird durch das Programm INIT.BAS bereitgestellt. Es wird immer zuerst geladen und dann die Beispiele eingetippt.
Die Beispielprogramme zu den verschiedenen Experimenten sind ebenfalls BASIC-Programme. Wenn Sie diese Programme benutzen wollen, benötigen Sie den "Schlüssel" INIT.BAS nicht, denn die entsprechenden Zeilen sind in jedem Beispielprogramm bereits enthalten.
Die Hintergrundbilder auf der Diskette sind mit der Namenserweiterung ".PIC" versehen. Diese Namenserweiterung geben Sie jedoch nicht an, wenn Sie die Kommandos CALL IGLOAD und CALL IGSAVE benutzen.

Auf der Arbeitsdiskette können Dateien mit der Namenserweiterung ".DAT" erscheinen. Diese Dateien enthalten Schildkröten-Routen, die von den Programmen ROUTEACH.BAS oder ROUTEDIT.BAS erzeugt wurden.

A 1.6 Anpassung des Interfacetreibers

IBM Personal Computer und kompatible Computer decken einen weiten Bereich von Rechenleistung ab. Dank ihres offenen Konzeptes besitzen sie die äußerste Anpassungsfähigkeit an die Bedürfnisse des Benutzers.
Auf der anderen Seite bedeutet dies, daß fast jedes Softwarepaket an die speziellen Betriebsbedingungen des PC angepaßt werden muß. Diese Anpassung kann teils automatisch erfolgen; in anderen Abschnitten muß der Benutzer die nötige Information bereitstellen.
Einige Eigenschaften des PC stellt der Interfacetreiber bei jedem Start fest:
FISCHER.COM prüft z.B. alle Druckeranschlüsse. Wenn nur einer vorhanden ist, so wird angenommen, daß an diesem das Interface angeschlossen ist. Wenn dagegen mehrere Druckeranschlüsse vorhanden sind, so frägt das Programm nach dem zu benutzenden. Sie können dieses Frage- und Antwortspiel aber auch vermeiden, wenn Sie gleich beim Aufruf von FISCHER.COM die richtige Druckernummer angeben:

A>**FISCHER/Dn**

wobei n die Nummer des Druckeranschlus-

ses ist (1,2 oder 3).
Die Arbeitsgeschwindigkeit des Computers beeinflußt stark die Kalibrierung der Analogkanäle, denn der Analogwert ist nichts weiter als die Zahl der Schleifendurchläufe, die der Computer während eines Impulses des Interface durchführen kann.
FISCHER.COM prüft daher das ROM-BIOS nach der Kennzeichnung des PC und nimmt eine grobe Kalibrierung in Schritten von jeweils eines Faktors zwei vor. Wenn eine unbekannte Kennzeichnung vorgefunden wird, so frägt das Programm nach einer Geschwindigkeitsangabe. Auch hier können Sie die richtige Antwort gleich vorausschicken:

A>**FISCHER/Cn**

wobei n die Werte 1 bis 4 annimmt. 1 steht für die Geschwindigkeitsklasse eines XT (4,77 MHz), 2 für einen schnellen XT oder einen AT, 3 für einen schnellen AT und 4 für die noch schnelleren AT-386. Da die schnellen Versionen der Computer mittlerweile eher zur Regel als zur Ausnahme zählen (z.B. 10 MHz XT-Computer), wird die Kennzeichnung des ROM-BIOS bezüglich der Geschwindigkeit häufig irreführend sein. Im genannten Beispiel wird sich daher auf jeden Fall empfehlen, den Interfacetreiber mit n=2 aufzurufen. Wenn n=0 gewählt wird, so zwingen Sie das Programm zu einer Abfrage der Rechnergeschwindigkeit.
Die Geschwindigkeitsanpassung ist grob und nur dafür gedacht, um die Funktionsfähigkeit des Interface zu gewährleisten. Der Heißleiter, der in Kapitel 7 und 8 eingesetzt wird, erfordert genauere Kalibrierungsmethoden. Es ist daher unbedingt notwendig, daß Sie die Kalibrierung mithilfe experimenteller Daten und des Programms KALIBR.BAS vornehmen, wie sie in Kapitel 7 beschrieben ist.
Manchmal werden FOR...NEXT-Schleifen zur Einhaltung von Zeitabschnitten verwendet. Die Zahl der Schleifendurchläufe muß ggf. auch auf die Geschwindigkeit des Computers abgestimmt werden.
Die Anpassung an weitere Eigenschaften des Computers erfolgt wiederum automatisch durch das Programm INITBAS.BAS wie in Kapitel 3 beschrieben.

CALL I1A **Motor 1 ausschalten**
Das Kommando schaltet Motor 1 ab. Es entspricht dem Kommando CALL M1(AUS) der Interface-Anleitung.

CALL I1L **Motor 1 Linkslauf**
Das Kommando schaltet Motor 1 in Linkslauf. Es entspricht dem Kommando CALL M1(LINKS) der Interface-Anleitung.

CALL I1R **Motor 1 Rechtslauf**
Das Kommando schaltet Motor 1 in Rechtslauf. Es entspricht dem Kommando CALL M1(RECHTS) der Interface-Anleitung.

CALL I1V **Motor 1 vorwärts**
Das Kommando startet Motor 1 in Linkslauf. Anschließend prüft das Kommando, ob Eingang E2 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL I1Z **Motor 1 zurück**
Das Kommando startet Motor 1 in Rechtslauf. Anschließend prüft das Kommando, ob Eingang E2 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL I2A **Motor 2 ausschalten**
Das Kommando schaltet Motor 2 ab. Es entspricht dem Kommando CALL M2(AUS) der Interface-Anleitung.

CALL I2L **Motor 2 Linkslauf**
Das Kommando schaltet Motor 2 in Linkslauf. Es entspricht dem Kommando CALL M2(LINKS) der Interface-Anleitung.

CALL I2R **Motor 2 Rechtslauf**
Das Kommando schaltet Motor 2 in Rechtslauf. Es entspricht dem Kommando CALL M2(RECHTS) der Interface-Anleitung.

CALL I2V **Motor 2 vorwärts**
Das Kommando startet Motor 2 in Linkslauf. Anschließend prüft das Kommando, ob Eingang E4 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL I2Z **Motor 2 zurück**
Das Kommando startet Motor 2 in Rechtslauf. Anschließend prüft das Kommando, ob Eingang E4 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL I3A **Motor 3 ausschalten**
Das Kommando schaltet Motor 3 ab. Es entspricht dem Kommando CALL M3(AUS) der Interface-Anleitung.

CALL I3L **Motor 3 Linkslauf**
Das Kommando schaltet Motor 3 in Linkslauf. Es entspricht dem Kommando CALL M3(LINKS) der Interface-Anleitung.

CALL I3R **Motor 3 Rechtslauf**
Das Kommando schaltet Motor 3 in Rechtslauf. Es entspricht dem Kommando CALL M3(RECHTS) der Interface-Anleitung.

CALL I3V **Motor 3 vorwärts**
Das Kommando startet Motor 3 in Linkslauf. Anschließend prüft das Kommando, ob Eingang E6 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL I3Z **Motor 3 zurück**
Das Kommando startet Motor 3 in Rechtslauf. Anschließend prüft das Kommando, ob Eingang E6 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL I4A **Motor 4 ausschalten**
Das Kommando schaltet Motor 4 ab. Es entspricht dem Kommando CALL M4(AUS) der Interface-Anleitung.

CALL I4L **Motor 4 Linkslauf**
Das Kommando schaltet Motor 4 in Linkslauf. Es entspricht dem Kommando CALL M4(LINKS) der Interface-Anleitung.

CALL I4R **Motor 4 Rechtslauf**
Das Kommando schaltet Motor 4 in Rechtslauf. Es entspricht dem Kommando CALL M4(RECHTS) der Interface-Anleitung.

CALL I4V **Motor 4 vorwärts**
Das Kommando startet Motor 4 in Linkslauf. Anschließend prüft das Kommando, ob Eingang E8 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL I4Z **Motor 4 zurück**
Das Kommando startet Motor 4 in Rechtslauf. Anschließend prüft das Kommando, ob Eingang E8 freigegeben (0) und anschließend wieder betätigt (1) wurde. Der Motor wird dann abgeschaltet.

CALL IDE(E1,...,E8) Digital-Eingabe
Das Kommando liest die digitalen Eingänge E1 bis E8 ein. Der Zahlenwert (0 für offen, 1 für mit +5V verbunden) wird in den Variablen E1 bis E8 abgelegt.

CALL IEX(EX) Analog-Eingabe EX
Das Kommando ermittelt den Widerstandswert, der an dem Eingang EX angeschlossen ist. Ein Widerstandswert von 0 Ohm (direkt mit +5V verbunden) ergibt einen kleinen Zahlenwert (ca. 20), ein Widerstandswert von 5 kOhm einen großen Zahlenwert (ca. 200). Das Verhalten für Widerstandswerte über 5 kOhm ist von Computer zu Computer unterschiedlich. Bei einigen Computer ist der größte Zahlenwert etwa 300, bei anderen gehen die Zahlenwerte weiter. Der Zahlenwert wird in der Variablen EX abgelegt.

CALL IEY(EY) Analog-Eingabe EY
Das Kommando ermittelt den Widerstandswert, der an dem Eingang EX angeschlossen ist, siehe auch Kommando CALL IEX. Der Zahlenwert wird in der Variablen EY abgelegt.

CALL IG0 Grafikstift aus
Das Kommando schaltet den Stift der Grafik-Schildkröte ab. Bei den folgenden Kommandos CALL IGV oder CALL IGZ wird keine Linie gezeichnet.

CALL IG1 Grafikstift ein
Das Kommando schaltet den Stift der Grafik-Schildkröte ein. Bei den folgenden Kommandos CALL IGV oder CALL IGZ wird eine Linie in der gewählten Stiftfarbe gezeichnet. Dies ist der Anfangszustand nach dem ersten Kommando CALL IGE.

CALL IGA Grafik aus
Das Kommando schaltet die Schildkrötengrafik ab. Um zum Textschirm zu gelangen, muß noch die Anweisung SCREEN 0 gegeben werden.

CALL IGC Grafik kopieren
Das Kommando kopiert das Bild des unsichtbaren Grafikschirms in den sichtbaren Grafikschirm. Der unsichtbare Grafikschirm ist entweder gelöscht (Hintergrundfarbe 0) oder durch das Kommando CALL IGLOAD mit einem Bild von der Diskette vorbesetzt. Auf diese Weise kann immer wieder ein "sauberer" Hintergrund erzeugt werden. Das Kommando setzt die Grafik-Schildkröte auf ihren Startpunkt; der Zustand des Grafikstiftes wird nicht verändert.

CALL IGE **Grafik einschalten**

Das Kommando teilt den Grafikbildschirm in einen Grafikbereich und einen Textbereich ein. Beide Schirme werden gelöscht, der Grafikbereich mit Hintergrundfarbe 0. Die Grafik-Schildkröte wird auf Ihren Startpunkt gesetzt und der Grafikstift in Farbe 1 eingeschaltet. Jeder nachfolgende Aufruf von CALL IGE ohne ein dazwischenliegendes Kommando CALL IGA löscht den Grafikbereich mit der zuvor gewählten Hintergrundfarbe. Der Grafikstift behält seinen zuvor gewählten Zustand (ein/aus, Farbe). Alle weiteren Grafikkommandos (mit Ausnahme CALL IGLOAD und CALL IGSAVE) erfordern, daß zuvor das Kommando CALL IGE gegeben wurde.

CALL IGF **Grafikfühler**
Das Kommando ermittelt die Punktfarbe, auf die die Grafik-Schildkröte zuletzt gefahren ist. Dies kann ein Punkt des Hintergrunds oder einer früher gemalten Linie sein. Die Punktfarbe (0 bis 3) wird in der Variablen GF zurückgegeben.

CALL IGH(F%) **Grafik-Hintergrund**
Das Kommando setzt die Hintergrundfarbe F% des Grafikschirms. Die Variable F% muß ganzahlig sein und im Bereich 0 bis 3 liegen. Die neue Hintergrundfarbe wird aber erst mit dem nächsten Kommando CALL IGE wirksam.

CALL IGK(GK) **Grafik-Kurs**
Das Kommando CALL IGK ermittelt den derzeitigen Kurs der Grafik-Schildkröte und legt ihn in der Variablen GK ab. Der Kurs ist 0° bei Blickrichtung nach oben, 90° bei Blickrichtung nach rechts, 180° bei Blickrichtung nach unten und 270° bei Blickrichtung nach links.

CALL IGL(D%) Grafik-Schildkröte links
Das Kommando dreht die Grafik-Schildkröte um D% Grad nach links. Die Variable D% muß ganzzahlig sein und im Bereich 0 bis 359 liegen.

CALL IGLOAD(F$) **Grafik laden**
Das Kommando lädt den Grafik-Hintergrundschirm von der Diskette. Das Bild ist mit dem in F$ enthaltenen Dateinamen, erweitert um die Dateitypkennzeichnung ".PIC", abgespeichert.

CALL IGR(D%) Grafik-Schildkröte rechts

Das Kommando dreht die Grafik-Schildkröte um D% Grad nach rechts. Die Variable D% muß ganzzahlig sein und im Bereich 0 bis 359 liegen.

CALL IGS(F%) Grafikstift

Das Kommando wählt die Stiftfarbe der Grafik-Schildkröte. Die Variable F% muß ganzahlig sein und die Werte 0, 1, 2 oder 3 annehmen.
Palette 1 enthält folgende Farben:
0 = Schwarz, 1 =Cyan, 2 = Magenta,
3 = Weiß.
Palette 0 enthält folgende Farben:
0 = Schwarz, 1 = Grün, 2 = Rot, 3= Gelb.

CALL IGSAVE(F$) Grafik speichern

Das Kommando schreibt den sichtbaren Grafikschirm auf die Diskette. Das Bild wird mit dem in F$ enthaltenen Dateinamen, erweitert um die Dateitypkennzeichnung ".PIC", abgespeichert.

CALL IGV(S%) Grafik-Schildkröte vorwärts

Das Kommando bewegt die Grafik-Schildkröte um S% Schritte vorwärts. Die Variable S% muß ganzahlig sein und im Wertebereich 0 bis 32767 liegen. Schritte, die größer als ca. 100 sind, lassen die Schildkröte meist vom Bildschirm verschwinden. Dennoch ist sie nicht verloren, die Position wird weiter berechnet, und durch geeignete Kommandos kann sie auch wieder auf den Schirm gebracht werden.
Wenn der Grafikstift eingeschaltet war, wird von der bisherigen Position zur Zielposition eine Linie in der eingestellten Farbe (s. Kommandos CALL IGE und CALL IGS) gezeichnet.

CALL IGX(GX) Grafik-Schildkröte X-Position

Das Kommando ermittelt die X-Position der Grafik-Schildkröte und legt sie in der Variablen GX ab. Die Koordinate X ist 0 im Startpunkt der Grafik-Schildkröte, nach rechts positiv und nach links negativ.

CALL IGY(GY) Grafik-Schildkröte Y-Position

Das Kommando ermittelt die Y-Position der Grafik-Schildkröte und legt sie in der Variablen GY ab. Die Koordinate Y ist 0 im Startpunkt der Grafik-Schildkröte, nach oben positiv und nach unten negativ.

CALL IGZ(S%) Grafik-Schildkröte zurück
Das Kommando bewegt die Grafik-Schildkröte um S% Schritte zurück. Weitere Details s. Kommando CALL IGV.

CALL IHA Zeilenverdopplung aus
Das Kommando schaltet die Verdopplung jeder zweiten Zeile bei der CGA-Emulation auf einem Monochromadapter ab. Damit treten keine Verzögerungen bei den Schritt- und Schildkrötenkommandos auf.

CALL IHE Zeilenverdopplung ein
Das Kommando schaltet die Verdopplung jeder zweiten Zeile bei der CGA-Emulation auf einem Monochromadapter ein. Damit wird die Bildfläche gleichmäßig gefüllt und es entsteht kein Streifenmuster. Dies ist die Standardeinstellung.

CALL IIN Initialisierung
Das Kommando initialisiert das Interface. Alle anderen Kommandos erfordern, daß zuvor das CALL IIN-Kommando gegeben wurde, weil das Kommando CALL IIN Systemgrößen ermittelt, Variablen einrichtet und dergleichen Vorbereitungsarbeiten durchführt.

CALL ITB(B%) Schildkrötenbremse
Das Kommando CALL ITB beeinflußt die Gegenstrombremse. Bei allen Schritt- und Schildkrötenkommandos CALL I1V, CALL I1Z,...,CALL ITV, CALL ITZ, CALL ITR, CALL ITL wird nach Beendigen des Schritts der Strom durch den Motor noch einmal kurz umgedreht und dann abgeschaltet. Dadurch bleibt der Motor schlagartig stehen. Die Dauer des Gegenstromimpulses wird durch das Kommando CALL ITB eingestellt. Die Variable B% ist ganzzahlig und liegt im Wertebereich1 bis 255. Das Kommando CALL ITB werden Sie normalerweise nicht benötigen, da der Fehlwert bereits optimal eingestellt wurde.

CALL ITI Schildkröteninitialisierung
Das Kommando CALL ITI ist vor dem Gebrauch weiterer Schildkrötenkommandos notwendig. Es legt den derzeitigen Standort und Kurs der Schildkröte (X- und Y-Position, Kurs) mit 0 fest. Das Kommando ist jederzeit später auch verwendbar, um einen neuen Standort als Ausgangspunkt festzulegen.

CALL ITK(TK) Schildkrötenkurs
Das Kommando CALL ITK ermittelt den

derzeitigen Kurs der Schildkröte und legt ihn in der Variablen TK ab. Der Kurs ist 0° bei Blickrichtung in Startrichtung, 90° bei Blickrichtung nach rechts, 180° bei Blickrichtung gegen die Startrichtung und 270° bei Blickrichtung nach links.

CALL ITL(D%,TS) Schildkröte links
Das Kommando dreht die Schildkröte um D% Grad nach links. Die Variable D% muß ganzzahlig und durch fünf teilbar sein und im Bereich 0 bis 355 liegen. Das Kommando setzt die Zahl der durchgeführten Schritte (nicht Winkelgrade!) in die Variable TS. Der Eingang E5 wird - im Gegensatz zu dem Kommando CALL ITV - nicht geprüft.

CALL ITR(D%,TS) Schildkröte rechts
Das Kommando dreht die Schildkröte um D% Grad nach rechts. Die Variable D% muß ganzzahlig und durch fünf teilbar sein und im Bereich 0 bis 355 liegen. Das Kommando setzt die Zahl der durchgeführten Schritte (nicht Winkelgrade!) in die Variable TS. Der Eingang E5 wird - im Gegensatz zu dem Kommando CALL ITV - nicht geprüft.

CALL ITV(S%,TS,E5) Schildkröte vorwärts
Das Kommando bewegt die Schildkröte um S% Schritte vorwärts. Ein Schritt ist 5 mm lang. Die Variable S% muß ganzzahlig sein und im Wertebereich 0 bis 32767 liegen. Das Kommando CALL ITV prüft den Eingang E5 (bei der Schildkröte die Stoßstange). Ist E5 geöffnet (0), wird das Kommando abgebrochen. Bei Beendigung des Kommandos werden die Variablen E5 und TS gesetzt. Die Variable E5 enthält den Zustand des Eingangs E5, die Variable TS die Anzahl der erfolgreich ausgeführten Schritte. Die Zahl in TS kann kleiner als S sein, wenn E5 vor Bahnende geöffnet wurde. Durch Kontrolle von E5 und TS kann also ermittelt werden, ob und wann eine Kollision stattgefunden hat.

CALL ITX(TX) Schildkröte X-Position
Das Kommando ermittelt die X-Position der Schildkröte und legt sie in der Variablen TX ab. Die Koordinate X ist 0 im Startpunkt der Schildkröte, nach rechts positiv und nach links negativ.

CALL ITY(TY) Schildkröte Y-Position
Das Kommando ermittelt die Y-Position der Schildkröte und legt sie in der Variablen TY ab. Die Koordinate Y ist 0 im Startpunkt der Schildkröte, in Startrichtung positiv und gegen die Startrichtung negativ.

CALL ITZ(S%,TS) Schildkröte zurück
Das Kommando bewegt die Schildkröte um S% Schritte zurück. Das Kommando setzt die Zahl der durchgeführten Schritte in die Variable TS. Der Eingang E5 wird - im Gegensatz zu dem Kommando CALL ITV - nicht geprüft.

CALL IW40 40-Zeichen-Modus

Das Kommando ist eine Erweiterung der BASIC-Anweisung WIDTH 40 insofern, als sie auch im Zusammenhang mit Monochrom-Bildschirmadaptern eingesetzt werden kann.

CALL IW80 80-Zeichen-Modus

Das Kommando ist eine Erweiterung der BASIC-Anweisung WIDTH 80 insofern, als sie auch im Zusammenhang mit Monochrom-Bildschirmadaptern eingesetzt werden kann.

Bildnachweis

Folgenden Firmen danken wir für die Bereitstellung von Bildmaterial zur Gestaltung dieses Experimentierhandbuchs:

Bleichert Förderanlagen GmbH, Osterburken, S. 81 und 93,

Valvo Unternehmensbereich Bauelemente der Philips GmbH, Hamburg, S. 42,

Wagner Fördertechnik GmbH&Co. KG, Reutlingen, S. 110.

Art.-Nr: 39510 · 0389 · Printed in Germany · Technische Änderungen vorbehalten · RS