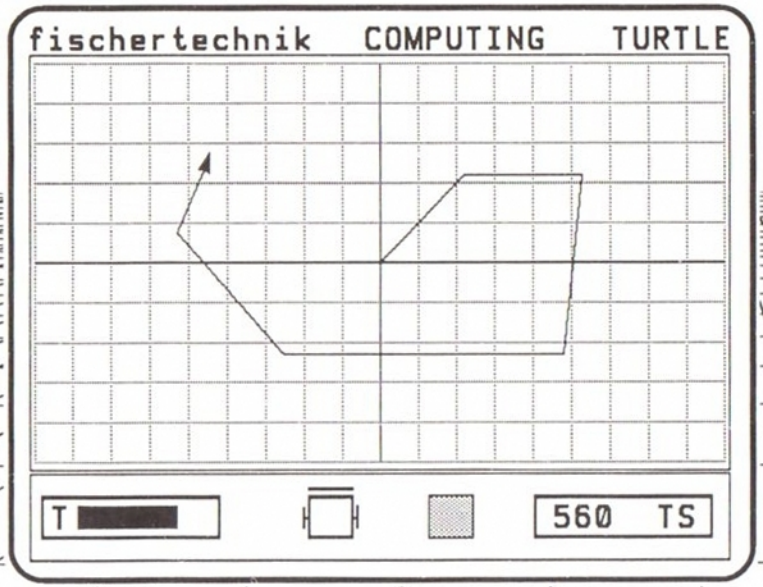


Experimentierhandbuch Commodore Amiga ®

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 Commodore-Computer der Amiga-Serie. Bitte vergewissern Sie sich, daß Ihr Computer ein Amiga 500 oder Amiga 2000 ist. Der Computer Amiga 1000 läßt sich bei Anfertigung eines Übergangskabels für die Druckerschnittstelle ebenfalls benutzen.
Der fischertechnik COMPUTING EXPERIMENTAL Baukasten kann mit anderer Software und anderem Handbuch auch an den Computern C64/128, Atari ST, IBM-PC und Kompatiblen 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 etwas 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 3½"-Diskette im 880K-Format. Darauf kommen wir im Kapitel 3 zu sprechen.

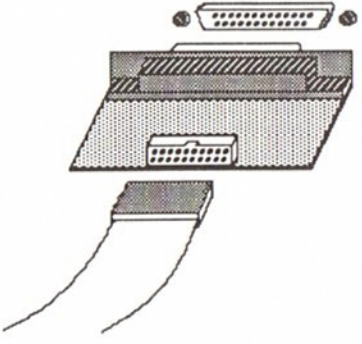


Bild 2.1: Das Interface wird mit Hilfe des Adapters an die Druckerschnittstelle angeschlossen

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 ein 25-poliger trapezförmige Buchsenstecker. Beim Amiga 1000 ist die Druckerschnittstelle ein 25-poliger Stiftstecker. In diesem Fall müssen Sie sich bei Ihrem Computerhändler ein Übergangskabel besorgen oder entsprechend der Anleitung in der Interface-Anleitung selbst zusammenlöten.
Schließen Sie nun den Interface-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 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 Bauanlei-

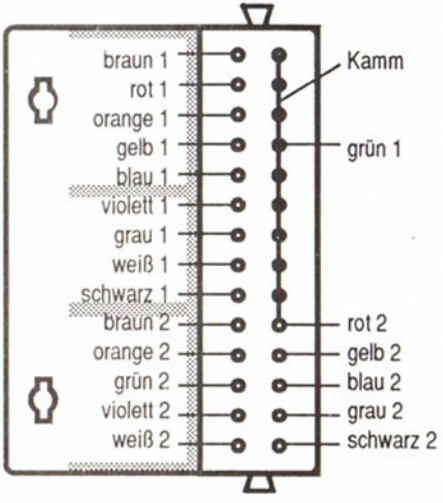


Bild 2.2: Anschluß des Interfacekabels an die 28-polige Steckbuchse. Beachten Sie, daß die Adern grün 1 und rot 2 zusammen mit dem Kamm montiert werden. (Ansicht von unten)

tung), 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 beide 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 Bauanleitung. 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,

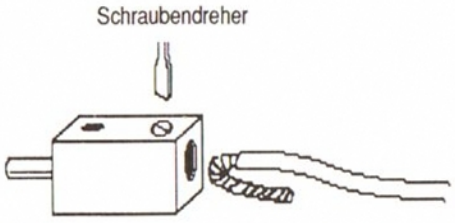
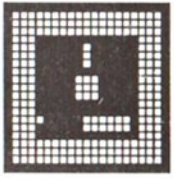


Bild 2.3: Montage eines fischertechnik Steckers

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.

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. Versehen Sie die Kabel mit fischertechnik Steckern. 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.
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 DISKCOPY-Programm der Amiga-WORKBENCH (in der System-Schublade) oder ein beliebiges anderes Kopierprogramm benutzen. Verfahren Sie so, wie es das AmigaDOS-Handbuch des Computers vorschreibt. Zum Schluß des Kopiervorgangs aktivieren Sie das Symbol der Kopiediskette und rufen die **Rename**-Funktion des Rolladens (pull-down-Menü) **Workbench** auf. Löschen Sie durch Betätigung der Del-Taste die Worte "Copy of" aus dem Diskettennamen. Drücken Sie die Eingabetaste (Enter, Return).

Später werden Sie sich sogar noch eine zweite Kopie ziehen. Eine Kopie dient zum Gebrauch der Beispielprogramme auf der Diskette. Von der anderen Kopie werden Sie die Beispielprogramme löschen, jedoch nicht die Interface- und Systemprogramme (s. nachstehende Beschreibung). Diese Diskette dient Ihnen als Arbeitsdiskette, wenn Sie die in diesem Experimentierhandbuch beschriebenen Experimente durchführen. Die Aufforderung zur Erstellung von Kopien 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 fi-

schertechnik 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.

In der folgenden Beschreibung gehen wir davon aus, daß der Amiga Computer mit einem Diskettenlaufwerk ausgestattet ist. Besitzer eines Amiga mit zwei Laufwerken oder einer Festplatte werden das nachfolgende Verfahren vereinfachen können, indem Sie Zugriff auf zwei Disketten gleichzeitig haben.

Für die nachfolgenden Experimente benötigen Sie ein Programmstück, das den Schlüssel zu dem Interface darstellt. Dieses Programm wird Interfacetreiber genannt und besteht aus der Library (Bibliothek) EXPER.LIBRARY. Diese ist auf der fischertechnik COMPUTING EXPERIMENTAL Diskette enthalten. Die Library enthält eine Sammlung von Unterprogrammen, die den Computer mit einer Reihe neuer "Befehle" versehen, die bislang noch nicht vorhanden waren. Durch Aufruf der Unterprogramme werden die fischertechnik Bauelemente über das Interface per Programm gesteuert.

Außerdem benötigen Sie AmigaBASIC, denn die ganzen Steuerprogramme für die Maschinen, Meßwerterfassungen und Roboter werden in BASIC programmiert. AmigaBASIC befindet sich auf der Diskette EXTRAS, die Sie mit Ihrem Amiga-Computer geliefert bekommen haben. AmigaBASIC muß auch auf die Kopie der fischertechnik COMPUTING EXPERIMENTAL Diskette kopiert werden, um auch den BASIC-Interpreter in ständigem Zugriff zu haben.

Letztendlich werden auch noch einige Programme des Betriebssystems AmigaDOS benötigt. Um mit fischertechnik COMPUTING EXPERIMENTAL zu experimentieren, müssen Sie nicht wissen, welche Betriebssystemfunktionen dies im einzelnen sind. Sie sind jedoch für so elementare Funktionen wie das Rechnen, die Ausgabe auf den Drucker usw. notwendig. Die Betriebssystemfunktionen befinden sich auf der Amiga WORKBENCH-Diskette und müssen ebenfalls auf die Kopie der fischertechnik COMPUTING EXPERIMENTAL Diskette kopiert werden. Um die anstehenden Kopiervorgänge automatisch ablaufen zu lassen, gibt es ein entsprechendes Programm auf der fischertechnik COMPU-

Sollte Ihnen das Klicken, Doppelklicken und Ziehen mit der Maus, das Öffnen der Rolladen, das Verschieben und Verändern der Fenster noch nicht geläufig sein, so schlagen Sie die dazugehörigen Kapitel des Handbuchs Ihres Computers nach.

TING EXPERIMENTAL Diskette. Legen Sie die Kopie der fischertechnik COMPUTING EXPERIMENTAL Diskette in ein Laufwerk und halten Sie die EXTRAS- und WORKBENCH-Diskette parat. Die Kopie der fischertechnik COMPUTING EXPERIMENTAL Diskette darf nicht schreibgeschützt sein!

Öffnen Sie das Diskettensymbol FISCHER, indem Sie mit dem Mauszeiger auf das Diskettensymbol zeigen und zweimal kurz hintereinander die linke Maustaste betätigen. Nun öffnet sich ein Fenster, in dem u.a. auch das Symbol eines leeren Zimmers zu sehen ist. Die Unterschrift des Symbols lautet EINRICHTEN. Doppelklikken Sie das Symbol und sofort füllt sich das Zimmer mit Möbel. Das Diskettenlaufwerk läuft an und auf dem Bildschirm erscheint eine Information zum weiteren Ablauf. Folgen Sie den Aufforderungen des Programms und legen Sie die jeweils benötigten Disketten in ein Diskettenlaufwerk. Wenn Ihr Computer nur mit einem Laufwerk ausgestattet ist, werden Sie relativ häufig die Disketten wechseln müssen. Wenn der Computer mit zwei Diskettenlaufwerken ausgestattet ist, sollte neben der fischertechnik COMPUTING EXPERIMENTAL Diskette vor allen Dingen die WORKBENCH zur Verfügung stehen.

Nach etwa zehn Minuten sind alle Kopiervorgänge abgelaufen und Sie besitzen jetzt alles, was Sie an Software zum Experimentieren benötigen, auf einer Diskette. Diese Diskette ist gemeint, wenn wir im Folgenden von der fischertechnik COMPUTING EXPERIMENTAL Diskette oder kurz fischertechnik-Diskette reden.

Sollten bei dem Kopiervorgang Fehler aufgetreten sein, so schlagen Sie die Hinweise im Anhang nach. Dort finden Sie eine Liste der Punkte, die Sie ggf. vor dem nächsten Versuch kontrollieren sollten.

Von der fischertechnik Diskette erstellen Sie sich eine zweite Kopie. Gehen Sie wieder wie zuvor beschrieben vor, d.h. benutzen Sie z.B. das Programm DISKCOPY der WORKBENCH. Nach dem Kopiervorgang löschen Sie wieder die Bezeichnung "copy of" in dem Diskettennamen mit Hilfe der **Rename**-Funktion. Nun haben Sie zwei Disketten mit dem Namen FISCHER. Öffnen Sie das Diskettensymbol einer der beiden Disketten. In dem Fenster, das sich jetzt auf dem Bildschirm auftut, finden Sie eine Schublade mit dem Namen EXPERIMENTAL. Öffnen Sie auch diese Schublade durch Doppelklick. Nun sehen Sie die Liste der ganzen Beispielprogramme auf

dem Bildschirm vor sich. Die meisten dieser Beispielprogramme sollen Sie nun löschen, denn Sie benötigen Platz auf der Diskette, um die Programme zu den kommenden Experimenten zu speichern. Die Beispielprogramme stehen Ihnen ja noch auf der anderen Diskette zur Verfügung. Aktivieren Sie daher alle Programmsymbole außer INIT und DIAGNOSE. Drücken Sie dazu die Hochstelltaste (Shift ⇧) und klicken Sie ein Symbol nach dem anderen an. Wenn alle Programmsymbole erfaßt sind, wählen Sie unter dem Rolladen **Workbench** die Funktion **Discard** an. Nach Bestätigung der Kontrollabfrage werden die Beispielprogramme gelöscht. Übrig bleiben nur INIT und DIAGNOSE. Die so entstandene Diskette nennen wir im Folgenden die Arbeitsdiskette.

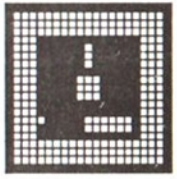
Legen Sie diese Arbeitsdiskette in das Laufwerk DF0: ein und fahren Sie Ihren Computer neu an. Drücken Sie dazu gleichzeitig die Tasten Ctrl und die beiden Tasten mit dem Amiga-Symbol links und rechts der Leertaste. Nach dem Anfahren des Computers erscheint wieder das Diskettensymbol FISCHER. Öffnen Sie das Symbol durch Doppelklick, um das fischertechnik-Fenster zu erhalten. Laden Sie den BASIC-Interpreter, indem Sie das Symbol AMIGA-BASIC doppelklicken. Wenn der BASIC-Interpreter geladen ist, erscheinen zwei Fenster auf dem Bildschirm. Das zuoberst liegende Fenster ist das List-Fenster. In diesem Fenster befindet sich normalerweise der Text des BASIC-Programms. Derzeit ist es allerdings noch leer, denn es wurde noch kein Programm geladen. Das darunterliegende Fenster dient der Programmausgabe und der Eingabe von Befehlen (Ausgabe-Fenster).

Aktivieren Sie das Ausgabefenster, indem Sie mit der Maus in das Fenster zeigen und die linke Maustaste betätigen. Geben Sie nun den Befehl

```
CHDIR"EXPERIMENTAL"
```

ein. Mit diesem Befehl verschaffen Sie sich Zugriff zu den Programmen in der Schublade EXPERIMENTAL.

Laden Sie das Programm INIT. Dazu drükken Sie die rechte Maustaste. Der Menübalken ändert sich und es erscheint **Project**, **Edit**, **Run** und **Windows**. Ziehen Sie den Mauszeiger auf **Project**. Es öffnet sich der Rolladen mit den Dateibefehlen. Ziehen Sie die Maus auf **Open** und lassen Sie die Taste los. Es erscheint ein Datei-



Auswahlfenster, in dem Sie den Programmnamen INIT eingeben und zum Abschluß die Eingabetaste drücken.
Wenn das List-Fenster nicht erscheinen sollte, so wählen Sie (wie eben für das Laden beschrieben) die Funktion **Show List** unter dem Rolladen **Windows** aus. Nun sehen Sie den Programmtext von INIT.
Das Programm INIT ist ein Bruchstück eines BASIC-Programms. Es enthält alle Befehle, die in immer wieder gleicher Form anfallen: um Zugriff auf den Interfacetreiber zu haben, ein Dialogfenster zur Verfügung zu stellen, ein geordnetes Ende des Programms zu gewährleisten usw. Um die nachfolgende Diskussion der Experimente nicht mit diesen Verwaltungsarbeiten zu belasten, wurde das Programm INIT zur Verfügung gestellt.
Aktivieren Sie das List-Fenster, indem Sie mit der Maus hineinzeigen und die linke Maustaste betätigen. Halten Sie nun die Hochstelltaste fest und betätigen Sie die Pfeiltaste ↓. Sie finden im Programmtext die Zeilen:

```
REM Programmanfang

REM Programmende
```

Zwischen diese beiden Zeilen wird das eigentlich Steuerungsprogramm eingefügt. In den folgenden Kapiteln werden wir nur die Zeilen angeben, die an dieser Stelle eingefügt werden, denn die übrigen Zeilen sind immer gleich, haben keinen Einfluß auf die Logik des Experiments (wohl aber dessen Funktionsfähigkeit!) und brauchen auch nicht unbedingt verstanden werden. Sollten Sie sich dennoch für die Zeilen interessieren, so schlagen Sie die Informationen im Anhang nach.
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. Dabei gehen Sie folgendermaßen vor:

- Speichern Sie ggf. das bisherige Programm im Arbeitsspeicher. Wählen Sie dazu den Menüpunkt **Save** bzw. **Save As** aus dem Rolladen **Project** und geben Sie im Datei-Auswahlfenster den gewünschten Dateinamen an.
- Laden Sie, wenn dies gefordert wird, das Programm INIT wie zuvor beschrieben: also rechte Maustaste drücken, **Project** anwählen, **Open** aussuchen und INIT im Datei-Auswahlfenster angeben.

Um die Beispielprogramme zu starten, brauchen Sie übrigens nicht eigens AmigaBASIC zu laden. Öffnen Sie das fischertechnik-Fenster, dann die EXPERIMENTAL-Schublade und doppelklicken Sie das gewünschte Programm. AmigaBASIC wird automatisch mitgeladen.

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 und von AmigaBASIC studieren und eventuell auch einen BASIC-Programmierkurs durchnehmen.

- Aktivieren Sie das List-Fenster
- Setzen Sie die Eingabemarke (Cursor) zwischen die Zeilen

```
REM Programmanfang
```

und

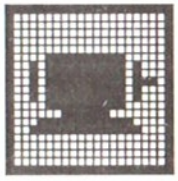
```
REM Programmende
```

indem Sie die dazwischenliegende Leerzeile anklicken.
- Geben Sie, beginnend in dieser Zeile, das Programm zu dem Experiment ein. Nach Beendigung einer jeden Zeile erscheint eine neue freie Zeile (Eingabe und Verbesserung von BASIC-Programmen ist in dem AmigaBASIC-Handbuch beschrieben.)
- Testen Sie das Programm, wie in der Experimentbeschreibung angegeben. Verbessern Sie ggf. aufgetretene Programmier- oder Tippfehler.
- Das Programm sollten Sie, wenn es funktioniert, auf die Arbeitsdiskette (nicht die fischertechnik Diskette!) abspeichern.

Kommen Sie mal mit einem Programm gar nicht klar oder wollen Sie schnell ein Programm vorführen oder sich ein paar Anregungen zum Verschönern der Programme holen, so greifen Sie zur fischertechnik Diskette. 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. Das Laden der Beispielprogramme erfolgt auf ähnliche Weise:

- Speichern Sie ggf. das bisherige Programm im Arbeitsspeicher. Wählen Sie dazu den Menüpunkt **Save** oder **Save As** und geben Sie im Datei-Auswahlfenster den gewünschten Dateinamen an.
- Wählen Sie **Open** an. Wählen Sie den Namen des gewünschten Programms aus und laden Sie es.
- Starten Sie das Programm mit **Run**.
- Testen Sie das Programm, wie in den Bildschirmmeldungen angegeben.

Ein Programm der fischertechnik-Diskette, das Sie in jedem Fall ansehen sollten, ist das Programm FISCHER. Es bringt eine kurze Inhaltsangabe der Beispielprogramme. Außerdem kann es Informationen enthalten, die zum Zeitpunkt der Drucklegung dieses Experimentierhandbuchs noch nicht vorlagen. Halten Sie also danach Ausschau!



Wenn nun alles vorbereitet ist, 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. Fahren Sie, wenn noch nicht geschehen, den Amiga Computer mit der Arbeitsdiskette an. Öffnen Sie das Diskettensymbol und laden Sie AmigaBASIC (s. Kapitel 3). Laden Sie das Programm INIT. Setzen Sie die Eingabemarke in das Programm INIT zwischen die Zeilen

```
REM Programmanfang
```

und

```
REM Programmende
```

Geben Sie ein:

```
init
```

und drücken Sie die Eingabetaste. Es erscheint ine neue Leerzeile. Geben Sie ein:

```
m1r
```

Bewegen Sie jetzt den Mauszeiger an den oberen Bildschirmrand, drücken die rechte Maustaste und öffnen den Rolladen **Run**. Ziehen Sie die Maus bis zu dem Befehl **Start** und lassen Sie die Taste los. Nun beginnt das Diskettenlaufwerk zu arbeiten (der Interfacetreiber wird geladen), ein neues Bildschirmfenster mit dem Namen EXPERIMENTAL erscheint und kurz darauf dreht sich der Motor für etwa eine halbe Sekunde. Dann erscheint wieder das Ausgabe- und das Listfenster.
Der Befehl init (initialisiere Interface) versetzt das Interface in den Grundzustand; dies muß man immer am Anfang eines Programms machen.
Mit dem Befehl m1r wird der Motor 1 angesprochen. Er soll sich rechts herum drehen (m1r = Motor 1 rechts). Wenn rechts möglich ist, dann sicher auch links! Probieren Sie's aus! Setzen Sie die Eingabemarke an die betreffende Stelle im Programm, löschen Sie den Befehl m1r und geben Sie stattdessen:

```
m1l
```

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 m1r nur angenommen, wenn zuerst init gegeben wurde.

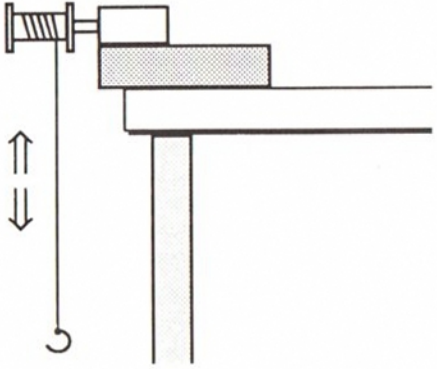


Bild 4.1: Eine Seilwinde kann man zum Experimentieren auch an einer Tischkante anbringen.

ein. Starten Sie wieder das Programm. 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. StellenSie 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 das Programm zu stoppen, indem Sie in dem **Run**-Rolladen die **Stop**-Funktion anwählen. Das bewirkt zwar den Abbruch 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.
Setzen Sie die Eingabemarke wieder zwischen die Bemerkungen Programmanfang und Programmende. Ergänzen Sie den Programmtext so, daß folgendes Programmstück vorliegt:

```
REM Programmanfang
init
Schleife:
  mlr
GOTO Schleife
REM Programmende
```

Wir möchten Sie daran erinnern, daß Sie zuvor das Programm INIT geladen hatten und jetzt die obigen vier Zeilen hinzugefügt haben. Ihr vollständiges Programm ist also beträchtlich länger. Zur Kontrolle:

```
LIBRARY "exper.library"

DECLARE FUNCTION e1& LIBRARY
DECLARE FUNCTION e2& LIBRARY
DECLARE FUNCTION e3& LIBRARY
DECLARE FUNCTION e4& LIBRARY
DECLARE FUNCTION e5& LIBRARY
DECLARE FUNCTION e6& LIBRARY
DECLARE FUNCTION e7& LIBRARY
DECLARE FUNCTION e8& LIBRARY
DECLARE FUNCTION ex& LIBRARY
DECLARE FUNCTION ey& LIBRARY
```

AmigaBASIC erlaubt für viele Befehle zwei oder drei verschiedene Arten der Befehlseingabe: eine mausorientierte, eine tastaturoriente und bei häufig gebrauchten Befehlen eine Kurzform. Um ein Programm zu starten können Sie folgendermaßen vorgehen:
a) mausorientiert:
Zeigen Sie mit dem Mauszeiger in die Titelleiste und drücken Sie die rechte Maustaste. Zeigen Sie auf den Rolladen **Run**, *der daraufhin herunterklappt. Ziehen Sie den Mauszeiger auf die Zeile* **Start** *und lassen Sie die rechte Maustaste los.*
b) tastaturorientiert:
Klicken Sie mit der Maus in das Ausgabefenster. Geben Sie ein:

RUN

c) Kurzform:
Halten Sie die rechte (nicht ausgefüllte) Amiga-Taste fest und geben Sie den Buchstaben R ein.

```
DECLARE FUNCTION gk& LIBRARY
DECLARE FUNCTION gx& LIBRARY
DECLARE FUNCTION gy& LIBRARY

DECLARE FUNCTION tk& LIBRARY
DECLARE FUNCTION tx& LIBRARY
DECLARE FUNCTION ty& LIBRARY
DECLARE FUNCTION ts& LIBRARY
DECLARE FUNCTION tb& LIBRARY

DEFLNG a-z

SCREEN 2,640,256,2,2
WINDOW 2,"EXPERIMENTAL",,0,2

ON BREAK GOSUB ende
BREAK ON

REM Programmanfang:
init
Schleife:
  m1R
GOTO Schleife:
REM Programmende

ende:
  LIBRARY CLOSE
  SCREEN CLOSE 2
  WINDOW CLOSE 2
  END
```

Für die nachfolgenden Experimente müssen Sie nicht die Bedeutung der Zeilen des INIT-Programms verstehen. Denken Sie nur daran, daß sie den notwendigen Schlüssel zu den Funktionen des Interface darstellen. Prüfen Sie immer bei jedem Programm für fischertechnik COMPUTING EXPERIMENTAL, daß der Interfacetreiber EXPER.LIBRARY auf der Diskette in der Schublade LIBS enthalten ist und durch obige Zeilen des Programms INIT angekoppelt werden. Außerdem muß vor dem Aufruf eines jeglichen Interface-Kommandos erst der Aufruf init ausgeführt wird. Beachten Sie auch noch: der genaue Wortlaut des Programms INIT kann je nach Version der Software auch von dem hier abgedruckten abweichen. Mehr Information über die Zeilen des Programms INIT erhalten Sie im Anhang.
Um das Anleitungsheft nicht mit dem Abdruck der DECLARE-Anweisungen und ähnlichen zu füllen, vereinbaren wir, daß immer nur der Teil zwischen
`REM Programmanfang` und
`REM Programmende`
einschließlich abgedruckt wird.
Starten Sie jetzt Ihr Programm durch **Start**. Der Motor läuft jetzt dauernd. Mit der Zeile m1r läuft er ein Stück, durch die Zeile mit

Das Anhalten eines Programms kann ebenfalls auf drei verschiedene Weisen erfolgen:
a) mausorientiert:
Zeigen Sie mit dem Mauszeiger in die Titelleiste und drücken Sie die rechte Maustaste. Zeigen Sie auf den Rolladen ***Run****, der daraufhin herunterklappt. Ziehen Sie den Mauszeiger auf die Zeile* ***Stop*** *und lassen Sie die rechte Maustaste los.*
b) tastaturorientiert:
Halten Sie die Taste Ctrl festen und drükken Sie den Buchstaben C. Die Tastenkombination Ctrl-C ist die klassische Form des Programmabbruchs, die nicht nur beim Amiga sondern auch auf vielen anderen Computern und Betriebssystemen funktioniert.
c) Kurzform:
Halten Sie die rechte (nicht ausgefüllte) Amiga-Taste fest und geben Sie das Satzzeichen Punkt (.) ein.

Es bleibt jedem überlassen, die ihm genehme Art der Bedienung zu wählen. Im Text geben wir jeweils die Bezeichnungen der Rolladen an, also ***Start*** *für Programmstart und* ***Stop*** *für den Programmabbruch. Zur Kennzeichnung erscheinen die Bezeichnungen in Fettschrift.*

der GOTO-Anweisung springt das Programm wieder zu der Zeile mit der Sprungmarke Schleife: 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 **Stop** an. Dies löst die Schutzschaltung aus und der Motor hält an. Im Interface ist jedoch noch gespeichert, daß der Motor zuletzt eingeschaltet wurde. Deshalb sollten wir, sofern die Programme keine Endlosschleife wie das vorliegende Programm enthalten, den Motor korrekt mit dem Befehl

```
m1a          (Motor 1 aus)
```

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 GOTO 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 m1r den Befehl m1l eingeben, läuft der Motor nach **Start** 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 m1r nur eine bestimmte Anzahl - hier 2000 mal - auszugeben, setzen wir ihn in eine sog. FOR...NEXT-Schleife:

```
REM Programmanfang
init
FOR z=1 TO 2000
  m1r
NEXT z
m1a
REM Programmende
```

Geben Sie das Programm anstelle des bisherigen Hauptprogramms ein. Dazu können Sie das bisherige mit **New** löschen, INIT mit **Open** wieder laden und das Programm eingeben. In den meisten Fällen geht es jedoch schneller, wenn Sie das bisherige Hauptprogramm als Block markieren und löschen. Danach wird das neue

In diesem und den folgenden Programmbeispielen haben wir die Struktur des Programms durch Einrückungen kenntlich gemacht. Die Einrückungen sind nicht notwendig, sie können auch weggelassen werden. Sie werden aber vom Editor des AmigaBASIC unterstützt und bereiten nicht viel Arbeit. Da sie die Lesbarkeit und Kontrolle der Programme verbessern, fanden wir sie durchaus sinnvoll.

Hauptprogramm eingegeben. Starten Sie es mit **Start**. 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? Die FOR-Anweisung enthält einen Zähler z. Er erhält den Anfangswert 1 (...z=1...). Danach wird die nächste Zeile ausgeführt: m1r d.h. der Motor 1 startet im Rechtslauf (bzw. setzt den Rechtslauf fort). Mit der NEXT-Anweisung wird der Zähler um 1 erhöht. Außerdem wird der Zählerstand geprüft. Solange der Zähler den Grenzwert (2000 in diesem Fall: ... TO 2000) nicht überschritten hat, springt die Programmausführung in die Zeile zurück, die der FOR-Anweisung folgt. Hat der Zähler z den Grenzwert überschritten, wird die Schleife verlassen und der nächste Befehl ausgeführt. Hier ist es m1a: der Motor wird ausgeschaltet. Wenn Sie die Schleife nur 1000 mal durchlaufen lassen wollen, ändern Sie die Zahl in der Schleifenanweisung:

```
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 die Zeile mit dem Motorbefehl:

```
m1l
```

Nach **Start** 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.

```
REM Programmanfang
init
FOR z=1 TO 2000
  m1r
NEXT z
FOR z=1 TO 2000
  m1l
NEXT z
m1a
REM Programmende
```

Mit **Start** 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:

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 das Programm INIT laden.

```
REM Pogrammanfang
init
Schleife:
  FOR z=1 TO 2000
    m1r
  NEXT z
  FOR z=1 TO 2000
    m1l
  NEXT z
  m1a
  FOR z=1 TO 10000
  NEXT z
GOTO Schleife
REM Programmende
```

Vor dem erneuten Abwärtslauf wartet das Programm eine Zeitlang. Auch dafür benutzen wir wieder eine FOR...NEXT-Schleife. Hier wird nichts anderes gemacht, als von 1 bis 10000 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".
Das Programm wurde mit der GOTO-Anweisung beendet, wodurch der Vorgang von neuem beginnt.
Mit **Start** wird die Seilwinde in Gang gesetzt, mit **Stop** läßt sie sich anhalten.

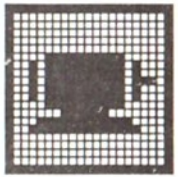
Wie Sie dem Bild auf dem Interface entnehmen können, lassen sich maximal vier Motoren ansteuern. Für jeden Motor sind zwei Leitungen vorgesehen, für Motor 2 z.B. grün und blau. Schließen Sie den Motor an diese Leitungen an, so ist der bisherige Befehl m1r nicht mehr wirksam. Motor 2 wird mit

`m2r` bzw. `m2l`

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

`m3r` bzw. `m3l`
`m4r` bzw. `m4l`

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, wählen Sie den Menüpunkt **Save As**. Geben Sie im Dateifenster z.B. den Namen MOTOR1 an. Anstelle MOTOR1 können Sie natürlich auch einen anderen Namen Ihrer Wahl verwenden. Nun löschen Sie den Programmspeicher und laden wieder die Schlüsselzeilen zum Interfacetreiber durch Anwahl von **Open** und Auswahl der Datei INIT.



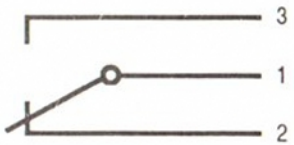


Bild 4.2: Schaltzeichen eines Tasters.

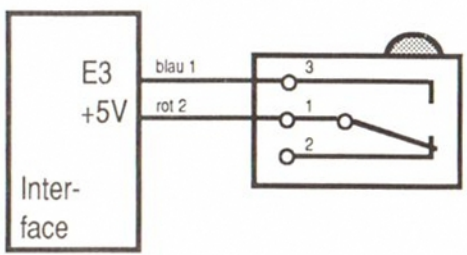


Bild 4.3: Verbindung vom Taster zum Interface.

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:

```
REM Programmanfang
init
Schleife:
  LOCATE 1,1
  PRINT e3
GOTO Schleife
REM Programmende
```

Die Funktion e3 liest den Wert der Eingabeleitung E3 des Interfaces ein. Auf dem Bildschirm wird der Wert des Eingangs E3 mit der PRINT-Anweisung angezeigt. Die Druckposition wird zuvor auf die linke obere Bildschirmecke gesetzt. Dies wird durch die Anweisung LOCATE bewirkt. Die Zahlen nach dem Befehlswort bedeuten Zeilen- und Spaltennummer.

Starten Sie das Programm mit **Start**. Der Bildschirm wird gelöscht und anschließend

der Schaltzustand laufend angezeigt. Zunächst 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 auf den Taster. Jetzt wird eine "1" angezeigt: der Schaltkontakt ist geschlossen, an E3 liegen +5V an.
Man nennt den Kontakt 3 einen "Schließerkontakt", da er beim Betätigen des Tasters schließt.
Die Befehle zum Einlesen des Tasters und zum Ausdrucken des Werts sind mit einer GOTO-Anweisung und der dazugehörigen Sprungmarke eingerahmt. Wir kennen diese Schleife bereits aus dem letzten Kapitel. Sie wird solange wiederholt, bis wir sie mit **Stop** unterbrechen.
Damit Sie auch wissen, was "0" und "1" bedeuten, ergänzen wir das Programm. An die Stelle der PRINT-Anweisung kommen die Zeile:

```
IF e3=0 THEN PRINT  "Taster aus"
     ELSE PRINT "Taster ein"
```

Nebenstehende Anweisung wird als eine einzige Zeile in das List-Fenster eingegeben. Das List-Fenster rutscht dabei nach rechts, sobald die rechte Begrenzung erreicht ist. Im Experimentierhandbuch müssen wir solche überlangen Anweisungen in zwei oder mehr Zeilen drucken.

Nach **Start** wird der entsprechende Text angezeigt. Die Zuordnung erreichen wir mit einer IF...THEN-Abfrage. Dieser Befehl prüft eine Bedingung: wenn (IF) etwas zutrifft (e3=0), dann (THEN) führe aus: PRINT "Taster aus".
In allen anderen Fällen (ELSE) wird der nächste Befehl ausgeführt:
PRINT "Taster ein"
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 **Start**. 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 neu ein:

Wir haben drei Sorten von Programmschleifen kennengelernt:
Mit der Anweisung GOTO und einer entsprechenden Sprungmarke haben wir Endlos-Schleifen programmiert. Sie werden solange durchlaufen, bis das Programm mit ***Stop*** *beendet wird.*
Mit dem Anweisungspaar FOR und NEXT haben wir Schleifen programmiert, die eine bestimmte Anzahl von Durchläufen durchgeführt werden.
Mit dem Anweisungspaar WHILE und WEND haben wir Schleifen programmiert, die solange durchlaufen werden, wie eine angegebene Bedingung erfüllt ist.

```
REM Programmanfang
init
PRINT"Wenn ein Feld erscheint"
PRINT"drücken Sie den Taster!"
PRINT
i=0
FOR z=1 TO 10000
NEXT z
COLOR 0,1
PRINT " "
COLOR 1,0
WHILE e3=1
  i=i+1
WEND
PRINT
PRINT"Stop nach: ";i
REM Programmende
```

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 100 erreichen, dann sind Sie wirklich topfit!
Die Bedeutung der meisten Programmzeilen kennen Sie bereits aus den bisherigen Beispielen. Das Feld wird durch Ausdrukken eines inversen Leerzeichens erzeugt. Dazu muß bei der Bildschirmdarstellung die Schriftfarbe mit der Hintergrundfarbe vertauscht werden. Dies wird durch die Anweisung COLOR 0,1 bewirkt. Sie vertauscht die normale Zuordnung. Danach wird das Leerzeichen gedruckt, das nun sichtbar als helles Rechteck erscheint. Dann wird durch die Anweisung COLOR 1,0 wieder auf die alte Zuordnung zurückgestellt
Neu ist auch die WHILE...WEND-Schleife. Beim Antreffen der WHILE-Anweisung wird die Bedingung der Anweisung geprüft: e3=1. Wenn die Bedingung erfüllt ist, werden alle nachfolgenden Anweisungen bis zu der Anweisung WEND ausgeführt. Danach erfolgt ein Rücksprung zu der WHILE-Anweisung mit erneuter Überprüfung der Bedingung. Ist die Bedingung irgendwann nicht mehr erfüllt, so fährt das Programm mit der Anweisung nach WEND fort. In diesem Fall führt das Drücken der Taste zu dem Wert 0 für e3 und damit zum Verlassen der Schleife.
Anschließend wird der Wert der Variablen i ausgedruckt. Dieser ist nichts anderes als die Anzahl der Schleifendurchläufe durch die WHILE...WEND-Schleife bis der Taster gedrückt wurde. 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. Laden Sie das Programm INIT und geben Sie ein:

```
REM Programmanfang
init
m1r
WHILE e3=0
WEND
m1a
REM Programmende
```

Nach **Start** wird der Motor dauernd laufen. Drücken Sie den rechten Taster, bleibt er stehen. Die Abfrage des Schaltzustandes erfolgt in der WHILE-Anweisung. Solange e3=0 ist, also der Schließerkontakt (1-3) offen ist, springt das Programm immer wieder zur WHILE-Anweisung. 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 **Stop** oder eine Fehlermeldung).
Trifft jedoch die Bedingung e3=1 zu (Taster gedrückt), wird die Schleife verlassen und der Motor ausgeschaltet.
Wir können für die Steuerung auch den anderen Taster benutzen:

```
WHILE e2=0
```

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

```
WHILE e2=0 OR e3=0
```

Bei dieser Abfrage genügt es, daß eine der beiden Bedingung erfüllt ist, um die Schleife zu durchlaufen. Man nennt dies eine logische ODER-Verknüpfung (OR=oder). Damit die Schleife verlassen und der folgende Befehl (m1a) ausgeführt wird, müssen beide Taster gedrückt sein. Prüfen Sie diese Verknüpfung nach Start des Programms mit **Start** nach!
Man kann aber auch eine ähnliche Aufgabe stellen: Der Motor soll solange laufen, bis eine der beiden Tasten (oder beide zugleich) gedrückt werden. Diesmal lautet die Schleifenbedingung, daß e2=0 und e3=0 erfüllt sein müssen, d.h. beide nicht gedrückt sein müssen. Dementsprechend benutzen wir die logische UND-Verknüpfung (AND=und):

```
REM Programmanfang
init
m1r
WHILE e2=0 AND e3=0
WEND
m1a
REM Programmende
```

Testen Sie auch dieses Programm!
Jetzt wollen wir den linken Taster für "Seilwinde aufwärts" und den rechten für "Seilwinde abwärts" einsetzen.

```
REM Programmanfang
init
Schleife:
  IF e3=1 AND e2=0 THEN CALL m1r
  IF e3=0 AND e2=1 THEN CALL m1l
GOTO Schleife
REM Programmende
```

Für die Bewegungsbefehle 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.
Neu ist die Anweisung CALL. Sie bewirkt den Aufruf eines Unterprogramms aus der LIBRARY. Das Anweisungswort kann vor jedem der neuen Befehle wie init, m1r, m1a usw. stehen. Es wird aber in den meisten Fällen nicht benötigt, so daß wir es bislang

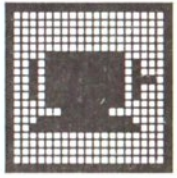
weggelassen haben. Bei der IF-Anweisung wird es jedoch benötigt, da der BASIC-Interpreter sonst den Namen des Unterprogramms mit einer Sprungmarke verwechseln würde. Dies würde als Syntax-Fehler gemeldet werden.
Geben Sie die Zeilen ein und starten das Programm mit **Start**. Sie können das Seil jetzt mit den beiden Tastern beliebig hinauf- und herunterfahren. Anhalten läßt sich der Motor mit **Stop**. In den Bedingungen finden wir wieder die UND-Verknüpfung (AND) von zwei Vergleichen, 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 ändern wir den Nachsatz der IF-Anweisung (die Zeilen nach THEN):

```
IF e3=1 AND e2=0 THEN
     GOSUB rechts
IF e3=0 AND e2=1 THEN
     GOSUB links
```

Die Anweisung GOSUB ruft ein BASIC-Unterprogramm auf (nicht zu verwechseln mit der Anweisung CALL, die ein Unterprogramm der LIBRARY aufruft). BASIC-Unterprogramme werden immer dann verwendet, wenn bestimmte Programmsequenzen mehrmals benötigt werden. Unterprogramme sind auch dann sinnvoll, wenn man sie in Zusammenhang mit bestimmten logischen Programmabschnitten bringt, wie es hier der Fall ist. Das Unterprogramm "rechts" erfüllt die Funktion, das Seil einmal ganz rechts herum abzuwickeln. Das Unterprogramm "links" wickelt das Seil links herum ganz auf.
Dei beiden Unterprogramme müssen ebenfalls noch in den Programmtext eingegeben werden. Ihr Platz ist am Ende der bisherigen Programmliste nach den drei Anweisungen mit dem Schlüsselwort CLOSE und der END-Anweisung:

```
rechts:
  FOR z=1 TO 2000
    mlr
  NEXT z
  mla
RETURN

links:
  FOR z=1 TO 2000
    mll
  NEXT z
  mla
RETURN
```



BASIC-Unterprogramme werden jeweils mit der Anweisung RETURN beendet. Diese Anweisung veranlaßt den BASIC-Interpreter, den Faden jeweils wieder mit der Anweisung aufzunehmen, die dem Unterprogrammaufruf folgt.
Wenn das Seil vollständig aufgewickelt ist, starten Sie das Programm mit **Start**. Durch kurzes Drücken des rechten Tasters läuft das Seil nach unten. Dazu dient das Unterprogramm in dem Nachsatz der ersten IF-Anweisung.
Zurückziehen läßt sich das Seil wieder durch kurzes Drücken des linken Tasters. Die Programmschritte hierzu finden wir in dem Unterprogramm des Nachsatzes der zweiten IF-Anweisung.
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 **Stop** an und ergänzen Sie das Programm:

```
REM Programmanfang
init
position=0
Schleife:
  IF e3=1 AND e2=0 AND position=0
      THEN GOSUB rechts
IF e3=0 AND e2=1 AND position=1
      THEN GOSUB links
GOTO Schleife
REM Programmende
      :
      :
rechts:
  FOR z=1 TO 2000
    m1r
  NEXT z
  m1a
  position=1
RETURN

links:
  FOR z=1 to 2000
    m1l
  NEXT z
  m1a
  position=0
RETURN
```

Am Anfang muß das Seil oben sein; deswegen wird die Variable position zu Programmstart, noch vor Beginn der Endlos-

*Genaugenommen hätte die Variable position im Anfangsteil des Programms nicht auf Null gesetzt werden müssen, denn der BASIC-Interpreter setzt zu Programmstart mit **Start** alle Variablen auf Null. So wie BASIC verfahren jedoch nicht alle Programmiersprachen und es gehört daher zum "sauberen" Programmieren, allen Variablen einen wohldefinierten Anfangswert zu verleihen.*

schleife, auf Null gesetzt. Die IF-Abfragen haben wir jeweils um eine zusätzliche Bedingung erweitert. So kann das Programm nur das Seil abwickeln, wenn Taster 3 gedrückt ist, Taster 2 frei ist und das Seil oben ist (position=0). In der zweiten IF-Anweisung ist es genau umgekehrt: Taster 3 frei, Taster 2 gedrückt und Seil unten (position=1). Dann wird das Seil hochgezogen. Die jeweilige Position halten wir in der Variablen position fest, nachdem die entsprechende Bewegung durchgeführt wurde. Beenden läßt sich das Programm wieder mit **Stop**.

Sie sehen, wie man mit den Tastern gezielt den Motor steuern kann - entweder durch direkte Laufbefehle (m1r, m1l) 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. 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. Laden Sie dazu den Interfacetreiber und geben Sie ein:

```
REM Programmanfang
```

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.

```
init
m11
WHILE e2=0
WEND
m1a
REM Programmende
```

Stellen Sie die Seiltrommel so, daß der Taster nicht gedrückt ist. Nach **Start** bewegt sich der Motor, bis der Taster schaltet. Danach bleibt er stehen und hat dabei eine halbe Umdrehung hinter sich gebracht. Die Programmschritte 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 Menüpunkt **Start** den nächsten Schritt aufrufen. Stellen Sie jetzt aber die Seiltrommel mal so, daß der Nocken den Taster drückt und rufen Sie **Start** auf. 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 der WHILE-Anweisung gleich beim ersten Mal erfüllt und das Programm wurde sofort beendet. Erweitern Sie das Programm, so daß es wie folgt aussieht:

```
REM Programmanfang
init
m11
WHILE e2=1
WEND
WHILE e2=0
WEND
m1a
REM Programmende
```

Nun wird zunächst gewartet, bis der Taster freigegeben ist. Danach kann 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 Unterprogramm in der Library:

```
m1v     (Motor 1 vorwärts)
```

Das Unterprogramm arbeitet zudem mit einer sogenannten Gegenstrombremse: kurz vor dem Befehl m1a wird der Motorstrom nochmals umgepolt. Damit bleibt der Motor schlagartig stehen. Die Positionierung wird also mit diesem Aufruf 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). Das Programm dazu sieht so aus:

```
REM Programmanfang
init
FOR z=1 TO 20
  m1v
NEXT z
REM Programmende
```

Ebenso läßt er sich in die andere Richtung drehen; ändern Sie das Kommando zum Vorwärtsdrehen in:

`m1z` (Motor 1 zurück)

Nach **Start** läuft der Motor zehn Umdrehungen zurück.
Neben den beiden Kommandos m1v und m1z gibt es diese Kommandos natürlich auch noch für die Motoren 2, 3 und 4. Also:

`m2v` und `m2z`
`m3v` und `m3z`
`m4v` und `m4z`

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

```
REM Programmanfang
init
position=0
Schleife:
  WHILE INKEY$=""
  WEND
  IF position=0 THEN GOSUB
     unten ELSE GOSUB oben
GOTO Schleife
REM Programmende

unten:
  FOR z=1 TO 20
    m1z
  NEXT z
  position=1
RETURN

oben:
  FOR z=1 TO 20
    m1v
  NEXT z
  position=0
RETURN
```

Das Seil auf der Winde befindet sich oben;

starten Sie das Programm mit **Start**. Der Motor bewegt sich noch nicht. Sie müssen jetzt eine Taste drücken, damit das Seil nach unten läuft. Die Abfrage der Tastatureingabe erfolgt in einer WHILE...WEND-Schleife, deren eigentlicher Schleifenkörper leer ist. Die INKEY$-Funktion liest den Code der gedrückten Taste ein.
Wird keine Taste gedrückt, ist INKEY$ leer (codiert durch die zwei Anführungszeichen, zwischen denen sich nichts befindet). Sobald eine Taste gedrückt wird, wird die WHILE...WEND-Schleife verlassen.
Das Seil läuft nach unten. Die Anfangsposition war oben (position=0); damit führt das Programm den ersten Teil der IF-Anweisung aus. 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 position =1 ist. Das Seil ist dabei im Uhrzeigersinn auf die Trommel gewickelt. Mit **Stop** wird das Programm beendet.
Wir können das Seil jetzt auch auf halber Strecke anhalten, indem wir in den FOR...NEXT-Schleifen in den Unterprogrammen nur zehn Drehschritte vorgeben:

```
FOR z=1 TO 10
```

Nach **Start** 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:

```
REM Programmanfang
init
position=0
INPUT "Schritte (1-20):";s
Schleife:
  WHILE INKEY$=""
  WEND
  IF position=0 THEN GOSUB
     unten ELSE GOSUB oben
GOTO Schleife
REM Programmende

unten:
  FOR z=1 TO s
    mlz
  NEXT z
  position=1
RETURN

oben:
  FOR z=1 TO s
    mlv
  NEXT z
  position=0
```

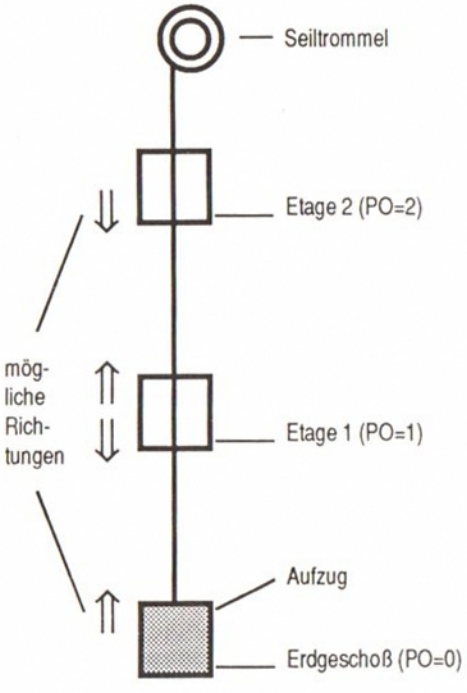


Bild 4.4: Arbeitsweise eines Aufzugs.

```
RETURN
```

Nach **Start** geben Sie die Schrittzahl ein; sie wird in s gespeichert. Nach einem Tastendruck läuft das Seil diese Anzahl Schritte vor und auch wieder zurück, denn der Endwert der FOR...NEXT-Schleifen ist der Wert von s.
Fahren Sie das Seil nun um 20 Schritte nach unten und beenden das Programm mit **Stop**. 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, laden das Programm INIT und geben ein:

```
REM Programmanfang
init
position=0
auf$=CHR$(28)
ab$=CHR$(29)
Schleife:
  LOCATE 1,1
  PRINT "Etage:";position
  a$=INKEY$
  IF a$=auf$ AND position<>2
     THEN GOSUB aufwaerts
  IF a$=ab$ AND position<>0
     THEN GOSUB abwaerts
GOTO Schleife
REM Programmende

aufwaerts:
  FOR z=1 TO 10
    m1v
  NEXT z
  position=position+1
RETURN

abwaerts:
  FOR z=1 TO 10
    m1z
  NEXT z
  position=position-1
RETURN
```

Diesmal ist das Seil zu Beginn ganz ausgefahren, der Lift steht in Grundstellung ganz unten (position=0).
Starten Sie das Programm mit **Start**. Auf dem Bildschirm erscheint die Anzeige, auf welcher Etage der Aufzug sich befindet. Geben Sie die Richtung des Fahrstuhls ein: z.B. ↑, wenn Sie nach oben möchten. Der

Lift fährt hoch nach Etage 1 und der Bildschirm 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 die Auswahl des entsprechenden Unterprogramms. Die Unterprogramme sind wieder nach der END-Anweisung angehängt. Die Etagenposition ist in den IF-Abfragen mit dem Cursortastencode UND-verknüpft. Die Codes für die Cursortasten sind am Programmanfang in den Variablen auf$ und ab$ 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 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 der Programmliste angegeben.
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 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. Laden Sie das Programm INIT und geben Sie folgendes Programm ein:

```
REM Programmanfang
init
z=0
Schleife:
  INPUT "Position (0-20):";s
  IF z<s THEN GOSUB aufwaerts
  IF z>s THEN GOSUB abwaerts
```

```
  z=s
GOTO Schleife
REM Programmende

aufwaerts:
  FOR i=z+1 TO s
    m1z
  NEXT i
RETURN

abwaerts:
  FOR i=z-1 TO s STEP -1
    m1v
  NEXT i
RETURN
```

Die Anfangsposition 0 wird zu Beginn mit z=0 markiert. Nach Programmstart mit **Start** geben Sie die gewünschte Position ein: eine Zahl zwischen 0 und 20. Das Programm prüft durch Vergleich, ob die Zielposition (Sollwert s) größer oder kleiner ist als die Startposition (Istwert z). Ist z<s, läuft der Motor die entsprechende Schrittzahl vor. Stören Sie sich nicht daran, daß hier m1z steht: das Seil ist im Uhrzeigersinn aufgewickelt und läuft nach unten. Ist die Sollposition kleiner als der Istwert (z>s), läuft der Motor in die andere Richtung zurück. Die zweite FOR...NEXT-Schleife zählt rückwärts (...STEP -1), also vom höheren Wert z zum kleineren Wert s in 1er-Schritten. Danach merkt sich das Programm die neue Position als Istwert in der Variablen z und wiederholt den ganzen Vorgang mittels der GOTO-Anweisung.
Zur Übung finden Sie auf der Diskette zwei Programme: mit Programm SCHRITT können Sie einen Motor schrittweise vor- und zurückdrehen, mit POSITION läßt er sich auf bestimmte Positionen bewegen. Der Ablauf wird jeweils auf dem Bildschirm dargestellt.

5. Schalten mit Licht
5.1. Berührungslos schalten: Gabellichtschranke

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, das zu dem Programm INIT hinzugefügt wird, wird die Funktion des Aufbaus geprüft:

```
REM Programmanfang
init
FOR i=1 TO 1000
  m1r
  m3r
NEXT i
m1a
m3a
REM Programmende
```

Nach dem Start des Programms mit **Start** 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:

```
REM Programmanfang
init
Schleife:
  PRINT e2
GOTO Schleife
REM Programmende
```

Am Bildschirm erscheint die Anzeige "0". Wenn nicht, ist vielleicht die Raumhelligkeit

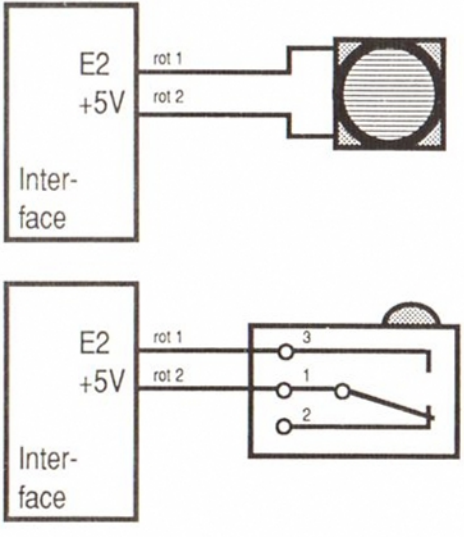


Bild 5.1: Ein Fotowiderstand ersetzt den Taster aus Bild 4.3.

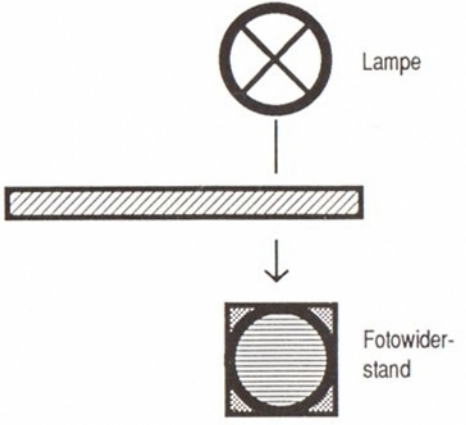


Bild 5.2: Prinzip einer Gabellichtschranke

zu groß. Achten Sie darauf, daß kein direktes Licht auf den Fotowiderstand fällt.
Im nächsten Schritt schalten wir die Lampe des Modells wieder ein:

```
REM Programmanfang
init
m3r
Schleife:
  PRINT e2
GOTO Schleife
REM Programmende
```

Nach **Start** 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 m3r 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.

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.

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 **Start** muß das Programm "0" anzeigen. Nach Programmabbruch mit **Stop** erweitern wir das Programm mit der Motorsteuerung:

```
REM Programmanfang
init
m3r
m1r
WHILE e2=0
WEND
m1a
m3a
REM Programmende
```

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

```
m1z
```

Löschen Sie das bisherige Hauptprogramm oder laden Sie das Programm INIT neu und geben Sie folgendes neues Programm ein:

```
REM Programanfang
init
FOR z=0 TO 200
  m3r
NEXT z
m1z
m3a
REM Programmende
```

Starten Sie das Programm mit **Start**. 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 den Befehl m1z sechsmal wiederholen und erzielen so eine ganze Umdrehung:

```
FOR i=1 to 6
  m1z
NEXT i
```

Mit der Programmänderung:

```
  m1v
```

dreht sich das Rad in die andere Richtung. Auf der Diskette finden Sie das Programm WINKEL, das die Drehscheibe in ähnlicher Weise steuert, wie das Programm POSITION 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°.

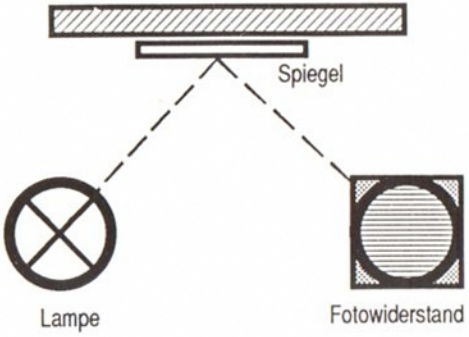


Bild 5.3: Arbeitsweise einer Reflexionslichtschranke

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 durch **Start.**
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 Testprogramm. Speichern Sie das bisherige Programm und geben Sie als Hauptprogramm jetzt ein:

```
REM Programmanfang
init
m3r
```

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.

```
Schleife:
  LOCATE 1,1
  PRINT e2
GOTO Schleife
REM Programmende
```

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 **Start**. Schieben Sie das Rad auf der Achse so weit nach oben oder unten, bis die Bildschirmanzeige "1" zeigt. 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 **Stop** halten Sie das Testprogramm an. Laden Sie Sie das vorige Programm durch Anklicken des Menüpunktes **Open** und Auswahl der Datei. Nun können Sie das Programm erneut mit **Start** starten.
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:

```
REM Programmanfang
init
FOR z=1 TO 200
  m3r
NEXT z
FOR i=1 TO 3
  m1v
NEXT i
Schleife:
  FOR i=1 TO 6
    m1z
  NEXT i
  FOR i=1 TO 6
    m1v
  NEXT i
GOTO Schleife
REM Programmende
```

Markieren Sie einen Punkt auf dem Rad und starten das Programm mit **Start**. 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.

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 das Programm INIT (s. Kapitel 3). Zum Einlesen eines Analogwertes - hier also eines Widerstandswertes - mittels des Eingangs EX benutzen Sie die Funktion ex. Sie können den Funktionswert in Bedingungen verwenden, einer anderen Variablen zuweisen oder ausdrucken, genau wie der Wert der Funktionen e1 bis e8. Daß der Fotowiderstand auf Lichtänderungen reagiert, können wir mit folgenden Programm feststellen:

```
REM Programmanfang
init
Schleife
  PRINT "Meßwert:";
  PRINT ex
  WHILE INKEY$=""
  WEND
GOTO Schleife
REM Programmende
```

Geben Sie die Zeilen ein und starten das Programm mit **Start**. Schwenken Sie nun den Fotowiderstand hin und her, so daß er unterschiedlich beleuchtet wird. Drücken Sie dabei immer wieder auf eine Taste des Computers; dann verläßt das Programm die WHILE...WEND-Schleife und der nächste Meßwert wird auf dem Bildschirm angezeigt. Man erkennt deutlich, daß bei jedem

Meßwert:	Licht
58	hell
87	
123	
174	
255	dunkel

Bild 6.1: Meßreihe einer Lichtmessung.

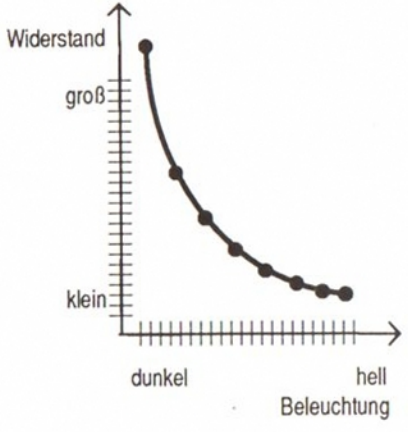


Bild 6.2: Allgemeiner Kennlinienverlauf des Fotowiderstandes.

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 gegebenenfalls wieder mit **Start**. 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 **Stop**. 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 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:

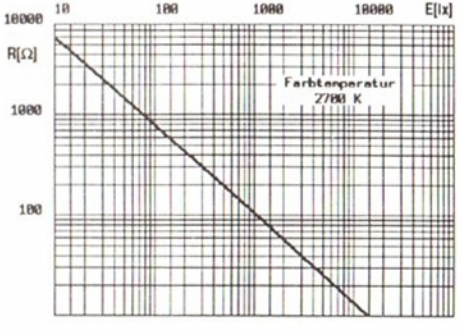


Bild 6.3: Kennlinie des fischertechnik Fotowiderstandes aus einem Datenbuch.

```
REM Programmanfang
init
WHILE INKEY$=""
WEND
letztes=ex
hm=letztes
Schleife:
  hell=ex
  IF hell<>letztes THEN
          GOSUB Anzeige
  letztes=hell
GOTO Schleife
REM Programmende

Anzeige:
  s=80*hm/ex
  CLS
  COLOR 0,1
  FOR i=1 TO s
    PRINT " ";
  NEXT i
  COLOR 1,0
RETURN
```

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 **Start**. Am Anfang des Programms finden wir das Programmstück zur Messung der hellsten Stelle im Raum. Wir starten den Vorgang durch Tastendruck (WHILE...WEND-Schleife).

Weiter geht's mit der Messung der Lichtstärke, die in den Variablen letztes und hm abgespeichert wird. Den Wert von hm benötigen wir später bei jeder Anzeige als Bezugsgröße.

Mit der Sprungmarke "Schleife" beginnt das Programmstück, das nun immer wiederholt wird. Wir messen die Helligkeit aus der Richtung, in die der Fotowiderstand gerade zeigt, und vergleichen diesen Wert mit dem vorigen mittels der IF-Anweisung. Nur wenn der Meßwert sich von dem vorigen unterscheidet, muß etwas angezeigt werden. Dies übernimmt das Unterprogramm Anzeige. Zunächst wird die Balkenlänge errechnet. Bei halber Lichtstärke ist hell ca. doppelt so groß wie hm. Daraus ergibt sich

s = 80*1/2 =40

Der Balken hat die halbe Länge wie vorher. In den nächsten Zeilen wird der Bildschirm gelöscht und darauf s Zeichen (FOR i=1 TO s) hintereinander angezeigt. Dieser Balken besteht aus inversen Leerzeichen. Die

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.

COLOR-Anweisung vor der FOR-Anweisung bewirkt dies, indem die Farbzuordnung für Vordergrund und Hintergrund vertauscht werden. Nach der Anzeige wird wieder auf normale Farbdarstellung zurückgeschaltet. Dies erfolgt durch eine ganz ähnliche COLOR-Anweisung. Mit der RETURN-Anweisung endet das Unterprogramm. Im Hauptprogramm wird der jeweils letzte Meßwert der Variablen letztes zugewiesen.
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 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:

```
REM Programmanfang
init
FOR i=1 TO 10
  m1v
NEXT i
REM Programmende
```

Nach **Start** 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 m1v dreht das Schneckenrad um ½ Umdrehung. 1 Umdrehung des Schneckenrades 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:

```
FOR i=1 TO 40
```

und **Start**. Denken Sie an das Kabel zum Fotowiderstand! Im Notfall halten Sie das Programm mit **Stop** an. Zurückdrehen läßt sich das Computerauge, indem der Befehl m1v durch den Befehl m1z ausgetauscht wird.

In eine bestimmte Position drehen läßt sich der Lichtsensor mit:

```
REM Programmanfang
init
re$=CHR$(30)
li$=CHR$(31)
Schleife:
  a$=INKEY$
  IF a$=re$ THEN CALL mlz
  IF a$=li$ THEN CALL mlv
GOTO Schleife
REM Programmende
```

In den ersten Zeilen werden den Variablen re$ und li$ die Tastaturcodes für Cursor nach rechts und Cursor nach links zugewiesen.
Starten Sie das Programm mit **Start**. Jetzt können Sie die Cursortaste ← oder → betätigen. In den beiden IF-Anweisungen wird die gedrückte Taste erkannt und ein entsprechender Drehschritt ausgeführt. Das Programm läuft solange in der Schleife, bis Sie **Stop** aufrufen.
Der Fotowiderstand ist nach wie vor am Analogeingang EX angeschlossen. Abgefragt wird er mit der Funktion ex. Der Wert der Funktion 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 folgendermaßen:

```
REM Programmanfang
init
re$=CHR$(30)
li$=CHR$(31)
Schleife:
  a$=INKEY$
  IF a$="" THEN GOTO Schleife
  IF a$=re$ THEN CALL mlz
  IF a$=li$ THEN CALL mlv
  CLS
  PRINT ex
GOTO Schleife
REM Programmende
```

Starten Sie es mit **Start**. 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 Aufruf von **Stop** ein:

```
REM Programmanfang
init
richtung=0
Schleife:
  WHILE INKEY$=""
```

```
  WEND
  IF richtung=0 THEN GOSUB
       links ELSE GOSUB rechts
  PRINT
GOTO Schleife
REM Programmende

links:
  FOR i=1 TO 10
    FOR j=1 TO 4
      m1v
      PRINT ex,
    NEXT j
  PRINT
  NEXT i
  richtung=1
RETURN

rechts:
  FOR I=1 TO 10
    FOR j=1 TO 4
      m1z
      PRINT ex,
    NEXT j
  PRINT
  NEXT i
  richtung=0
RETURN
```

Nach Programmstart mit **Start** wartet das Programm auf eine Tastenbetätigung. Danach geht das Programm weiter bis zur IF-Anweisung, die entscheidet, 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 richtung. Sie wird am Anfang auf 0 gesetzt. Damit verläuft die erste Drehung nach rechts (erster Nachsatz der IF-Anweisung). In dem Unterprogramm "rechts" erhält der Merker richtung den Wert 1. Bei dem nächsten Durchlauf springt das Programm deshalb in den zweiten Nachsatz der IF-Anweisung (nach ELSE). Der Fotowiderstand wird durch das Unterprogramm "links" zurückgedreht. Nun folgt wieder eine Rechtsdrehung, da richtung auf 0 gesetzt wurde.

Bei der Drehung werden die Meßwerte in vier Spalten 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, 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, das eine grafische Darstellung der o.a. Lichtmessung erzeugt. Es benutzt das Malinstrument; und wie jenes funktioniert, erfahren wir im nächsten Kapitel.

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.

6.3. Darstellung von Meßwerten: Computergrafik

Neben der normalen Textausgabe kann unser Computer auch die Bildpunkte (Pixel) des Bildschirms einzeln ansteuern. Dabei gibt es verschiedene Betriebsarten, die sich in der Anzahl der Bildpunkte und deren Farbmöglichkeiten unterscheiden. In der Betriebsart, die mit der Anweisung

```
SCREEN 2,640,256,2,2
```

im Vorspann des Programms INIT gewählt wurde, wird der Bildschirm in 640 x 256 Bildpunkte aufgeteilt (sog. medium resolution), von denen jeder einzeln angesteuert und in einer von vier Farben dargestellt werden kann. Damit lassen sich sehr gute Bilder und Grafiken erstellen. Das Malinstrument wird durch das Kommando ge aktiviert. Zusätzlich muß dem Kommando aber auch noch ein Anknüpfungspunkt zu der Bildschirmverwaltung mitgegeben werden. Die Funktion WINDOW(7) liefert einen Zeiger auf den sog. INTUITION WINDOW Datensatz. Funktionsaufruf und Kommando können in einer Zeile zusammengefaßt werden:

```
ge(WINDOW(7))
```

Zusammenfassung aller Grafikbefehle:

ge(WINDOW(7))
Grafik einschalten / löschen
ga
Grafik ausschalten, zurück zum Textschirm
gv(s)
Grafik-Schildkröte um s Schritte vor
gz(s)
Grafik-Schildkröte um s Schritte zurück
gr(d)
Grafik-Schildkröte um d Grad rechts schwenken
gl(d)
Grafik-Schildkröte um d Grad links schwenken
g1
Grafikstift einschalten

Mit der Anweisung wird der Bildschirm gelöscht und es 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.
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.

```
REM Programmanfang
init
ge(WINDOW(7))
gv(20)
```

Bevor wir das Programm starten, wollen wir uns überlegen, wie wir uns die Grafik anschauen können, bevor mit dem Programmende der Bildschirm gleich wieder gelöscht wird. Fügen Sie folgende Zeilen dem Programm hinzu:

```
LOCATE 27,1
PRINT "Beliebige Taste ->
     Programmende"
WHILE INKEY$=""
WEND
ga
REM Programmende
```

Es wurde eine Abfrageschleife der Tastatur programmiert, die auf einen Tastendruck wartet. Das Malinstrument wird mit der Anweisung ga wieder ausgeschaltet.
Nach **Start** bewegt sich die Schildkröte nach oben und hinterläßt auf dem Bildschirm eine Linie. Dies wird durch die Anweisung gv(20) bewirkt. Die Schildkröte geht 20 Schritte vor und zeichnet dabei eine Linie. In der nächsten Erweiterung des Programms, die vor der Tastaturabfrage eingeschoben wird, wird der Grafikstift ausgeschaltet - der Bleistift vom Papier angehoben.

```
g0
gv(10)
```

Beachten Sie, daß das erste Kommando mit der Ziffer Null und nicht mit dem Buchstaben O endet.
Wenn Sie das Programm mit **Start** starten,

g0
Grafikstift ausschalten
gc
Grafik kopieren (Hintergrundgrafik wird angezeigt)
gk
aktueller Kurs der Grafik-Schildkröte nach gk speichern
gx
X-Koordinate der Grafik-Schildkröte nach gx speichern
gy
Y-Koordinate der Grafik-Schildkröte nach gy speichern
gload(SADD(f$))
Grafikbild aus Datei, deren Name in f$ angegeben ist, laden
gsave(SADD(f$))
aktuelles Grafikbild auf Diskette unter dem Dateinamen, der in f$ angegeben ist, speichern

wird zunächst wieder der 20 Schritte 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. Auch dieser wird wieder vor der Tastaturabfrage eingefügt:

```
gl
gr(90)
gv(20)
```

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.

```
gl(45)
gz(30)
```

Starten Sie das Programm mit **Start**. Die Grafik-Schildkröte zeichnet nun zusätzlich eine diagonale Linie von 30 Schritten.

Sie können auch ein Viereck zeichnen. Geben Sie folgendes neues Programm ein:

```
REM Programmanfang
init
ge(WINDOW(7))
FOR i=1 TO 4
  gv(30)
  gr(90)
NEXT i
LOCATE 27,1
PRINT "Beliebige Taste ->
     Programmende"
WHILE INKEY$=""
WEND
ga
REM Programmende
```

Nach **Start** zeichnet die Grafik-Schildkröte ein Viereck auf den Bildschirm. Wir können auch ein größeres malen:

```
  gv(40)
```

Nach Programmstart mit **Start** erscheint jetzt ein größeres Quadrat auf dem Schirm. Erweitern Sie das Programm:

```
REM Programmanfang
init
ge(WINDOW(7))
```

```
FOR k=1 TO 9
  FOR i=1 TO 4
    gv(30)
    gr(90)
  NEXT i
  gr(40)
NEXT k
LOCATE 27,1
PRINT "Beliebige Taste ->
     Programmende"
WHILE INKEY$=""
WEND
ga
REM Programmende
```

Nach Programmstart mit **Start** werden mehrere Vierecke auf den Bildschirm gezeichnet, die jeweils um 40° 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 gsave zusammen mit einem Dateinamen in einer Textvariablen oder Textkonstanten verwenden, z.B.:

```
bild$="STERN.PIC"+CHR$(0)
gsave(SADD(Bild$))
```

Fügen Sie obige Zeile vor der Tastaturabfrage ein. 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).
Wenn Sie das abgespeicherte Bild wieder einladen möchten, geben Sie das Kommando gload zusammen mit dem Dateinamen in gleicher Weise ein.
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. Hierzu wird das Programm weiterhin vor der Tastaturabfrage ausgebaut:

```
CLS
FOR i=1 TO 10000
NEXT i
gload(SADD(bild$))
```

Das geladene Bild steht in einer anderen Bildebene als das gezeichnete Bild, dem sogenannten Hintergrund. Es kann durch keine normalen Zeichenbefehle des Malinstruments oder des BASIC-Interpreters zerstört oder gelöscht werden. Weitere Zeichnungen mit der Grafik-Schildkröte werden nun über das Hintergrundbild ge-

Der Zugriff auf eine Datei erfordert den Umweg wie in dem Programmbeispiel: Zuerst wird in eine Textvariable der Dateiname eingegeben. Dann wird ein Nullzeichen (nicht die Ziffer 0!) mit Hilfe der CHR$-Funktion angehängt.
Derartige Textvariablen lassen sich aber nicht so einfach in ein Bibliotheks-Unterprogramm übergeben. Deshalb wird mit der Funktion SADD (String-Address) ein Verweis auf den Speicherplatz der Textvariablen ermittelt. Dieser kann letztlich im Aufruf verwendet werden.

zeichnet, als würde man auf eine Transparentfolie malen. In der Tat ist die Farbe 0 jetzt als transparent definiert. Die Transparentfolie läßt sich mit dem Kommando

```
gc
```

sauber wischen. Nach Ausführung des Kommandos gc liegt also wieder ein sauberes Hintergrundbild vor. 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 gc löscht alle darüberliegenden Zeichnungen und Sie können auf dem frischen Radarschirm die nächsten Meßdaten aufzeichnen.
Auch die Farben des Zeichenstiftes lassen sich (wie die Textausgabe) mit der COLOR-Anweisung verändern. Sie können vier verschiedene Farben für den Zeichenstift und den Hintergrund auswählen. Dabei gilt folgende Zuordnung:

0 : Blau (bzw. Transparent, wenn Hintergrundbild geladen)
1 : Weiß
2 : Schwarz
3 : Orange

Die Zuordnung der Farben zu den Farbnummern läßt sich durch die PALETTE-Anweisung ändern. Schlagen Sie dazu in dem Handbuch zum BASIC-Interpreter nach.
In dem nachfolgenden Beispiel zeichnen wir wieder ein Quadrat, diesmal aber jede Seite in einer anderen Farbe:

```
REM Programmanfang
init
ge(WINDOW(7))
 FOR i=1 TO 4
  COLOR i-1
  gv(30)
  gr(90)
NEXT i
LOCATE 27,1
PRINT "Beliebige Taste ->
     Programmende"
WHILE INKEY$=""
WEND
ga
REM Programmende
```

Passen Sie auf, daß Sie der Schildkröte nicht die gleiche Farbe wie dem Hintergrund zuweisen, wie das Beispiel aufzeigt.

Mit der Grafik-Schildkröte erzeugte Bilder können Sie auch auf Ihren Drucker bringen. Die Fachleute sagen dazu Hardcopy. Um eine Hardcopy auszugeben, benutzen Sie das entsprechende Programm der Amiga-EXTRAS-Diskette.

Dann wären die Spuren der Schildkröte unsichtbar, obwohl der Zeichenstift eingeschaltet ist!
Man kann auch die aktuelle Position der Grafik-Schildkröte mit folgenden Funktionsaufrufen abfragen:

```
gk
gx
gy
```

Dabei liefert gk den momentanen Kurs der Grafik-Schildkröte in Winkelgraden zur Startrichtung, gx die X-Koordinate und gy die Y-Koordinate der Grafik-Schildkröten-Position. Probieren Sie diese Funktionen aus:

```
REM Programmanfang
init
ge(WINDOW(7))
gv(30)
gr(90)
gv(10)
PRINT "Kurs:";gk
PRINT "X   :";gx;"   Y   :";gy
LOCATE 27,1
PRINT "Beliebige Taste ->
    Programmende"
WHILE INKEY$=""
WEND
ga
REM Programmende
```

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:

Auf dem Bildschirm wird der Kurs mit 90 (°) und die Position mit X=20, Y=30 ausgewiesen.Woher kommt die Abweichung von der erwarteten Koordinatenangabe X=10, Y=20? Um die Zeichnungen der Grafik-Schildkröte in etwa verzerrungsfrei erscheinen zu lassen, also dem Quadrat aus dem vorangegangen Beispiel gleich lange Seiten in waagrechter und senkrechter Richtung zuzuweisen, läuft die Schildkröte in senkrechter Richtung pro Schritt immer ein Bildpunkt weit, in waagrechter Richtung jedoch zwei Bildpunkte weit. Damit ergibt sich ein Bewegungsfeld von 320 x 256 Bildpunkten für die Schildkröte, was einigermaßen den Seitenverhältnissen des Bildschirms entspricht. Die Rückgabe der Funktion gx erfolgt jedoch in Bildpunkten, also doppelt soviel wie die Schrittzahl der Schildkröte.

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:

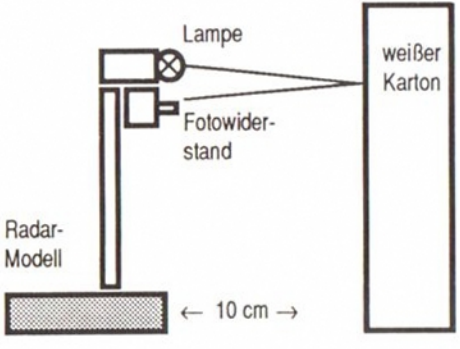


Bild 6.4: Abstandsmessung mit einem Radarmodell.

```
REM Programmanfang
init
m3r
Schleife:
  PRINT ex
  WHILE INKEY$=""
    m3r
  WEND
GOTO Schleife
REM Programmende
```

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 (s. Kapitel 5). Wenn Sie die WHILE-Schleife am Ende des Programms entfernen würden, könnten Sie diesen Effekt genau beobachten. Durch die Trägheit der Lampe ist die Anzeige erst nach 10 bis 15 Schleifendurchläufen stabil. Den Wert, den das Programm ab dem zweiten Tastendruck anzeigt, merken wir 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

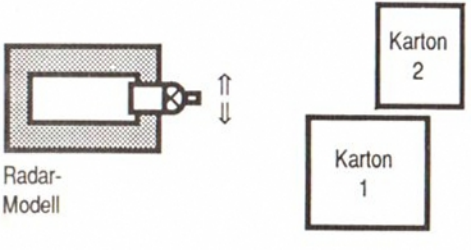


Bild 6.5: Radarabtastung mehrerer Gegenstände.

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:

```
REM Programmanfang
init
ge(WINDOW(7))
FOR i=1 TO 200
  m3r
NEXT i
FOR i=1 TO 40
  hell=ex/3
  g0
  gv(hell)
  gl
  gr(90)
  gv(5)
  gz(5)
  g0
  gl(90)
  gz(hell)
  mlv
  gl(9)
NEXT i
m3a
FOR i=1 TO 40
  mlz
NEXT i
LOCATE 27,1
PRINT "Beliebige Taste ->
     Programmende"
WHILE INKEY$=""
WEND
ga
REM Programmende
```

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.

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, das Sie von der Diskette laden können.

7 Messen und Regeln

7.1 Temperaturen messen: Thermometer

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 das Programm INIT ein:

```
REM Programmanfang
init
Schleife:
  LOCATE 1,1
  PRINT USING"###";ey
GOTO Schleife
REM Programmende
```

Die erste Zeile in der Wiederholschleife setzt die Ausdruckposition in die linke obere Bildschirmecke, so daß der Ausdruck immer wieder an der gleichen Stelle erfolgt. Die Funktion ey liest den Wert am Analogeingang EY - also den Widerstandswert unseres Heißleiters - ein. In der PRINT-Anweisung 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.

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.

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.
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.
Der Zahlenwert ändert sich, wie Sie sehen, normalerweise kaum. Klar, unsere Raumtemperatur ist ja konstant. Fassen Sie nun jedoch 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 **Stop**.
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 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 auslö-

Temperatur (Grad Celsius)	Wert EY
0	148
5	131
10	116
15	102
20	90
25	79
30	70
35	62
40	55
45	48
50	43

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

sen. 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 **Start** 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 **Stop**. 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 ist von Interface zu Interface wegen der Wertestreuung der elektronischen Bauelemente verschieden.

Nehmen Sie den NTC-Widerstand wieder aus dem Gefäß und lassen ihn auf Zimmertemperatur abkühlen. Nach erneutem Programmstart wird ein Zahlenwert angezeigt, den der Computer in den richtigen Temperaturwert umrechnen soll. Dazu benutzen wir die zuvor ermittelte Meßwerttabelle. Ein Anzeigewert von 79 entspricht 25 Grad Celsius, ein Anzeigewert von 55 entspricht 40 Grad Celsius, und ein Anzeigewert von 148 entspricht 0 Grad Celsius. Wie wir sehen, sind die beiden Zahlenreihen gegenläufig, also zum höchsten 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. Fügen Sie als erste Zeile nach der Anweisung "REM Programmanfang" ein:

```
DEF FNt(x)=INT(200-40*LOG(x))
```

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.u.); 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.

Die PRINT-Anweisung wird ebenfalls geändert:

```
PRINT "Temperatur= "; FNt(ey);
      "Grad Celsius"
```

und starten Sie das Programm mit **Start**. In der ersten eingefügten Zeile wird eine Umrechnungsformel von Analogwerten in °C definiert (DEF FNt...). Die Funktion LOG wird darin benutzt; sie ist der natürliche Logarithmus. In der PRINT-Anweisung 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 Interface zu Interface sollten Sie sich Ihre ganz individuelle Umrechnungsfunktion ermitteln. Verwenden Sie das Programm KALIBR 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)=INT(200-40*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:

```
REM Programmanfang
DEF FNt(x)=INT(200-40*LOG(x))
init
ge(WINDOW(7))
GOSUB neuesBild
```

```
Schleife:
  LOCATE 1,1
  PRINT "Temperatur= ";FNt(ey);
          "Grad Celsius"
  GOSUB Anzeige
GOTO Schleife
REM Programmende
```

Das Unterprogramm "neuesBild" erfüllt die Funktion "Bildschirm löschen", das Unterprogramm "Anzeige" die Funktion "Meßwert zeichnen". Hier nun die beiden Unterprogramme:

```
Anzeige:
  IF gx=318 THEN GOSUB
          neuesBild
  schritte=FNt(ey)
  gv(schritte)
  gl
  gv(1)
  g0
  gz(schritte+1)
  gr(90)
  gv(1)
  gl(90)
RETURN

neuesBild:
  gc
  g0
  gz(82)
  gl(90)
  gv(160)
  gr(90)
RETURN
```

Das Unterprogramm "Anzeige" 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 Schritte 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 der Bildschirm gelöscht werden. Hierzu dient das Unterprogramm "neuesBild". Nach dem Löschen der Grafik wird die Schildkröte in die linke untere Ecke des Bildschirms rangiert. Übrigens: die Zahlenwerte 82 und 160 betreffen die Bildschirmabmessungen. Sie stellen die Fahrstrecken der Grafik-

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.

Schildkröte dar, um von der Bildmitte an den linken Bildschirmrand, fast an der unteren Kante, zu gelangen.
Das Programm wird mit **Stop** angehalten. 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 die Fahrtstrecken der Grafik-Schildkröte im Unterprogramm "Anzeige" ändern:

```
schritte=(FNt(ey)-20)*8
```

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 Hauptprogramm, so daß nach jeder Messung die Temperatur gleichzeitig in °C, Kelvin und °F auf dem Bildschirm angezeigt wird:

```
tc=FNt(ey)
LOCATE 1,1
PRINT "Temperatur=";tc;
        "Grad Celsius"
tk!=tc+273.15
LOCATE 2,14
PRINT USING"###.##";tk!;
PRINT "Kelvin"
tf!=tc*1.8+32
LOCATE 3,14
PRINT USING"###.##";tf!;
PRINT "Grad Fahrenheit"
```

Nach Programmstart mit **Start** 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 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 kein 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 früherer schon gelernt haben - mit dem Befehl m3l bzw. m3r, ausgeschaltet mit m3a. Der Heißleiter ist mit dem Analogeingang EY verbunden und wird also mit ey abgefragt. Geben Sie Folgendes ein:

```
REM Programmanfang
DEF FNt(x)=INT(200-40*LOG(x))
init
tsoll=30
Schleife:
  te=FNt(ey)
  LOCATE 1,1
  PRINT "Temperatur=";te;
        "Grad Celsius"
  IF te<tsoll THEN CALL m3r
  IF te>tsoll THEN CALL m3a
GOTO Schleife
REM Programmende
```

Die meisten Programmzeilen kennen wir schon; die Umrechnungsfunktion in °C, die Sie natürlich wieder durch Ihre bessere, individuelle ersetzen sollten, wird in der ersten Zeile definiert. Mit der nächsten Zeile wird das Interface in den Grundzustand gebracht (initialisiert). Die ersten Zeilen in der Wiederholschleife messen die Temperatur am Heißleiter, die umgerechnet, in der Variablen te gespeichert und ausgedruckt wird.

In der Variablen tsoll geben wir eine Temperatur vor, bei der der Brenner abschalten soll (Sollwert), beispielsweise 30 °C. Natürlich müssen wir jetzt den tatsächlichen Temperaturwert (Istwert) nach jeder Messung mit dem Sollwert vergleichen. Dazu dienen die beiden IF-Anweisungen.

Wenn die Isttemperatur te kleiner als die Solltemperatur tsoll ist, wird der Brenner eingeschaltet (CALL m3r). Hat der Istwert den Sollwert überschritten, wird der Brenner mit CALL m3a ausgeschaltet. Danach wird der Programmabschnitt wiederholt - es folgt eine erneute Temperaturmessung.

Und nun zur Praxis: Starten Sie das Programm mit **Start**. 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. 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. Wenn dagegen die Lampe blinkt, so ist das völlig normal: die Berechnung der Temperatur und der Ausdruck am Schirm dauern etwa gerade ½ Sekunde, so daß die Schutzschaltung des Interface immer wieder den Ausgang M3 kurzfristig abschaltet.)

Nach Erreichen der Solltemperatur heizt der Brenner 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

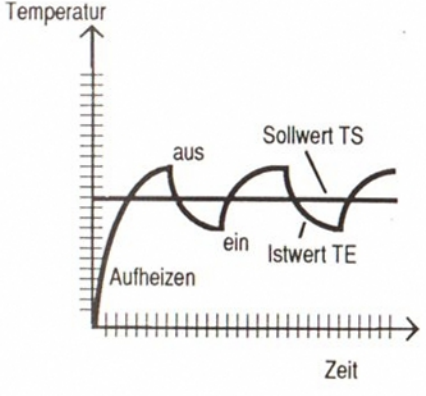


Bild 7.2: Regelkurve eines Zweipunktreglers

7.1 zur Anzeige der Temperatur ("Anzeige") und zum Löschen des Bildschirms ("neuesBild") hinzufügen. Die Unterprogramme werden wie in Kapitel 7.1 aufgerufen, d.h. GOSUB neuesBild nach dem Initialisieren des Interface und GOSUB Anzeige nach der PRINT-Anweisung.
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 **Stop** ändern Sie die Zeile mit der Festlegung des Sollwerts und die IF-Anweisungen ab:

```
tsoll=30
ot=tsoll+2
ut=tsoll-2

  IF te<ut THEN CALL m3r
  IF te>ot THEN CALL m3a
```

und starten das Programm wieder mit **Start**. 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:

```
tsoll=35
```

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:

```
tsoll=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 dem der Regelvorgang übersichtlich dargestellt wird. Sein Name ist OFEN.

7.3. Steuerung der Kühlung: Gebläse

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.

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 Kühlgebläse.
Dieses Prinzip wird oftmals dann benutzt, wenn sich die Wärmequelle nicht abschalten läßt. So kann man z.B. einen Automotor während der Fahrt 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 ein Testprogramm, ob alles richtig verdrahtet ist. Geben Sie bitte ein:

```
REM Programmanfang
init
FOR i=1 TO 100
  m1r
  m3r
  LOCATE 1,1
  PRINT ey
NEXT i
m1a
m3a
REM Programmende
```

Nach Programmstart muß der Lüfter und die Lampe aufleuchten. Außerdem 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:

```
REM Programmanfang
DEF FNt(x)=INT(200-40*LOG(x))
init
ot=30
ut=25
m3r
Schleife:
  te=FNt(ey)
  LOCATE 1,1
  PRINT ;"Temperatur=";te;
        "Grad Celsius"
  IF te>ot THEN CALL mlr
  IF te<ut THEN CALL mla
GOTO Schleife
REM Programmende
```

Damit sind die ersten beiden Bedingungen erfüllt. Die Lampe brennt dauernd (m3r), und die Temperatur wird gemessen (erste Zeile der Wiederholschleife). Sie ist in der Variablen te gespeichert. Die beiden IF-Anweisungen bilden den Soll-/Istwert-Vergleich für den Regelkreis.
Starten Sie nun das Programm mit **Start**. Die Lampe brennt und wärmt den Heißleiter auf. Das braucht zunächst seine Zeit. Wird die obere Grenztemperatur (ot) überschritten, schaltet der Lüfter ein. 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.
Um die Temperaturen zu verfolgen, bei denen das Gebläse ein- und ausschaltet, wurde die PRINT-Anweisung vorgesehen.
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 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 Programm ein:

```
REM Programmanfang
DEF FNt(x)=INT(200-40*LOG(x))
init
```

```
ot=30
ut=25
m3r
Schleife:
  te=FNt(ey)
  LOCATE 1,1
  PRINT ;"Temperatur=";te;
          "Grad Celsius"
  IF te>ot THEN CALL m1r
  IF te<ut THEN CALL m1a
IF INKEY$="" THEN GOTO Schleife
m3a
m1r
WHILE te>20
  te=FNt(ey)
  LOCATE 1,1
  PRINT "Temperatur= ";te;
          " Grad Celsius"
WEND
m1a
REM Programmende
```

Nach **Start** läuft das Programm zunächst wie vorher. Abschalten läßt sich der "Motor" (Lampe) nun durch Drücken einer beliebigen Taste. Der Lüfter läuft noch solange weiter, bis die Temperatur unter 20 °C gesunken ist. Danach endet das Programm. Die Temperaturabfrage ist diesmal Bestandteil einer WHILE...WEND-Schleife. Beachten Sie, daß beim Übergang in den zweiten Teil des Programms te zuerst geprüft und dann erst durch die Temperaturmessung bestimmt wird. Dies geht in diesem Fall gut, weil te noch von der letzten Messung des ersten Teils eine korrekte Temperaturangabe enthält.

Wenn Sie den Temperaturverlauf mit der Schildkröte mitprotokolliert hatten, so sollten Sie nicht vergessen, die Grafik am Programmende abzuschalten.

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.

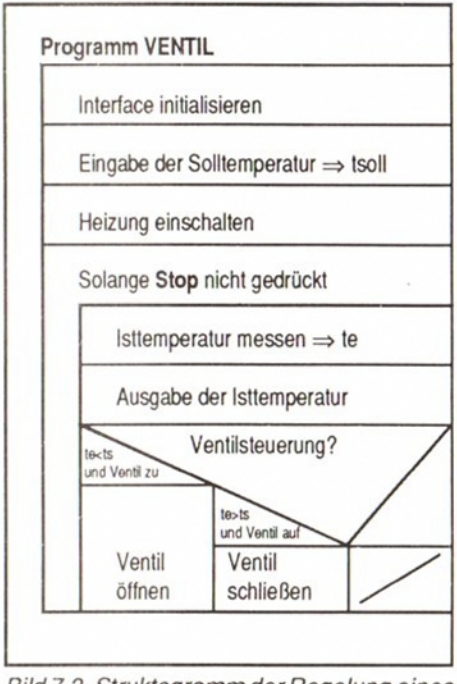


Bild 7.3. Struktogramm der Regelung eines Thermostatventils

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 Schalter (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.

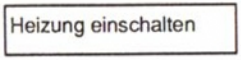
Unser Programm läuft nun folgendermassen ab: Man gibt eine Solltemperatur vor, die das Modell erreichen und halten muß (z.B. 36°C). Der Brenner wird eingeschaltet, der Schieber geschlossen und die Isttemperatur gemessen. Ist sie kleiner als der Sollwert, wird der Schieber geöffnet. Er bleibt solange 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.

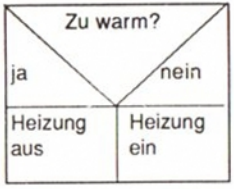
Hat's geklappt? Prima! Sehen wir uns aber nun das Programm an:

```
DEF FNt(x)=INT(200-40*LOG(x))
init
INPUT "Solltemperatur:";tsoll
```

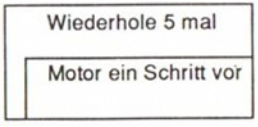
Unter einem Struktogramm versteht man eine zeichnerische Darstellung des Programms. Das Struktogramm wird von oben nach unten gelesen. Treffen Sie ein Rechteck an, so wird die darin beschriebene Aktion ausgeführt:



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:




```
ot=tsoll+1
ut=tsoll-1
s=0
m3r
Schleife:
  te=FNt(ey)
  LOCATE 1,1
  PRINT "Temperatur =";te;
          "Grad Celsius"
  IF te<ut  AND s=0 THEN
          s=1 : m1v
  IF te>ot AND s=1 THEN
          s=0 : m1v
GOTO Schleife
REM Programmende
```

Zu Beginn des Experiments soll der Schieber zwischen Lampe und Heißleiter stehen (s=0). Starten Sie das Programm mit **Start**. Die Eingabe der Solltemperatur erfolgt mit der INPUT-Anweisung. Wählen Sie den Wert nicht zu niedrig, da die Lampe ganz schön heizt.
Die nächsten beiden Zeilen legen die Grenzwerte für "Schieber öffnen" und "Schieber schließen" fest. Die Variable s hält die Schieberstellung fest (0=geschlossen). Nach der Temperaturmessung erfolgt der Soll-/Istwert-Vergleich. Ist die Isttemperatur te kleiner als der Sollwert und (AND) der Schieber geschlossen, öffnet der Schieber (Nachsatz des ersten IF). Das merken wir uns mit s=1. In der nachfolgenden IF-Anweisung ist es genau umgekehrt: Ist te>ot und (AND) der Schieber offen (s=1), wird er geschlossen. Das Programm wiederholt den Regelmechanismus in einer Endlosschleife.
Auch zu diesem Versuch gibt es wieder ein fertiges Programm, das den Namen VENTIL 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:

```
REM Programmanfang
init
m11
REM Programmende
```

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.

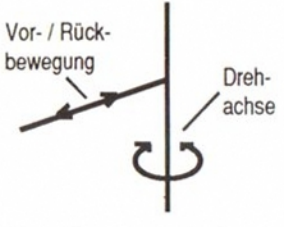


Bild 8.1: Bewegungsachsen unseres Schweißroboters.

```
m1r
```

anstelle m1l und erneutem Programmstart dreht er sich wieder zurück. Eine präzisere Drehung des Roboters erhalten Sie wieder mit der Verwendung der Schrittkommandos:

```
m1v  bzw.  m1z
```

Wenn Sie

```
m2l
```

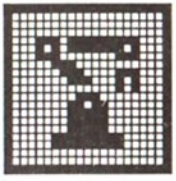
eingeben und das Programm starten, wird sich die Spindel am Roboterarm kurz drehen. Stellen Sie die Spindel wieder in die Grundstellung (Schweißarm ganz eingezogen). Mit

```
REM Programmanfang
init
FOR i=1 TO 12
  m2v
NEXT i
REM Programmende
```

und **Start** fährt der Arm ganz aus. Ändern Sie in dem Schleifenkörper den Befehl in m2z, 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

```
REM Programmanfang
init
FOR i=1 TO 10
  m1v
NEXT i
REM Programmende
```

und **Start** 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 m1z im Schleifenkörper 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 mit der Programmierung unseres Roboters befassen. 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.

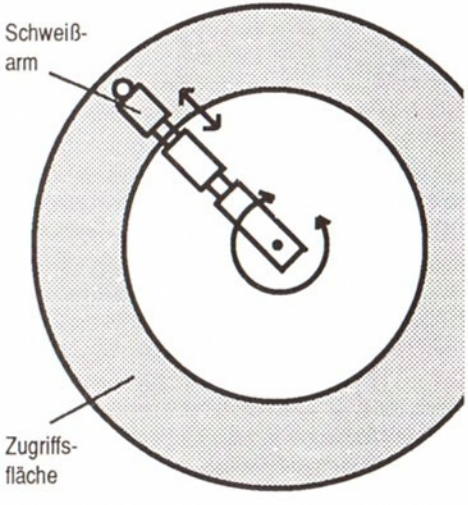


Bild 8.2: Zugriffsfläche des Schweißroboters.

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

m1v bzw. m1z
für die Drehachse und

m2v bzw. m2z
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

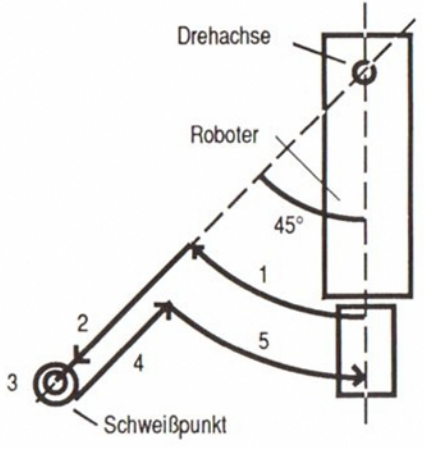


Bild 8.3: Bewegungsablauf beim Schweissen.

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.

```
REM Programmanfang
init
Schleife:
  REM Arm rechts
  FOR i=1 TO 5
    m1z
  NEXT i
  REM Arm vor
  FOR i=1 TO 12
    m2v
  NEXT i
  REM Schweißen
  FOR i=1 TO 1000
    m3r
  NEXT i
  m3a
  REM Arm zurück
  FOR i=1 TO 12
    m2z
  NEXT i
  REM Arm links
  FOR i=1 TO 5
    m1v
  NEXT I
  REM Pause
  FOR i=1 TO 10000
  NEXT i
GOTO Schleife
REM Programmende
```

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 m1z ausgeführt. Dabei dreht sich der Arm jedesmal um 9°. Auch das Schweißen ist mit

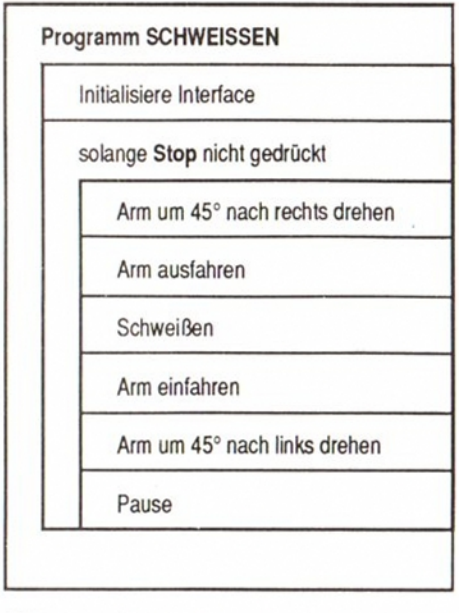


Bild 8.4: Struktogramm zum Programm "Schweißen".

FOR...NEXT-Schleifen aufgebaut. Hier dienen sie aber dazu, eine bestimmte Zeit verstreichen zu lassen. Die Pause am Ende des Durchlaufs ist ebenfalls mit einer FOR...NEXT-Schleife programmiert.
Wenn Sie das Programm eingegeben haben und es mit **Start** starten, wird der Roboter seine Arbeit nach dem Programm genauso ausführen, wie wir es ihm vorgeschrieben haben. Anhalten läßt er sich mit der **Stop**-Funktion.
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.:

12**Schritte_bis_zum_Objekt_**V

Z - Schweißarm **z**urück
R - Roboter nach **r**echts drehen
L - Roboter nach **l**inks drehen
S - **s**chweißen
P - **P**ause
E - **E**nde 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:

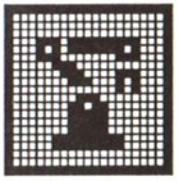
5R	(Roboter um fünf Schritte nach rechts)
12V	(Schweißarm um zwölf Schritte vor)
1000S	(1000 Schleifendurchläufe lang schweißen)
12Z	(Schweißarm um zwölf Schritte zurück)
5L	(Roboter um fünf Schritte nach links)
10000P	(10000 Schleifendurchläufe lang pausieren)
E	(Ende des Programms)

Die Tabelle im Computer wird in Form von DATA-Zeilen geführt, z.B.:

```
Bewegung:
DATA 5R,12V,1000S,12Z,5L,
        10000P,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 der READ-Anweisung. In der Variablen der READ-Anweisung 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:

```
REM Programmanfang
init
RESTORE Bewegung
Aktion:
```



```
  READ a$
  w=VAL(a$)
  k$=RIGHT$(a$,1)
  IF k$="V" THEN GOSUB Vor
  IF k$="Z" THEN GOSUB Zurueck
  IF k$="R" THEN GOSUB Rechts
  IF k$="L" THEN GOSUB Links
  IF k$="S" THEN GOSUB
          Schweissen
  IF k$="P" THEN GOSUB Pause
IF k$<>"E" THEN GOTO Aktion
REM Programmende
```

Nach der END-Anweisung werden die Unterprogramme und die DATA-Zeile eingefügt:

```
Bewegung:
DATA 5R,12V,1000s,12Z,5L,
          10000P,E

Vor:
  FOR i=1 TO w
    m2v
  NEXT i
RETURN

Zurueck:
  FOR i=1 TO w
    m2z
  NEXT i
RETURN

Rechts:
  FOR i=1 TO w
    m1z
  NEXT i
RETURN

Links:
  FOR i=1 TO w
    m1v
  NEXT i
RETURN

Schweissen:
  FOR i=1 TO w
    m3r
  NEXT i
  m3a
RETURN

Pause:
  FOR i=1 TO w
  NEXT i
RETURN
```

Starten Sie das Programm mit **Start**. Der Roboter durchläuft einmal den vorigen Bewegungsablauf.

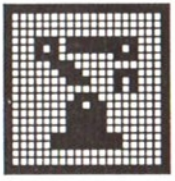
Probieren Sie nun eigene Arbeitsabläufe aus - Sie brauchen nur die Tabelle entsprechend zu ändern. Geben Sie z.B. ein

```
Bewegung:
DATA 6R,12V,200S,12Z,12L
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. Betten Sie den Hauptteil des Programms in eine Endlosschleife ein:

```
REM Programmanfang
init
Schleife:
  RESTORE bewegung
  Aktion:
    READ a$
    w=VAL(a$)
    k$=RIGHT$(a$,1)
    IF k$="V" THEN GOSUB Vor
    IF k$="Z" THEN GOSUB
          Zurueck
    IF k$="R" THEN GOSUB
          Rechts
    IF k$="L" THEN GOSUB Links
    IF k$="S" THEN GOSUB
          Schweissen
    IF k$="P" THEN GOSUB Pause
  IF k$<>"E" THEN GOTO Aktion
GOTO Schleife
REM Programmende
```

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.



8.4 Sensorführung des Roboters: Mit eigenen Sinnen

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".

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 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.

Zu Beginn des Programms starten wir wieder mit der wohlbekannten Funktionsdefinition zur Umrechnung des Analogwertes in °C gemäß Kapitel 7.

```
DEF FNt(x)=200-40*LOG(x)
```

Dort steht auch beschrieben, wie Sie sich

durch Kalibrierung des NTC eine genauere Temperaturanzeige des NTC verschaffen. Den Programmabschnitt zum Schweißen ersetzen wir durch die folgenden Zeilen:

```
Schweissen:
  m3r
  WHILE FNt(ey)<w
  WEND
  m3a
RETURN
```

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:

```
bewegung:
DATA 5R,12V,35S,12Z,5L,10000P,E
```

Geben Sie die Zeilen ein und starten das Programm mit **Start**. 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.
Eine andere Variante des Programms ist ROBOTGRA. 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.

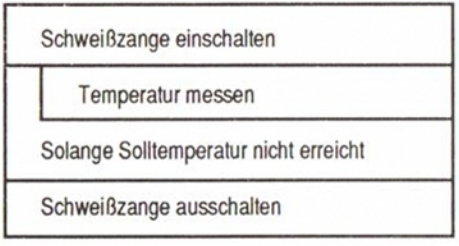


Bild 8.5: Struktogramm zum Programm "Schweißen mit Sensor".

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 eine Stoßstange, hinter der 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 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 Amiga-BASIC und das Programm INIT geladen haben, geben Sie Folgendes ein:

```
REM Programmanfang
init
m1v
REM Programmende
```

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 m2v anstelle m1v und nochmaligem Programmstart passiert dasselbe mit dem linken Motor. Die Schildkröte dreht sich nach rechts. Lassen wir sie im Kreis fahren:

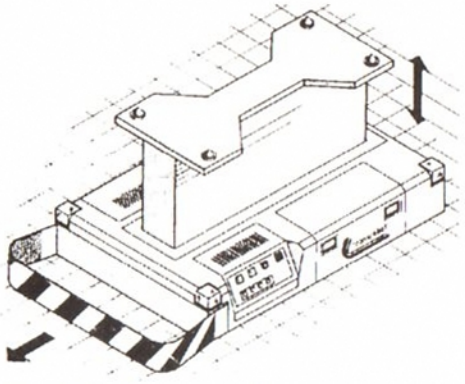


Bild 9.1: Ein fahrerloses Transportfahrzeug mit Hubtisch

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.

```
REM Programmanfang
init
FOR i=1 TO 150
  m2v
NEXT i
REM Programmende
```

Geben Sie die Zeilen ein und starten Sie das Programm mit **Start**. 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

```
m1v : m2v
```

als Schleifenkörper 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:

```
REM Programmanfang
init
ti
tv(1)
REM Programmende
```

Starten Sie das Programm und die Schildkröte fährt vorwärts. Das Kommando ti 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

Fassen wir die Schildkrötenbefehle zusammen:

ti
Die Schildkröte wird initialisiert.

tv(s)
*Die Schildkröte fährt um s*5 mm vorwärts.*

tz(s)
*Die Schildkröte fährt um s*5 mm zurück.*
(s = 0 .... 32767)

tr(d)
Die Schildkröte dreht sich um d Grad nach rechts.

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

Schritt vorwärts. Wird es ersetzt durch:

```
tz(1)
```

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

```
tv(20)
```

zurück. Nun fährt sie genau 10 cm vor. Damit kennen wir auch schon die Schrittweite für ein Vorwärtskommando tv: 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 als Bewegungskommando:

```
tz(20)
```

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 als Bewegungskommando:

```
tr(5)
```

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

```
tl(5)
```

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, der in dem Bewegungskommando angegeben wird:

```
tr(90)
```

Bevor Sie das Programm mit **Start** starten, 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:

```
tr(37)
```

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:

```
tr(400)
```

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:

```
tr(40)
```

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:

```
REM Programmanfang
init
ti
tv(10)
tr(90)
tz(16)
REM Programmende
```

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!

```
REM Programmanfang
init
ti
RESTORE Bewegung
Aktion:
  READ a$
  w=VAL(a$)
  k$=RIGHT$(a$,1)
  IF k$="V" THEN CALL tv(w)
  IF k$="Z" THEN CALL tz(w)
  IF k$="R" THEN CALL tr(w)
  IF k$="L" THEN CALL tl(w)
IF k$<>"E" THEN GOTO Aktion
REM Programmende
```

Die Bahn der Schildkröte wird wieder in DATA-Zeilen codiert, die nach der END-Anweisung angehängt werden, z.B.:

```
Bewegung:
DATA 20V,60R,20V,60R,20V,60R,
       20V,60R,20V,60R,20V,300L,E
```

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 ge-

setzt. Ein ähnliches Programm befindet sich auf der Diskette; sein Name ist ROUTENUM.
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 tv das Kommando gv 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 also in jedem Nachsatz der IF-Anweisungen ergänzt:

```
REM Programmanfang
init
ti
ge(WINDOW(7))
RESTORE Bewegung
Aktion:
  READ a$
  w=VAL(a$)
  k$=RIGHT$(a$,1)
  IF k$="V" THEN
     CALL tv(w) : CALL gv(w)
  IF k$="Z" THEN
     CALL tz(w) : CALL gz(w)
  IF k$="R" THEN
     CALL tr(w) : CALL gr(w)
  IF k$="L" THEN
     CALL tl(w) : CALL gl(w)
IF k$<>"E" THEN GOTO Aktion
ga
REM Programmende

Bewegung:
DATA 20V,60R,20V,60R,20V,60R,
       20V,60R,20V,60R,20V,300L,E
```

Die Programme auf der Diskette benutzen Hintergrundbilder, die mit gload geladen werden (s. Kap. 6). Dies steht Ihnen ebenso frei. Fügen Sie nach dem ge-Kommando ein:

```
COLOR 2
Bild$="bilder:TURTLE.PIC"+CHR$(0)
gload(SADD(Bild$))
```

Außerdem fügen Sie vor dem ga-Kommando ein:

```
COLOR 1
```

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 Arbeitsdiskette die Schublade "Bilder" enthalten muß, damit das Bild geladen werden kann. Dies ist normalerweise der Fall, wenn Sie entsprechend der Anweisungen aus Kapitel 3 vorgegangen sind und die Schublade "Bilder" nicht gelöscht haben.

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 das Programm INIT geladen ist. Geben Sie dann ein:

```
REM Programmanfang
init
ti
tv(1)
REM Programmende
```

Nach dem Start des Programms muß sich die Schildkröte 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

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?

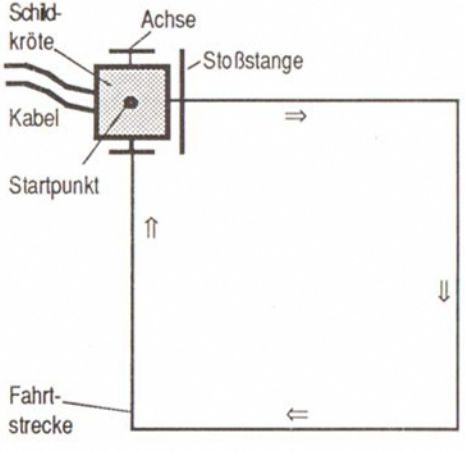


Bild 9.2: Fahrstrecke der Schildkröte.

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:

```
REM Programmanfang
init
ti
auf$=CHR$(28)
ab$=CHR$(29)
re$=CHR$(30)
li$=CHR$(31)
Schleife:
  Taste:
    k$=INKEY$
  IF k$<>auf$ AND k$<>ab$ AND
        k$<>re$ AND k$<>li$
        AND UCASE$(k$)<>"E"
        THEN GOTO Taste
  IF k$=auf$ THEN CALL tv(1)
  IF k$=ab$ THEN CALL tz(1)
  IF k$=re$ THEN CALL tr(5)
  IF k$=li$ THEN CALL tl(5)
IF UCASE$(k$)<>"E" THEN
     GOTO Schleife
REM Programmende
```

In den ersten Zeilen nach der Initialisierung werden die Zeichencodes für die Cursortasten festgelegt.
Die Tastaturabfrage zu Beginn der Wiederholschleife wartet solange, bis eine der zugelassenen Tasten (Cursortasten und "E") gedrückt wird. Der Tastencode steht

dann in der Variablen k$. Welche Taste gedrückt wurde, wird durch IF-Anweisungen 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 **Start**. 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. Außerdem werden noch folgende Variablen benötigt:

Die Variable 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.

Mit der Variablen schritt wird die Anzahl gleicher Schritte gezählt. Außerdem wird noch eine Kopie des Cursorcodes in l$ 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 "Tabelleneintrag".

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

Dort werden die vereinbarten Kommandos aus der Schrittzahl in der Variablen schritt und dem Kennbuchstaben in code$ zusammengesetzt. Die Schrittzahl wird wieder zurückgesetzt und außerdem eine neue Kopie des Cursorcodes angelegt. Der Zeiger i in das Feld a$ wird auch erhöht.

```
REM Programmanfang
DIM a$(500)
init
ti
auf$=CHR$(28)
ab$=CHR$(29)
re$=CHR$(30)
li$=CHR$(31)
l$=""
i=1
schritt=0
Schleife:
  Taste:
    k$=INKEY$
  IF k$<>auf$ AND k$<>ab$ AND
     k$<>re$ AND k$<>li$
     AND UCASE$(k$)<>"E"
     THEN GOTO Taste
  IF k$<>l$ THEN
     GOSUB Tabelleneintrag
  IF k$=auf$ THEN
   CALL tv(1):schritt=schritt+1
  IF k$=ab$ THEN
   CALL tz(1):schritt=schritt+1
  IF k$=re$ THEN
   CALL tr(5):schritt=schritt+5
  IF k$=li$ THEN
   CALL tl(5):schritt=schritt+5
IF UCASE$(k$)<>"E" THEN
   GOTO Schleife
a$(i)="E"
REM Programmende

Tabelleneintrag:
  IF l$<>"" THEN GOTO Ausgang
    IF l$=auf$ THEN
      code$="V"
    IF l$=ab$ THEN
      code$="Z"
    IF l$=re$ THEN
      code$="R"
    IF l$=li$ THEN
      code$="L"
    a$(i)=STR$(schritt)+code$
    i=i+1
    schritt=0
  Ausgang:
  l$=k$
RETURN
```

Die Funktion STR$ dient dazu, einen Zahlenwert wie hier die Anzahl der Schritte in der Variablen schritt in eine Zeichenkette umzuwandeln, d.h. in diesem Fall in eine Folge von Dezimalziffern.

Starten Sie das Progamm jetzt mit **Start**.

Bislang merken Sie keinen Unterschied zu dem vorigen Programm. Beenden Sie das Programm durch Drücken der Taste "E". Geben Sie folgende Erweiterung am Ende des Hauptprogramms ein:

```
i=1
WHILE a$(i)<>"E"
  PRINT a$(i)
  i=i+1
WEND
WHILE INKEY$=""
WEND
END
```

Lassen Sie die Schildkröte fünf Schritte vorwärts fahren und drücken Sie dann "E". Auf dem Bildschirm erscheint

5V

Dies ist der erste erzeugte Befehl. Anschließend müssen Sie noch eine beliebige Taste drücken, um das Programm zu beenden. Jetzt wollen wir den erlernten Weg selbständig von der Schildkröte abfahren lassen. Dazu tauschen wir das Programmstück zum Ausdrucken, das wir eben eingegeben haben, gegen ein ähnliches Programmstück wie bei der Ausführung der DATA-Zeilen aus (s. voriges Kapitel). Das Lesen der DATA-Zeilen wird jetzt durch ein Lesen des Feldes a$ ersetzt. Außerdem wurde der Programmteil etwas kürzer formuliert.

```
Ausführschleife:
  i=1
  WHILE a$(i)<>"E"
    PRINT a$(i)
    w=VAL(a$(i))
    k$=RIGHT$(a$(i),1)
    IF k$="V" THEN CALL tv(w)
    IF k$="Z" THEN CALL tz(w)
    IF k$="R" THEN CALL tr(w)
    IF k$="L" THEN CALL tl(w)
    i=i+1
  WEND
  WHILE INKEY$=""
  WEND
GOTO Ausfuehrschleife
REM Programmende
```

Geben Sie die Programmzeilen ein. Zu Beginn des Ausführteils wird der Zähler i wieder auf den Anfang des Feldes a$ gesetzt. Danach wird jeder Befehl gelesen, ausgedruckt und ausgeführt. Nach jedem Durchlauf wartet das Programm auf einen Tastendruck, bevor es mit dem nächsten

Durchlauf beginnt.
Starten Sie nun das Programm mit **Start**. 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, lösen Sie die **Stop**-Funktion aus, setzen die Schildkröte wieder auf den Ausgangspunkt und beginnen das Programm erneut mit **Start**.
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, drücken Sie eine beliebige Taste der Tastatur.
Durch die Kreisfahrten der Schildkröte wikkelt 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. Es kann zudem die Routen auf der Diskette abspeichern und wieder laden, auf dem Drucker oder dem Bildschirm ausgeben.

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.

9.5 Routenplanung am Bildschirm: Voraussicht

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. Wir bringen die nachfolgenden Änderungen in dem ersten Teil des Programms an. Der Ausführteil und das Unterprogramm "Tabelleneintrag" bleiben unverändert:

```
REM Programmanfang
DIM a$(500)
init
ti
ge(WINDOW(7))
auf$=CHR$(28)
ab$=CHR$(29)
re$=CHR$(30)
li$=CHR$(31)
l$=""
i=1
schritt=0
Schleife:
  Taste:
    k$=INKEY$
  IF k$<>auf$ AND k$<>ab$ AND
     k$<>re$ AND k$<>li$
     AND UCASE$(k$)<>"E"
     THEN GOTO Taste
  IF k$<>l$ THEN
     GOSUB Tabelleneintrag
  IF k$=auf$ THEN
   CALL gv(1):schritt=schritt+1
  IF k$=ab$ THEN
   CALL gz(1):schritt=schritt+1
  IF k$=re$ THEN
```

Das Programm ROUTEDIT 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, das ja Routen von der Diskette laden kann. Dafür entschädigt Sie ROUTEDITmit 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.

```
   CALL gr(5):schritt=schritt+5
  IF k$=li$ THEN
   CALL gl(5):schritt=schritt+5
IF UCASE$(k$)<>"E" THEN
     GOTO Schleife
ga
CLS
a$(i)="E"
Ausführschleife:
     :
```

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.
Die Zeilen werden nach dem ge-Kommando eingefügt:

```
COLOR 2
Bild$="bilder:TURTLE.PIC"+CHR$(0)
gload(SADD(Bild$))
```

Die folgende Zeile kommt nach das ga-Kommando:

```
COLOR 1
```

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.

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:

```
REM Programmanfang
init
Schleife:
  LOCATE 1,1
  PRINT e5
GOTO Schleife
REM Programmende
```

Geben Sie die Zeilen ein, und starten Sie das Programm mit **Start**. 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 E5, an dem der Schalter liegt, wird mit Funktion e5 abgefragt. Mit **Stop** 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:

```
REM Programmanfang
init
ti
tv(200)
REM Programmende
```

Die Schildkröte fährt vorwärts und stoppt, wenn die Stoßstange an das Buch stößt. Das Ganze geschah ohne zusätzliche Abfrage, denn das Kommando tv prüft schon selbst die Betätigung des Tasters. Es arbeitet in dieser Beziehung genauso wie die Funktion e5, nur daß der Funktionswert

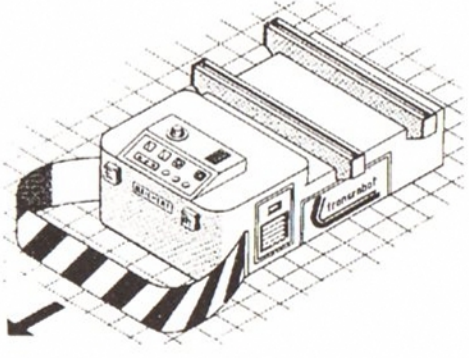


Bild 10.1: Industrieroboter mit Not-Stop-Bügel

nicht zurückgegeben wird, sondern gegebenenfalls gleich in einen Motorstopp umgesetzt wird. Sie können daher folgende Ergänzung des Programms nach dem Bewegungskommando vornehmen:

```
PRINT e5,ts
WHILE INKEY$=""
WEND
REM Programmende
```

Starten Sie das Programm mit **Start**. 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 der Funktion 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 tv vorgenommen, denn die Schildkröte hat nur vorne eine Stoßstange. Insbesondere muß es der Schildkröte möglich sein, auch bei gedrückter Stoßstange mit dem Kommando tz zurückzufahren, um sich wieder vom Hindernis zu entfernen.
Wie weit ist die Schildkröte gekommen, bis sie anstieß? Die Funktion ts zeigt dies an. Sie enthält nach Beendigung des Kommandos tv die tatsächlich gefahrene Schrittzahl. Wurde der Weg in voller Länge gefahren, so enthält sie natürlich gerade den Zahlenwert, der im Argument des tv-Kommandos 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 schwenken.

```
REM Programmanfang
init
ti
Schleife:
  WHILE e5=1
    tv(200)
  WEND
  tz(10)
```

Fassen wir die verschiedenen Rückmeldungen der Schildkröte zusammen:

tx

Die Funktion ermittelt die derzeitige X-Position der Schildkröte.

ty

Die Funktion ermittelt die derzeitige Y-Position der Schildkröte.

tk

Die Funktion ermittelt den derzeitigen Kurs der Schildkröte.

e5

Die Funktion ermittelt den derzeitigen Zustand der Stoßstange der Schildkröte: 1=frei, 0=gedrückt.

ts

Die Funktion ermittelt die Zahl der Schritte des letzten Schildkrötenkommandos. Im Fall der Vorwärtsfahrt und vorzeitigem Stop durch Aktivieren der Stoßstange ist die Schrittzahl geringer als der Wert des Parameters des tv-Kommandos. Bei den Drehkommandos wird die Anzahl der Schritte, nicht der Drehwinkel, ermittelt!

```
  tr(90)
  WHILE INKEY$=""
  WEND
GOTO Schleife
REM Programmende
```

Setzen Sie die Schildkröte parallel zu einer Wand und starten das Programm mit **Start**. Sie wird jetzt nacheinander bis zu jeder Wand fahren, dort anhalten, sich um 90° drehen und nach Tastendruck weiterfahren. Vor der Drehbewegung muß sie etwas zurücksetzen, damit sie mit den Rädern nicht anstößt. Mit **Stop** läßt sie sich anhalten. Wenn wir die Schritte zwischen zwei gegenüberliegenden Wänden abfragen, wissen wir auch, wie groß der Raum ist. Geben Sie ein:

```
REM Programmanfang
init
ti
FOR k=1 TO 4
  WHILE e5=1
    tv(200)
  WEND
  IF k=3 THEN sb=ts
  IF k=4 THEN sl=ts
  tz(10)
  tr(90)
NEXT k
PRINT "Breite:";sb*5+80;" mm"
PRINT "Länge :";sl*5+80;" mm"
WHILE INKEY$=""
WEND
REM Programmende
```

Setzen die Schildkröte parallel zur Querwand mit Fahrtrichtung nach rechts und starten mit **Start**. Sie wird jetzt alle vier Wände anfahren und stoppen. Dies erreichen wir mit der FOR...NEXT-Schleife (vier Durchläufe). Die Übertragung der Schrittzahl für die Breite erfolgt bei der Fahrt von rechts nach links, also bei k=3. 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 Schildkrötenfunktio-

nen zur Verfügung:

```
tx
```

ermittelt die derzeitige X-Position der Schildkröte. Ähnliches gilt für:

```
ty
```

Diese Funktion ermittelt die Y-Position. Auch der derzeitige Kurs der Schildkröte kann ermittelt werden:

```
tk
```

Der Kurs zählt in Winkelgraden im Uhrzeigersinn, beginnend mit 0 in der Startrichtung. Die Positionsangaben beziehen sich ebenfalls immer auf die Position der Schildkröte zu dem Zeitpunkt, wo das ti-Kommando gegeben wurde. Es kann daher durchaus sinnvoll sein, das ti-Kommando mehr als einmal im Programm zu benutzen, wenn man sich eine neue Ausgangslage wählen will. Mit diesen Kommandos schreiben wir obiges Programm um:

```
REM Programmanfang
init
ti
FOR k=1 TO 4
  WHILE e5=1
    tv(200)
  WEND
  IF k=1 THEN oben=ty
  IF k=2 THEN rechts=tx
  IF k=3 THEN unten=ty
  IF k=4 THEN links=tx
  tz(10)
  tr(90)
NEXT k
PRINT "Breite:";(rechts-links)
     *5+80;" mm"
PRINT "Länge :";(oben-unten)
     *5+80;" mm"
WHILE INKEY$=""
WEND
REM Programmende
```

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 heißt. Mit ihm können Sie ebenfalls die Schildkrötenwelt erforschen.

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:

```
REM Programmanfang
init
ti
WHILE e5=1
  tv(200)
WEND
REM Programmende
```

Geben Sie die Zeilen ein und starten das Programm mit **Start**. Die Schildkröte bleibt 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. Da wir den Bewegungsablauf des Ausweichens noch öfter brauchen werden, schreiben wir ihn als Unterprogramm:

```
REM Programmanfang
init
ti
WHILE e5=1
  tv(200)
WEND
GOSUB Ausweichen
REM Programmende

Ausweichen:
  tz(10)
  tr(90)
  tv(24)
  tl(90)
  tv(34)
RETURN
```

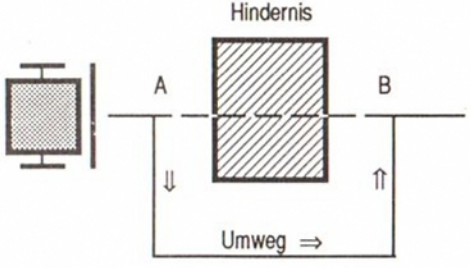


Bild 10.2: Fahrt um ein Hindernis.

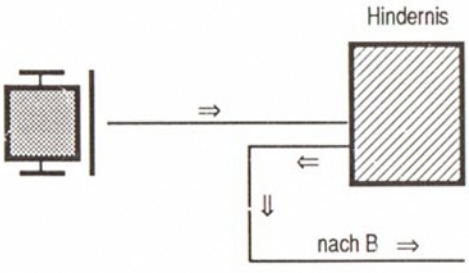


Bild 10.3: Ausweichen der Schildkröte nach rechts.

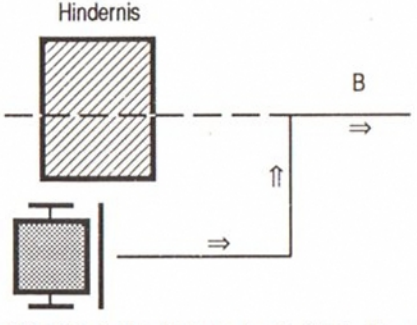


Bild 10.4: Rückfahrt der Schildkröte nach links.

Das Unterprogramm kommt wieder an das Ende des Programms, nach der Anweisung END.
Setzen Sie die Schildkröte zurück und starten das Programm mit **Start**. Nach dem Kommando tl(90) steht sie neben dem Hindernis. Die Schildkröte setzt ihren Weg nach vorn fort. Als Maß 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 Unterprogramm "ausweichen" so schreiben, daß die Schildkröte ein Hindernis beliebiger Breite abtasten kann und dann ihren Weg am Hindernis vorbei fortsetzt.

```
Ausweichen:
  nochmal:
    tz(10)
    tr(90)
    tv(24)
    tl(90)
    tv(34)
  IF ts<34 THEN GOTO nochmal
RETURN
```

Was jetzt noch fehlt, ist eine automatische Abtastung der Hindernislänge. Dazu benutzen wir natürlich auch wieder die Stoßstange und das gleiche Unterprogramm. Nehmen Sie in dem Hauptprogramm die entsprechenden Ergänzungen vor:

```
REM Programmanfang
init
ti
WHILE e5=1
  tv(200)
WEND
GOSUB Ausweichen
tl(90)
tv(34)
IF e5=0 THEN GOSUB Ausweichen
REM Programmende
```

Jetzt fehlt nur noch der Weg hinter dem Hindernis zurück auf die ursprüngliche Route. Bild 10.4 zeigt diesen Ablauf.
Wir ergänzen unser Programm vor der Anweisung "REM Programmende" mit folgenden Zeilen:

```
sx=tx
IF sx<0 THEN CALL tz(ABS(sx))
     ELSE CALL tv(sx)
tr(90)
```

```
tv(10)
REM Programmende
```

Zu Beginn des neuen Abschnitts benutzen wir die Funktion tx 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 **Start**. 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 HINDERNIS.

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:

```
REM Programmanfang
init
ti
tr(180)
tr(180)
REM Programmende
```

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

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.

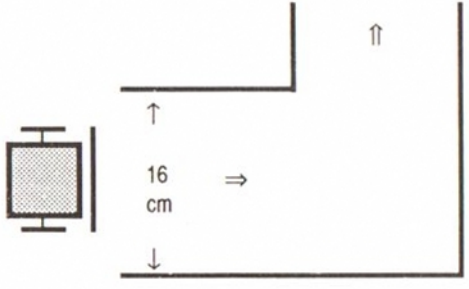


Bild 10.5: Labyrinth mit Abbiegung links.

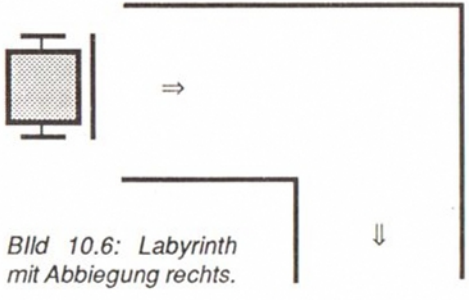


Bild 10.6: Labyrinth mit Abbiegung rechts.

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:

```
REM Programmanfang
init
ti
GOSUB BisZurWand
REM Programmende

BisZurWand:
  WHILE e5=1
    tv(200)
  WEND
  tz(8)
RETURN
```

Die Vorwärtsbewegung mit Erkennung des Hindernisses verlegen wir in das Unterprogramm "BisZurWand", da wir diesen Bewegungsteil später öfters benötigen. Das Unterprogramm kommt wieder ganz an das Programmende. Mit der dritten Zeile des Hauptprogramms sprechen wir das Unterprogramm an. Bei einem Hindernis fährt die Schildkröte sofort acht Schritte zurück, damit sie Platz für die anschließende Drehung hat.
Starten Sie das Programm mit **Start**; 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 Hauptprogramm:

```
REM Programmanfang
init
ti
GOSUB BisZurWand
REM rechts probieren
tr(90)
GOSUB BisZurWand
REM links probieren
tl(180)
GOSUB BisZurWand
REM Programmende
```

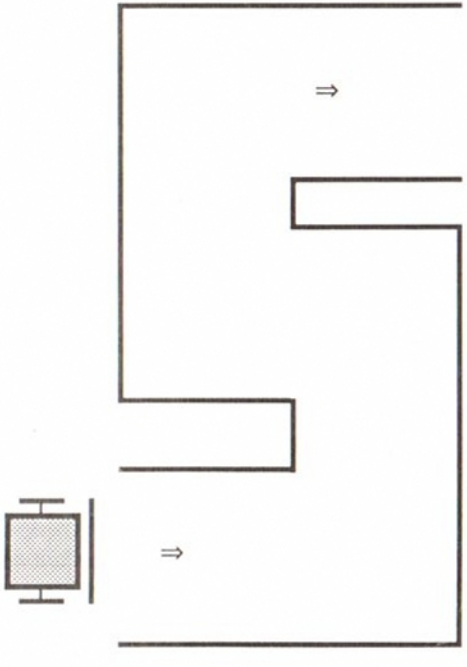


Bild 10.7: Einfaches Labyrinth.

Setzen Sie die Schildkröte wieder an den Eingang und starten das Programm. An der Querwand dreht sie sich um 90° nach rechts und fährt vor. Dafür benutzen wir das Unterprogramm "BisZurWand". 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° und fährt weiter vorwärts (ebenfalls mit dem Unterprogramm "BisZurWand"). Da hier der Ausgang unseres Labyrinths ist, wird sie ungehindert weiterfahren. Anhalten läßt sie sich mit der **Stop**-Funktion.

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 **Start**.

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 sowohl das Hauptprogramm als auch das Unterprogramm ergänzen:

```
REM Programmanfang
init
ti
GOSUB BisZurWand
NaechsteWand:
REM rechts probieren
tr(90)
GOSUB BisZurWand
IF s>20 THEN GOTO NaechsteWand
REM links probieren
tl(180)
GOSUB BisZurWand
IF s>20 THEN GOTO NaechsteWand
REM Programmende

BisZurWand:
  s=0
  WHILE e5=1
    tv(200)
    s=s+ts
  WEND
  tz(8)
RETURN
```

Im Unterprogramm werden die Vorwärtsschritte mit Hilfe der Funktion ts in s aufaddiert. Fährt die Schiidkröte nach einer Rechtsdrehung ungehindert weiter und stößt bei der nächsten Abbiegung gegen die Querwand, hätte sie beim bisherigen Programm eine Drehung um 180° ge-

Bild 10.8: Sackgasse.

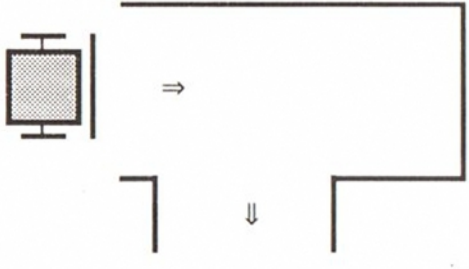


Bild 10.9: Sackgasse mit seitlichem Ausgang.

macht. Da sie aber bis dahin mehr als 20 Schritte gemacht hat, springt das Programm wieder zum Anfang (GOTO NaechsteWand). Die gleiche Abfrage findet sich auch nach dem zweiten Unterprogrammaufruf für die Abbiegung nach links.
Geben Sie die zusätzlichen Zeilen ein und starten Sie das Programm mit **Start**. 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 **Stop** 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! Nach der Sprungmarke "NaechsteWand:" wird eingefügt:

```
w=s-8
```

Vor dem Programmende kommt dann der Programmabschnitt zum Wenden:

```
REM zurück aus der Sackgasse
tr(90)
tz(w)
```

Die Schildkröte fährt erst zur rechten, dann zur linken Wand. Stößt sie dagegen, hat sie bis dahin jeweils weniger als 20 Schritte gemacht. Das Programm geht weiter zu dem neuen Abschnitt. Hier dreht sie sich wieder in Fahrtrichtung und fährt den Weg aus der Sackgasse zurück. Die Schrittzahl hat sie sich vorher in w gemerkt.
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.
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 zusätzlichen Programmzeilen und Veränderungen ein. Zur Kontrolle hier wieder das vollständige Programm:

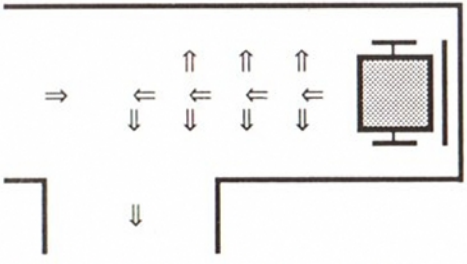


Bild 10.10: Ausfahrt aus der Sackgasse.

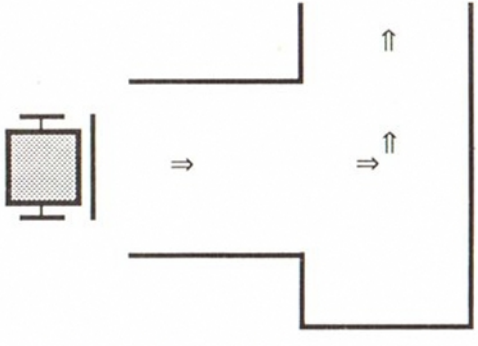


Bild 10.11: Abzweigung in eine Sackgasse.

```
REM Programmanfang
init
ti
GOSUB BisZurWand
NaechsteWand:
w=s-8
RechtsTesten:
tr(90)
GOSUB BisZurWand
IF s>20 THEN GOTO NaechsteWand
REM links probieren
tl(180)
GOSUB BisZurWand
IF s>20 THEN GOTO NaechsteWand
REM Zurück aus der Sackgasse
tr(90)
tz(10)
w=w-10
IF w>20 THEN GOTO RechtsTesten
tz(w)
REM Programmende

BisZurWand:
  s=0
  WHILE e5=1
    tv(200)
    s=s+ts
  WEND
  tz(8)
RETURN
```

Die Schildkröte fährt 10 Schritte zurück und zieht diese von der Gesamtschrittzahl w ab 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 (Sprungmarke "RechtsTesten:"). Sind beide Seiten zu, fährt sie weiter zurück wie in einer Sackgasse.
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 (Programmabschnitt "Zurück aus der Sackgasse"). 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 und zurücksetzen. Liegt die Ausfahrt rechts, wird sie diese sofort finden, da sie diese zuerst prüft. Bei einer Sackgasse rechts sucht die

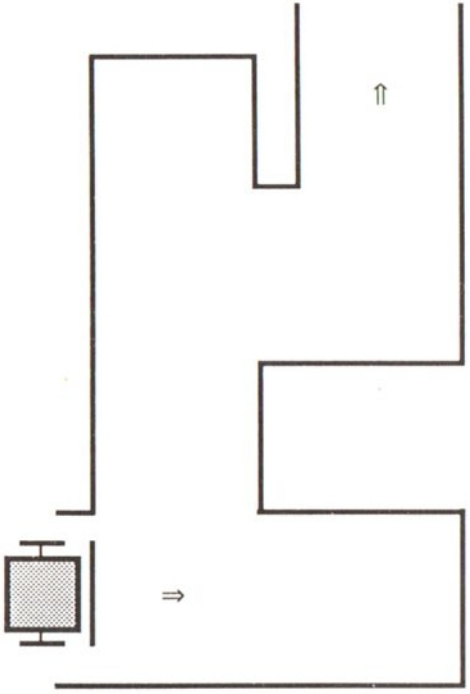


Bild 10.12: Schwieriges Labyrinth.

Schildkröte rückwärts nach einem Ausweg, 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. Die entsprechende Zeile wird vor dem Programmende eingefügt:

```
tl(180)
```

Damit Sie dann weitersucht, wird das gesamte Hauptprogramm in eine Endlosschleife eingebunden. Also:

```
REM Programmanfang
init
ti
Schleife:
  :
  tl(180)
GOTO Schleife
REM Programmende
```

Geben Sie die Zeile 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 (LABYRINTH).
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 weiterfahren 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 füh-

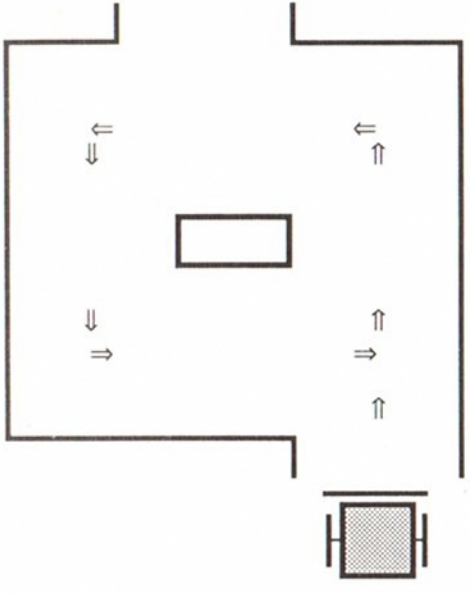


Bild 10.13: Bei diesem Labyrinth findet die Schildkröte nicht den oberen Ausgang.

ren 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 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:

```
REM Programmanfang
init
ti
Schleife:
  tv(10)
  IF e5=1 THEN GOTO Weiter
  schritte=ts
  tz(schritte)
  tl(90)
GOTO Schleife
Weiter:
  tr(90)
GOTO Schleife
REM Programmende
```

Den Zahlenwert 10 in der fünften Zeile 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

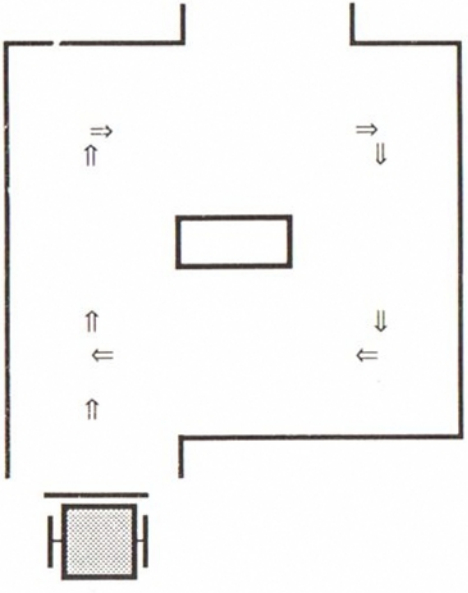


Bild 10.14: In diesem Labyrinth verirrt sich die Schildkröte.

befindet, muß gewährleistet sein, daß sie innerhalb der Zahl der Schritte dieses tv-Kommandos auch tatsächlich die rechte Wand findet.

Es steht Ihnen frei, das Programm mit einer Darstellung des Wegs der Schildkröte auf dem Bildschirm auszustatten. Das Programm STRATEGIE auf der Diskette macht dies ebenfalls.

Bleibt noch die Frage wie die Schildkröte auf dem kürzesten Wege durch ein Labyrinth 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

```
REM Programmanfang
init
PRINT ex
WHILE INKEY$=""
WEND
REM Programmende
```

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:

```
REM Programmanfang
init
ti
Schleife:
  IF ex<128 THEN CALL tv(1)
GOTO Schleife
REM Programmende
```

Geben Sie die Zeilen ein und starten das Programm mit **Start**. 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 die IF-Anweisung 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. Tauschen Sie

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.

das Bewegungskommando aus:

```
IF ex<128 THEN CALL tr(5)
```

Stellen Sie die Schildkröte so, daß sie ins Licht schaut, z.B. zum Fenster und starten das Programm mit **Start**. Sie dreht sich solange, bis es ihr dunkel genug ist. Umgekehrt geht's auch. Nach **Stop** geben Sie statt der bisherigen IF-Anweisung ein:

```
IF ex>128 THEN CALL tl(5)
```

Nach **Start** 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.
Dazu schreiben wir ein Programm, das die Schildkröte bei ausreichend Licht nach vorn fahren läßt, bei Dunkelheit aber eine Lichtsuche einschaltet. Die Grenze zwischen "hell" und "dunkel" wurde mit 128 festgelegt. Sie kann natürlich den speziellen Umgebungsbedingungen der Schildkröte angepaßt werden.

```
REM Programmanfang
init
Schleife:
  IF ex>128 THEN GOSUB suchen
     ELSE CALL tv(1)
GOTO Schleife
REM Programmende

Suchen:
  h=ex
  tr(5)
  Probieren:
    IF ex<h THEN RETURN
    tl(10)
    IF ex<h THEN RETURN
    tr(10)
  GOTO Probieren
```

Starten Sie das Programm mit **Start**. 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 (Nachsatz der ELSE-Anweisung).
Die Schildkröte bewegt sich solange nach vorn, bis die Lichtmessung einen Wert über 128 ergibt, es also dunkler wird. Nun kommt das Unterprogramm "Suchen" zum Zuge. Das Programm merkt sich die letzte Licht-

stärke und prüft zunächst nach rechts, ob es hier heller ist. Wenn das der Fall ist, also die Taschenlampe jetzt aus dieser Richtung strahlt, springt das Programm aus der Endlosschleife und beendet das Unterprogramm (IF-Anweisung). Ansonsten dreht sie sich zurück und prüft die Lichtstärke in der anderen Richtung. 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 **Stop**.
Damit die Schildkröte auch erkennt, wann die Taschenlampe ausgeschaltet wird oder die Schildkröte an ein Hindernis stößt, ergänzen wir das Hauptprogramm:

```
REM Programmanfang
init
ti
n=0
Schleife:
  IF ex>128 THEN GOSUB suchen
     ELSE CALL tv(1)
  erfolgreich=(n<5) AND (e5=1)
IF erfolgreich THEN GOTO
          Schleife
REM Programmende

Suchen:
  h=ex
  tr(5)
  FOR n=1 TO 5
    IF ex<h THEN RETURN
    tl(10)
    IF ex<h THEN RETURN
    tr(10)
  NEXT n
RETURN
```

Der Schleifenzähler n zählt die Suchvorgänge nach Licht. War die Suche in beiden Richtungen fünfmal erfolglos, wird die Variable erfolgreich auf logisch "falsch" gesetzt. Auch wenn die Schildkröte bei der Vorwärtsfahrt anstieß, wird erfolgreich auf logisch "falsch" gesetzt. Das Programm endet in diesen Fällen (IF-Anweisung).
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.

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:

```
REM Programmanfang
init
ti
h=1000
r=0
FOR w=0 TO 355 STEP 5
  hell=ex
  IF hell<h THEN h=hell : r=w
  PRINT "Winkel ";w;" : Hellig
     keit";hell
  tr(5)
NEXT w
REM Programmende
```

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 **Start**. Die Schildkröte dreht sich rechts herum und mißt bei jedem Drehschritt die Lichtstärke. Außerdem werden Winkelrichtung w und der Wert ex auf dem Bildschirm angezeigt 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 vor dem Programmende die folgende Zeile hinzu:

```
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. Anschließend drehen wir die Schild-

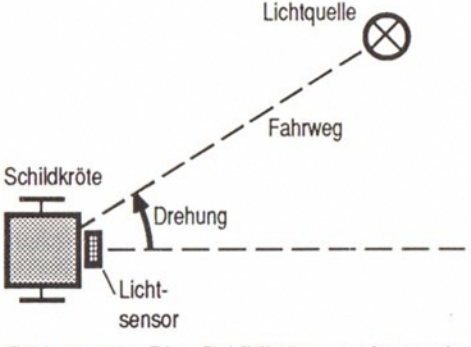


Bild 10.15: Die Schildkröte sucht nach Licht.

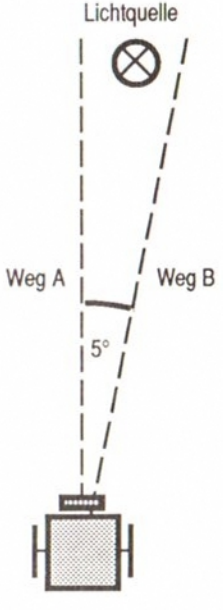


Bild 10.16: Winkelauflösung beim Messen.

kröte in die ermittelte Richtung. Beides läßt sich vereinen mit der Anweisung:

```
tl(360-r)
```

So findet die Schildkröte die richtige Richtung schneller. Wie im Kapitel 9 bereits gesagt wurde, versteht die Schildkröte keinen Drehwinkel von 360°. Deshalb wird obige Anweisung noch in eine IF-Abfrage gepackt und ebenfalls vor dem Programmende eingebaut.

```
IF r>0 THEN CALL tl(360-r)
tv(200)
```

Das letzte Kommando läßt die Schildkröte zur Lichtquelle fahren.
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 **Stop** 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. Das letzte Kommando, tv(200), wird durch folgendes Programmstück ersetzt:

```
Schleife:
  tv(20)
  tl(15)
  h=1000
  r=0
  FOR w=0 TO 30 STEP 5
    hell=ex
    IF hell<h THEN h=hell : r=w
    tr(5)
```

Bild 10.17: Richtungskorrektur bei der Anfahrt auf die Lichtquelle.

```
  NEXT w
  tl(30-r)
GOTO Schleife
REM Programmende
```

Nach **Start** 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,
5: Weiterfahren.

Der Merkvorgang für die Richtung mit der größten Helligkeit entspricht dem des ersten Programmstücks.
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:

```
IF e5=0 THEN GOTO Ende
```

nach dem Kommando tv(20) fährt sie nur noch bis zum "Hindernis" Lampe. Wenn die Stoßstange betätigt wird (E5=0), erfolgt ein Sprung aus der Wiederholschleife und damit ein Stop des Programms.
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. Dort wird am Bildschirm Aufbau und Ablauf des Versuchs "Schildkröte sucht nach Licht" demonstriert.

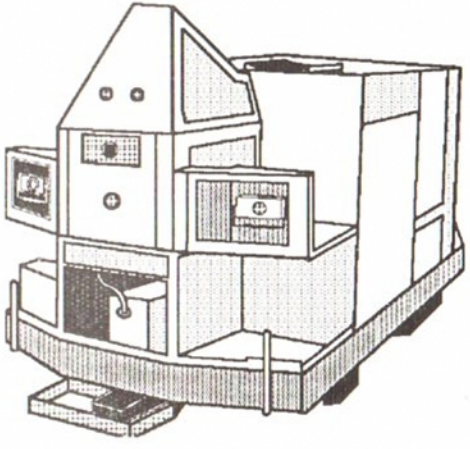


Bild 10.18: Industrieller Fahrroboter mit optischen und Ultraschall-Sensoren. Die Stoßstange und die Verkleidung sind teilweise entfernt. Die Lenkung dient nur dem manuellen Eingriff.

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:

```
REM Programmanfang
init
PRINT ex
WHILE INKEY$=""
WEND
REM Programmende
```

Nach Programmstart erscheint eine Zahl, die der Helligkeit der Fahrbahn vor der Schildkröte entspricht. Die Funktion ex liest den Wert ein, der mit PRINT 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-

Bild 10.19: Fahrspur für die Schildkröte.

Isolierband) als dunkle Spur. Damit läßt sich dann ein Weg markieren, den die 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:

```
REM Programmanfang
init
FOR i=1 TO 200
  m3r
NEXT i
ti
h=ex
WHILE ex>h-20
  tv(1)
WEND
m3a
REM Programmende
```

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 **Start**.
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 und sich diese gemerkt, also in h abgespeichert. Nun folgt die Wiederholschleife. Vor jedem Vorwärtsschritt wird erneut gemessen. Dieser Funktionswert von ex wird mit dem vorherigen in 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. Der Funktionswert 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!

Bild 10.20: Richtungskorrektur auf die Spur.

Bild 10.21: Abbiegung nach links bzw. rechts.

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. Dieser Teil wird vor dem Programmende, genauer: vor dem Kommando m3a, eingefügt. Geben Sie ein:

```
  tr(5)
IF ex>h-20 THEN
     GOTO NaechsterSchritt
tl(10)
IF ex>h-20 THEN
     GOTO NaechsterSchritt
```

Die Sprungmarke "NaechsterSchritt" befindet sich vor der WHILE-Anweisung, die die Geradeausfahrt steuert.
Setzen Sie die Schildkröte, wie in Bild 10.20 gezeigt, etwas schräg auf die Spur am Fahrbahnanfang, und starten Sie mit **Start**. Die Schildkröte wird nach kurzer Strecke die Bahn verlassen. Nun dreht sie nach rechts 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. 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 dreht sie sich wieder in die ursprüngliche Fahrrichtung und fährt zunächst 13 Schritte vor, damit ihre Drehachse über dem Bahnende steht. Dann dreht sie sich um 90° nach rechts und prüft anhand des Helligkeitsunterschieds, ob sich dort die Bahn befindet. Wenn ja, fährt sie weiter vorwärts. Ansonsten dreht sie sich in die andere Richtung und testet, ob es dort weitergeht. Wenn sie

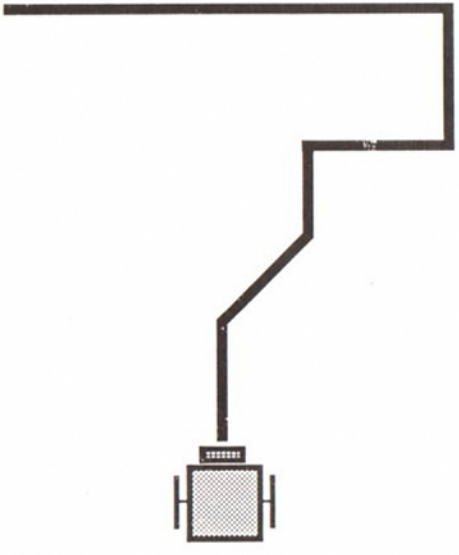


Bild 10.22: Fahrspur mit mehreren Abbiegungen.

keinen Weg findet, ist die Bahn zu Ende. Fügen Sie also ein weiteres Programmstück vor dem Kommando m3a ein:

```
tr(5)
tv(13)
tr(90)
IF ex>h-20 THEN
     GOTO NaechsterSchritt
tl(180)
IF ex>h-20 THEN
     GOTO NaechsterSchritt
PRINT "Bahnende!"
WHILE INKEY$=""
WEND
```

Setzen die Schildkröte wieder an den Bahnanfang und starten mit **Start**. Sie wird die Abbiegung erkennen, egal ob nach links oder rechts. Wenn Sie eine größere Strekke 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 zuletzt eingesetzte Programmstück ab:

```
tr(5)
tv(13)
tr(90)
FOR w=0 TO 180 STEP 5
  IF ex>h-20 THEN
       GOTO NaechsterSchritt
  tl(5)
NEXT w
IF ex>h-20 THEN
     GOTO NaechsterSchritt
PRINT "Bahnende!"
WHILE INKEY$=""
WEND
```

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 gleichmäßig helles Licht herrscht. Schon ein Schatten kann die Messung beeinflussen, so daß die Schildkröte vom Weg abkommt.
Wegführungen dieser Art werden Sie z.B. bei Flurförderfahrzeugen sehr oft finden.

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.

Meist werden sie jedoch wegen der höheren Störsicherheit durch Leitungen im Boden hergestellt. Der Roboter wird dann nicht durch Licht, sondern induktiv über ein im Boden verlegtes Kabel gesteuert.

Programme, die Fahrzeugsteuerungen nach Bildaufzeichnungssystemen ermöglichen, werden auch dem Begriff "Künstliche Intelligenz" zugeordnet. Dieser Begriff verursacht derzeit Aufregung.

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 "richtung" ein. War die letzte Kurskorrektur nach rechts, so wird richtung=0 gesetzt, bei links richtung=1. Wird wieder eine Kurskorrektur notwendig, soll das Programm bei richtung=0 zuerst rechts, bei richtung=1 zuerst links probieren.

Das Programm ist eine Erweiterung des vorangegangenen Programms; der Über-

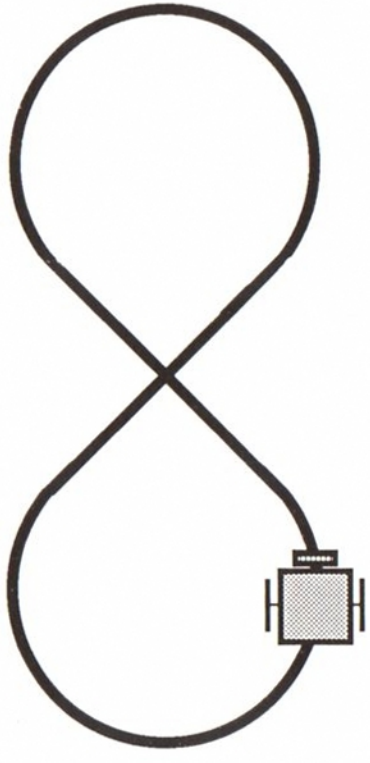


Bild 10.23: Achterschleife für die Schildkröte.

sicht wegen wird aber das ganze Programm abgedruckt:

```
REM Programmanfang
init
FOR i=1 TO 200
  m3r
NEXT i
ti
h=ex
richtung=0
NaechsterSchritt:
  WHILE ex>h-20
    tv(1)
  WEND
  IF richtung=1 THEN GOTO Links
  Rechts:
  richtung=1
  tr(5)
  IF ex>h-20 THEN
       GOTO NaechsterSchritt
  tl(10)
  IF ex>h-20 THEN
       GOTO NaechsterSchritt
  tr(5)
GOTO Abbiegung
  Links:
  richtung=0
  tl(5)
  IF ex>h-20 THEN
       GOTO NaechsterSchritt
  tr(10)
  IF ex>h-20 THEN
       GOTO NaechsterSchritt
  tl(5)
Abbiegung:
tv(13)
tr(90)
FOR w=0 TO 180 STEP 5
  IF ex>h-20 THEN GOTO
       Naechster Schritt
  tl(5)
NEXT w
IF ex>h-20 THEN
     GOTO NaechsterSchritt
m3a
PRINT "Bahnende!"
WHILE INKEY$=""
WEND
REM Programmende
```

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 richtung gerade umgedreht, also

```
richtung=1-richtung
```

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 und beinhaltet eine noch verbesserte Helligkeitsjustierung.

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 sind 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 ist und das Programm INIT geladen ist, setzen wir die

Schildkröte 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:

```
REM Programmanfang
init
FOR i=0 TO 200
  m3r
NEXT i
PRINT ex
m3a
WHILE INKEY$=""
WEND
REM Programmende
```

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. Starten Sie das Programm nochmals; jetzt ist die Zahl am Bildschirm wesentlich höher, da die Bahn dunkler ist als der Untergrund.
Damit 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:

```
REM Programmanfang
init
FOR i=0 TO 200
  m3r
NEXT i
hell=ex
Schleife:
  IF ex>hell+20 THEN z$=
    "binär 0" ELSE z$="binär 1"
  PRINT Z$
  WHILE INKEY$=""
    m3r
  WEND
GOTO Schleife
REM Programmende
```

Setzen Sie die Schildkröte mit dem Lesekopf auf eine helle Stelle und klicken Sie

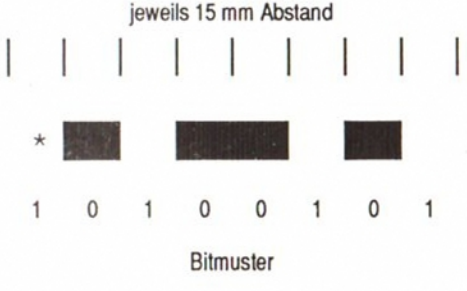


Bild 10.24: Informationscode auf der Fahrspur.

Start an. Der Meßwert, der einem hellen Untergrund entspricht, wird in der Variablen hell festgehalten. Danach folgt die Wiederholschleife. Befindet sich der Lesekopf mittlerweile auf einer dunklen Fläche, so liegt der Analogwert mindestens um 20 höher als der Wert in hell. Das entspricht dem Zustand "binär 0". Im anderen Fall wird der Zustand "binär 1" angenommen. Mit der PRINT-Anweisung erscheint die Information auf dem Bildschirm. Jetzt wartet das Programm auf einen Tastendruck (WHILE-Schleife), bevor die nächste Runde der Endlosschleife erfolgt. Setzen Sie die Schildkröte auf verschiedene Stellen und messen Sie 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 die Informationsbits auf die Fahrbahn kleben, soll noch etwas zu der Strichbreite gesagt werden. Die 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.
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. Vergleichen Sie:

```
REM Programmanfang
init
FOR i=0 TO 200
  m3r
NEXT i
hell=ex
ti
PRINT "Bitmuster: ";
Schleife:
  IF ex>hell+20 THEN z$=" 0"
```

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.

```
        ELSE z$=" 1"
  PRINT z$;
  tv(3)
GOTO Schleife
REM Programmende
```

Die Schildkröte steht zunächst mit dem Lesekopf auf der Startposition (in Bild 10.24 durch * gekennzeichnet). Wenn Sie das Programm 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 **Stop**-Funktion, 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:

```
REM Programmanfang
init
FOR i=0 TO 200
  m3r
NEXT i
hell=ex
mitte=hell+15
ti
SucheStartbit:
WHILE ex<mitte
  tv(1)
WEND
dunkel=ex
mitte=(dunkel+hell)/2
WHILE ex>=mitte
  tv(1)
```

```
WEND
hell=ex
REM Startbit identifiziert
tv(4)
code$=""
FOR i=0 TO 3
  mitte=(dunkel+hell)/2
  h=ex
  IF h<mitte THEN code$=code$
          +" 1" : hell=h
  IF h>=mitte THEN code$=code$
          +" 0" : dunkel=h
  tv(3)
NEXT i
PRINT "Code: ";code$
m3a
WHILE INKEY$=""
WEND
REM Programmende
```

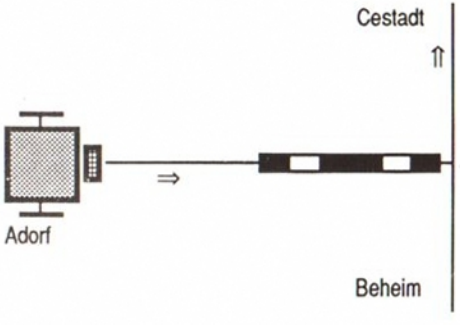


Bild 10.25: Autobahnnetz mit Verkehrsleitsystem.

Das Einlesen der Datenbits erfolgt jetzt in einer FOR...NEXT-Schleife. 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 **Start**. 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 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

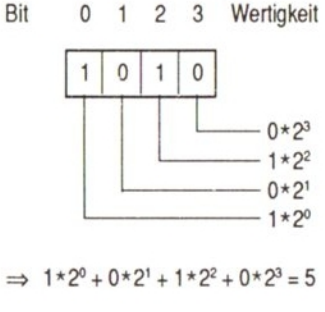


Bild 10.26: Bits im Informationscode.

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.

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. Die nachfolgenden Zeilen ersetzen die PRINT-Anweisung am Ende des Programms:

```
IF code$=" 0 0 0 0" THEN
     GOTO SucheStartbit
Rechtsab:
IF code$<>" 0 0 0 1" THEN
     GOTO Linksab
  tv(13)
  tr(90)
  GOTO SucheStartbit
Linksab:
IF code$<>" 0 0 1 0" THEN
     GOTO Ungueltig
  tv(13)
  tl(90)
  GOTO sucheStartbit
Ungültig:
IF code$=" 0 1 0 0" THEN
     GOTO Bahnende
  PRINT "Ungültigen Code
     gefunden."
  GOTO sucheStartbit
Bahnende:
PRINT "Bahnende!"
```

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 **Start**. Sie wird, wenn keine Datenübertragungsfehler auftreten, in Cestadt ankommen. Denken Sie wieder an eine gleichmäßige Beleuchtung und den richtigen Abstand des Lesekopfes von der Fahrbahn!

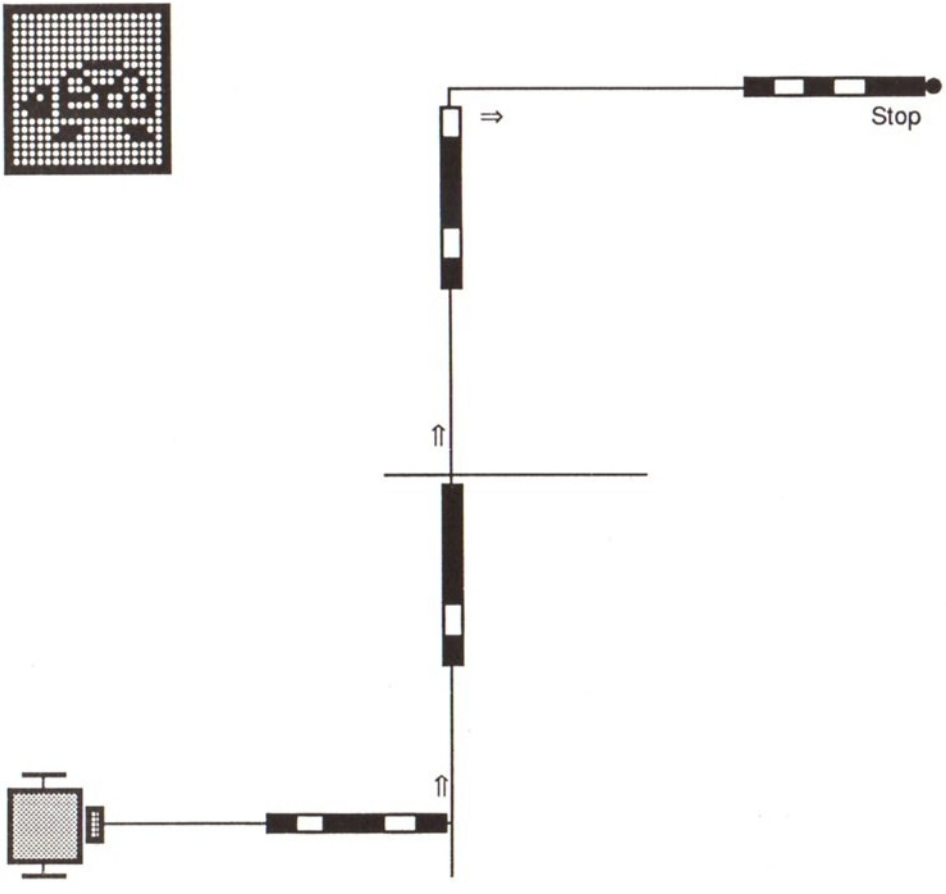


Bild 10.27: Die Schildkröte sollte auch eine längere Strecke meistern.

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. 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 ...

A1.1 Der Interfacetreiber

Jedes Programm des fischertechnik COMPUTING EXPERIMENTAL Baukastens, sei es ein Beispielprogramm der fischertechnik-Diskette oder ein Programm aus einem der vorangegangenen Kapitel, besteht aus zwei Teilen:

Dem Hauptteil des Programms; er enthält die gesamte logische Struktur der Aufgabe genau passend zu dem Aufbau des Experiments,

und den Anweisungen des Programms INIT; diese beinhalten eine Reihe von Deklarationen (Erklärungen) gegenüber dem AmigaBASIC-Interpreter, mit denen dieser sich zusätzliche Informationen verschafft, was in bestimmten Situationen zu veranlassen ist.

Die wichtigsten Anweisungen des Programms INIT sind all jene, die das Schlüsselwort LIBRARY enthalten. Die Library (zu deutsch: Bibliothek) stellt eine Sammlung von Unterprogrammen dar, mit denen die Funktionen des Interface komfortabel aufgerufen werden können. Diese Unterprogrammsammlung ist bei allen Programmen die gleiche Datei EXPER.LIBRARY aus der Schublade LIBS, d.h. es steht immer der gleiche Umfang von Kommandos zur Verfügung. Die Bibliothek EXPER.LIBRARY wird auch Interfacetreiber genannt, da ihre Kommandos das Interface "antreiben".

Der Interfacetreiber ist aus Gründen höherer Arbeitsgeschwindigkeit in der Maschinensprache des Mikroprozessors M68000 des Amiga verfaßt. Wenn Sie die Arbeitsweise des Interfacetreibers studieren wollen, so schlagen Sie im Kapitel "Funktionsweise des Interface und des Interface-Treiberprogramms" der Interfaceanleitung nach. Wer es ganz genau wissen will, weil er z.B. eigene Kommandos ergänzen will, muß die Quelltexte der Schublade SOURCE studieren. Er findet in der Schublade SOURCE auch die Datei EXPER_LIB.FD, die alle Unterprogramm-Einsprünge und die dazu benutzten Register des Mikroprozessors aufweist. Aus dieser Datei wurde mit Hilfe des Programms ConvertFD der EXTRAS-Diskette die Datei EXPER.BMAP erzeugt. Sie stellt die eigentliche Verbindung zwischen BASIC-Interpreter und Interfacetreiber her und befindet sich ebenfalls in der LIBS-Schublade.

Derartige Studien sind nur mit dem CLI (Command Line Interpreter) der WORKBENCH möglich. Die Vorgehensweise wird hier nicht näher erläutert; wer solche Unternehmungen plant, sollte sich im Betriebssystem des Amiga hinreichend gut auskennen.

Die Dateien EXPER.LIBRARY und

EXPER.BMAP werden mit der Anweisung

```
LIBRARY "EXPER.LIBRARY"
```

von der Diskette geladen. Deshalb muß die fischertechnik Diskette bzw. die Arbeitsdiskette bei Programmstart im Diskettenlaufwerk liegen.
Was im vorangegangenen Text als Interface-Kommando dargestellt wurde, sind Aufrufe der Unterprogramme des Interfacetreibers mit der CALL-Anweisung. Beachten Sie, daß das Schlüsselwort CALL in den meisten Fällen weggelassen werden kann, d.h. anstelle

```
CALL init
```

kann auch kurz und bündig geschrieben werden

```
init
```

Eine Ausnahme ist die Verwendung im Nachsatz einer IF-Anweisung. Nach dem Schlüsselwort THEN bzw. ELSE erwartet der BASIC-Interpreter entweder eine Sprungmarke oder eine Anweisung. Taucht kein Anweisungsschlüsselwort (wie CALL) auf, wird eine Sprungmarke vermutet. Die Suche nach der Sprungmarke bleibt jedoch erfolglos und verursacht deshalb eine Fehlermeldung. Deshalb:
Nach THEN bzw. ELSE immer das Schlüsselwort CALL den Interface-Kommandos hinzufügen.
Einige Unterprogramme des Interfacetreibers liefern dem rufenden BASIC-Programm Werte zurück; denken Sie z.B. an die Interface-Eingänge E1 bis E8, EX und EY. Diese Art von Unterprogrammen nennt man Funktionen. Funktionen können beliebig in Rechenausdrücken (d.h. auf der rechten Seite des Gleichheitszeichens) und in PRINT-Anweisungen verwendet werden. Die Funktionen müssen dem BASIC-Interpreter benannt werden, da sonst die Funktionen mit Variablen verwechselt werden würden. Dies bewirken die nachfolgenden Anweisungen des Programms INIT:

```
DECLARE FUNCTION e1& LIBRARY
      usw.
```

Das dem Funktionsnamen nachgestellte Und-Zeichen & weist die Funktion als eine vom Typ "lange Ganzzahl" aus.
Selbstverständlich müssen dem BASIC-In-

terpreter nur die Funktionen angezeigt werden, die auch tatsächlich benutzt werden. Um aber Flüchtigkeitsfehlern vorzubeugen, enthält der Vorspann des Programms INIT alle Funktionen des Interfacetreibers. Bei der Programmierung des Hauptteils des Programms braucht man also die Funktionserklärung nicht mehr zu beachten.

Wichtiger Hinweis:
Funktionswerte der Funktionen des Interfacetreibers dürfen nicht in anderen Kommandos oder Funktionen des Interfacetreibers verwendet werden!

Obwohl einer solchen Verwendung aus syntaktischen Gründen nichts entgegensteht und eine solche Kombination auch recht nützlich sein kann, wird das Betriebssystem des Amiga resp. der BASIC-Interpreter AmigaBASIC dadurch überfordert. Das Resultat ist ein Systemabsturz ("Guru-Meditation Error"), nach dem man nur wieder durch neues Anfahren des Computers weitermachen kann. Ein BASIC-Programm, das vorher nicht abgespeichert wurde, wäre danach verloren.

Um diese Situation zu vermeiden, weisen Sie den Funktionswert immer erst einer Variablen zu und verwenden die Variable im nachfolgenden Kommando- oder Funktionsaufruf. Beispiel:

```
gv(ex)                falsch!

hell=ex
gv(hell)              richtig.
```

Zum Ende des Programms INIT wird ein letztes Mal auf den Interfacetreiber Bezug genommen. Die Anweisung

```
LIBRARY CLOSE
```

meldet die Bibliothek EXPER.LIBRARY wieder ab.

A1.2 Textbildschirm

Die Bildschirmdarstellung des Amiga kann in verschiedene Betriebsarten geschaltet werden. Das Unterprogramm INIT trifft hier auch schon eine Vorentscheidung durch die Anweisungen

```
SCREEN 2,640,256,2,2
WINDOW 2,"EXPERIMENTAL",,0,2
```

Die Anweisungen definieren ein Bildschirmfenster, das den ganzen Bildschirm einnimmt und (in der gewählten "mittleren" Auflösungsstufe) 640 x 256 Bildpunkte besitzt, deren jeder eine von vier Farben annehmen kann. Das Bildschirmfenster ist mit der Titelleiste "EXPERIMENTAL" ausgezeichnet. Das Bildschirmfenster wird in den Beispielprogrammen zur textlichen Ein- und Ausgabe benutzt. Die Farben der Schriftzeichen und des Zeichenhintergrunds kann durch die COLOR-Anweisung gewählt werden. Solange keine speziellen Vereinbarungen getroffen wurden, gelten folgende Farbzuordnungen:

Farbe 0 (Hintergrund):	blau
Farbe 1 (Schriftstandard):	weiß
Farbe 2 :	schwarz
Farbe 3 :	orange

Die Farbzuordnung kann durch das PALETTE-Kommando auch beliebig anders geregelt sein, s. z.B. die Beispielprogramme der fischertechnik Diskette.

A1.3 Grafikbildschirm:

Wenn die Grafikschildkröte eingeschaltet wird erscheint ein ähnliches Fenster, jedoch ohne Titelleiste. In der Mitte des Bildschirms wird die Grafik-Schildkröte eingeblendet. Dieser Schirm kann noch in gleicher Weise zur Textausgabe benutzt werden. Zusätzlich sind aber die Schildkröten-Grafikkommandos möglich. Die Schildkröte benutzt ebenfalls die mit der COLOR-Anweisung zugeteilten Farben. Genauso wie bei der Textdarstellung können diese Farben durch die PALETTE-Anweisung auch beliebig neu verteilt werden.
Der Hintergrund des Grafikfensters ist, wenn nicht durch die COLOR- oder PALETTE-Anweisung anderes vereinbart wurde, blau (Farbe 0). Wird jedoch mit dem gload-Kommando ein Hintergrundbild geladen, so wird der Hintergrund des Grafikschirms transparent. Man sieht durch den Grafikschirm hindurch auf das geladene Hintergrundbild. Alle Zeichnungen der Grafik-Schildkröte, aber auch beliebiger anderer Zeichenanweisungen des BASIC-Interpreters (z.B. LINE) erfolgen auf die "Transparentfolie". Aufgrund dieser Eigenschaften ist das Hintergrundbild durch die üblichen Zeichenbefehle nicht zerstörbar und jedes Löschen des Bildschirms löscht nur den

Inhalt der Transparentfolie, erzeugt also wieder einen unversehrten Hintergrund. Dies ist zum Unterlegen von Skalen und dergleichen unter die Meßwerte eine große Erleichterung. Diese Eigenschaften können aber sicherlich in Ihren eigenen Programmen noch weitaus vielfältiger eingesetzt werden.

Wenn Sie eigene Experimente mit fischertechnik COMPUTING EXPERIMENTAL durchführen wollen, so werden Sie vielleicht den Wunsch haben, diese Experimente mit passend dazu gestalteten Hintergrundgrafiken zu hinterlegen. Um solche Hintergrundgrafiken zu erzeugen, wird die Grafik-Schildkröte nicht immer das passende Hilfsmittel sein. Sie werden sich vielleicht leistungsfähiger interaktiver Grafikprogramme bedienen wollen.
Die meisten Grafikprogramme geben Hinweise darüber, wie erzeugte Bilder im Rahmen von BASIC-Programmen geladen werden können. Laden Sie das Programm INIT. Geben Sie unter AmigaBASIC die Zeilen

```
init
ge(WINDOW(7))
```

ein. Fügen Sie die entsprechenden Zeilen wie im Malprogramm angegeben, hinzu. Fügen Sie dem Programm an der Stelle, wo das Bild bereits auf dem Bildschirm dargestellt ist, folgende Anweisungen hinzu:

```
Bild$="df0:MeinBeispiel"+CHR$(0)
gsave(SADD(Bild$))
```

Der Dateiname und der Datenpfad sind natürlich nur ein Beispiel und können frei gewählt werden. Durch das gsave-Kommando werden die Bilder im richtigen Datenformat abgespeichert und können von nun an als Hintergrundgrafiken benutzt werden.
Sofern die Bilder des Malprogramms im IFF-Format vorliegen (dies ist in den meisten Fällen so) und entweder in der mittleren Auflösung (640 x 256 Bildpunkte) oder der niedrigen Auflösung (320 x 256 Bildpunkte) mit zwei Farbebenen (= vier Farben) gezeichnet wurden, können sie direkt mit dem gload-Kommando übernommen werden.

Das wichtigste Stück Software der Diskette ist der Interfacetreiber. Er besteht aus den Dateien EXPER.LIBRARY und EXPER.BMAP. Beide sind in der Schublade LIBS enthalten.
Die Schublade SOURCE enthält die Quelltexte des Interfacetreibers und der damit verbundenen Dateien.
Die Schublade BILDER enthält die Hintergrundgrafiken von fischertechnik COMPUTING EXPERIMENTAL.
Die Schublade EXPERIMENTAL enthält die Beispielprogramme der fischertechnik COMPUTING EXPERIMENTAL Diskette. Zur Anfertigung der Arbeitsdiskette wird diese Schublade bis auf die Programme INIT und DIAGNOSE geleert, so daß die Programme des Experimentierhandbuchs Platz finden.
Die übrigen Dateiverzeichnisse, TRASHCAN, C, L, DEVS und S, enthalten Funktionen des Betriebssystems. Der Mülleimer (TRASHCAN) dürfte Ihnen bekannt sein. In ihm bewahrt man alle Dateien auf, die man zu löschen gedenkt. Aus Sicherheitsgründen haben wir im fischertechnik-Fenster den Mülleimer ein wenig versteckt, so daß ein Anfänger nicht unabsichtlich Unheil anrichten kann. Von besonderem Interesse

kann die Datei STARTUP-SEQUENCE aus dem Dateiverzeichnis S sein. In ihr steht die Vorschrift zum Anfahren der Diskette. Erfahrene Programmierer werden hier sicherlich noch das eine oder andere verändern oder hinzufügen können. Insbesondere kann man bei Amiga Computern mit ein Megabyte Speicher (oder mehr) empfehlen, durch die Anfahrvorschrift eine RAM-Disk einzurichten und den Interfacetreiber und ggf. auch die benötigten Hintergrundbilder in die RAM-Disk zu kopieren. Durch entsprechende Änderung der Zuweisungen muß dann die RAM-Disk als Datenquelle vereinbart werden:

```
ASSIGN Bilder: ram:Bilder
ASSIGN LIBS: ram:LIBS
```

Das Dateiverzeichnis T ist leer, es dient lediglich als Zeichen dafür, daß die Diskette bereits eingerichtet ist.
Die übrigen Verzeichnisse enthalten Betriebssystemfunktionen. Während des Durchlaufs der Stapeldatei EINRICHTEN werden Sie mit Dateien der Workbench gefüllt.
Die Datei LIESMICH ist eine Textdatei, die eine kurze Zusammenfassung der Installation der Software und der Experimente enthält. Außerdem kann diese Datei wichtige aktuelle Informationen enthalten, die in dem vorliegenden Experimentierhandbuch noch nicht aufgenommen werden konnten.
Das Programm FISCHER ist ein BASIC-Programm und enthält die gleichen bzw. noch darüber hinausgehende Informationen.

A1.6 Hinweise auf mögliche Fehlerursachen

An dieser Stelle tragen wir einige Hinweise auf Eigenheiten des Amiga Betriebssystems und typische Bedienungsfehler zusammen, die sich im Verlaufe der Benutzung der ersten Version als notwendig erwiesen haben.

Recht häufig wird (insbesondere mit 512K-Computern) die Fehlermeldung **Out of Heap-Space** gemeldet, dies oft gar schon beim Laden des AmigaBASIC Interpreters. In diesem Fall steht nicht ausreichend Speicherplatz zur Verfügung. Höchstwahrscheinlich stehen zuviele Dateien in der RAM-Disk oder Sie haben zuviele speicherresidente Hilfsprogramme geladen. Prüfen Sie die Speicherbelegung oder fahren Sie den Computer mit der fischertechnik-Diskette oder der Arbeitsdiskette neu an.

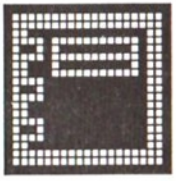
Generell sollte der Computer zum Experimentieren mit fischertechnik COMPUTING EXPERIMENTAL mit der fischertechnik Diskette oder der Arbeitsdiskette angefahren werden. Wenn Sie den Computer mit einer anderen Diskette angefahren haben und erst später die fischertechnik-Diskette oder die Arbeitsdiskette eingelegt haben, wird wegen fehlerhafter Zuordnungen jedes Programm, das den Interfacetreiber nutzt, mit der Fehlermeldung **file not found** anhalten. Die gleiche Fehlermeldung tritt auch auf, wenn Sie vergessen haben, den Vorsatz "copy of" des Diskettennamens zu löschen (s. Kap. 3).

Falls Sie Ihren Amiga-Computer nicht in Deutschland erworben haben, kann es sein, daß Ihre EXTRAS-Diskette nicht den Namen **ExtrasD**, sondern einen anderen Namen trägt. In diesem Fall bricht der Vorgang des Einrichtens ab, da nicht die korrekte Diskette gefunden werden kann. Ändern Sie in diesem Fall mit dem Texteditor der Workbench die Datei einrichten. Ersetzen Sie an allen vorkommenden Stellen **ExtrasD** durch die in Ihrem Fall richtige Bezeichnung. Prüfen Sie in diesem Fall auch, ob das Kommando

```
setmap D
```

der Datei startup-sequence die richtige Wahl für Ihre Tastatur ist.

In einigen Fällen ist es zu **fehlerhaften Funktionen der Schildkrötengrafik** gekommen. Schauen Sie in diesem Fall in dem LIBS-Ordner Ihrer Workbench nach der Datei mathtrans.library (sie ist für Funktionen wie Sinus und Cosinus verantwortlich). Die Versionsnummer der mathtrans.library darf nicht unter 33.5 liegen. Gegebenenfalls müssen Sie sich eine neuere Version der Workbench besorgen.



Sie werden feststellen, daß in der Übersicht der Interface-Kommandos mehr Kommandos aufgeführt sind, als in den Experimenten benutzt wurden. Diese Kommandos stellen Sonderfunktionen oder Zusammenfassungen anderer Kommandos dar und sind in erster Linie für den fortgeschrittenen Programmierer gedacht.

e1 Digital-Eingabe E1 bis E8
e2
e3
e4
e5
e6
e7
e8

Die Funktionen lesen die digitalen Eingänge E1 bis E8 ein. Der Funktionswert ist 0 für offen und 1 für mit +5V verbunden.

ex Analog-Eingabe EX
ey bzw. EY

Die Funktionen ermitteln den Widerstandswert, der an dem Eingang EX bzw. EY angeschlossen ist. Ein Widerstandswert von 0 Ω (direkt mit +5V verbunden) ergibt einen kleinen Zahlenwert (ca. 20), ein Widerstandswert von 5 kΩ einen großen Zahlenwert (ca. 200). Bei Widerstandswerten über 5 kΩ kann die Wandlungsdauer so lang werden, daß spürbare Verzögerungen des Programmablaufs auftreten und evt. sogar die Schutzschaltung aktiv wird.

g0 Grafikstift aus

Das Kommando schaltet den Stift der Grafik-Schildkröte ab. Bei den folgenden Kommandos gv oder gz wird keine Linie gezeichnet.

g1 Grafikstift ein

Das Kommando schaltet den Stift der Grafik-Schildkröte ein. Bei den folgenden Kommandos gv oder gz wird eine Linie in der gewählten Stiftfarbe gezeichnet. Dies ist der Anfangszustand nach dem ersten Kommando ge.

ga Grafik aus

Das Kommando schaltet die Schildkrötengrafik ab.

gc Grafik löschen

Das Kommando löscht das Bild des vorderen Grafikschirms. Der Hintergrundschirm ist entweder gelöscht (Hintergrundfarbe 0) oder durch das Kommando gload 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.

ge(zeiger) Grafik einschalten

Das Kommando schaltet die Schildkrötengrafik ein. Die Grafik-Schildkröte wird auf Ihren Startpunkt gesetzt. Um Bezug auf das aktuelle Bildschirmfenster zu nehmen, wird die Funktion WINDOW(7) abgefragt.

Sie liefert den Zeiger auf den Intuition Window Datensatz:

```
zeiger&=WINDOW(7)
CALL ge(zeiger&)
```

Die Anweisungen können zu einer zusammengefaßt werden:

```
CALL ge(WINDOW(7))
```

Alle weiteren Schildkröten-Grafikkommandos (mit Ausnahme gload und gsave) erfordern, daß zuvor das Kommando ge gegeben wurde.

gk **Grafik-Kurs**

Die Funktion gk ermittelt den derzeitigen Kurs der Grafik-Schildkröte. Der Kurs ist 0° bei Blickrichtung nach oben, 90° bei Blikkrichtung nach rechts, 180° bei Blickrichtung nach unten und 270° bei Blickrichtung nach links.

gl(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.

gload(zeiger) **Grafik laden**

Das Kommando lädt den Grafik-Hintergrundschirm von der Diskette. Der Verweis auf den Dateinamen erfolgt durch die Zeigervariable. Dazu wird der Dateinamme, ggf. mit Angabe des Datenpfades, in einer Textvariablen gespeichert. Die Textvariable wird mit einem Nullcode als Textendezeichen ergänzt.

```
Bild$="bilder:TURTLE.PIC"+CHR$(0)
```

Die Anfangsdresse der Textvariablen wird ermittelt und dem Aufruf als Zeiger mitgegeben:

```
zeiger&=SADD(Bild$)
CALL gload(zeiger&)
```

Die Anweisungen können zusammengefaßt werden.

go(nr,ri) **Schrittbefehl**

Der Befehl go ist eine Verallgemeinerung der Schrittbefehle m1v, m1z, usw. bis m4z. Motornummer und Richtung sind nicht mehr Bestandteil des Kommandonamens, sondern durch Parameter wählbar. Bei fortgeschrittener Programmierung kann dieser Befehl sehr nützlich sein, da die Betriebsbedingungen von vorangegangen Berechnungen abhängig gemnacht werden können.

Die Motornummer nr muß im Bereich 1 bis 4 liegen; die Richtungsangabe ri muß 170 für vorwärts und 85 für rückwärts sein.

gr(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.

gsave(zeiger) Grafik speichern
Das Kommando schreibt den Vordergrund des Grafikschirms auf die Diskette. Der Verweis auf den Dateinamen erfolgt durch die Zeigervariable. Dazu wird der Dateinamme, ggf. mit Angabe des Datenpfades, in einer Textvariablen gespeichert. Die Textvariable wird mit einem Nullcode als Textendezeichen ergänzt.

```
Bild$="bilder:TURTLE.PIC"+CHR$(0)
```

Die Anfangsdresse der Textvariablen wird ermittelt und dem Aufruf als Zeiger mitgegeben:

```
zeiger&=SADD(Bild$)
CALL gload(zeiger&)
```

Die Anweisungen können zusammengefaßt werden.
Das Bild kann mit dem gload-Kommando wieder geladen werden, dann allerdings in den Hintergrundschirm.

gv(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. Die Bildschirmabmessungen, gemessen vom Startpunkt der Grafik-Schildkröte betragen 128 Schritte nach oben, 127 Schritte nach unten, 159 Schritte nach rechts und 160 Schritte nach links. Schritte, die größer als ca. 200 sind, lassen daher 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 gezeichnet.

gx Grafik-Schildkröte X-Position
Die Funktion ermittelt die X-Position der Grafik-Schildkröte. Die Koordinate X ist 0 im Startpunkt der Grafik-Schildkröte, nach rechts positiv und nach links negativ. Die Zählung erfolgt in Bildpunkten und nicht in Schildkrötenschritten. Beispiel: Wenn sich die Schildkröte von ihrer Startposition mit 159 Schritten an den rechten Bildrand begeben hat, so liefert die Funktion gx den Wert 318.

gy **Grafik-Schildkröte Y-Position**

Die Funktion ermittelt die Y-Position der Grafik-Schildkröte. Die Koordinate Y ist 0 im Startpunkt der Grafik-Schildkröte, nach oben positiv und nach unten negativ. Die Zahl der Schildkrötenschritte und der Bildpunkte fällt zusammen. Beispiel: Wenn sich die Schildkröte von ihrer Startposition mit 128 Schritten an den oberen Bildrand begeben hat, so liefert die Funktion gy den Wert 128.

gz(s) **Grafik-Schildkröte zurück**

Das Kommando bewegt die Grafik-Schildkröte um s Schritte zurück. Weitere Details s. Kommando gv.

in(e) **Einlesefunktion**

Mit der Funktion in können alle Eingänge des Interface abgefragt werden. Der Parameter e verweist auf den Eingangskanal:

e=1	E1
e=2	E2
e=4	E3
e=8	E4
e=16	E5
e=32	E6
e=64	E7
e=128	E8
e=144	EY
e=160	EX

Der Funktionswert der Digitaleingänge ist 0 für offen und 1 für mit +5V verbunden. Zum Funktionswert der Analogeingänge s. Beschreibung der Funktionen ex und ey.

init **Initialisierung**

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

m1(ri) **Steuerung des Motors 1**

Das Kommando m1 steuert den Motor 1. Der Parameter ri gibt die Drehrichtung des Motors an:

Rechtslauf	85
Linkslauf	170
Motorstopp	255

m1a **Motor 1 ausschalten**

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

m1l **Motor 1 Linkslauf**
Das Kommando schaltet Motor 1 in Linkslauf. Es entspricht dem Kommando CALL M1(LINKS) der Interface-Anleitung.

m1r **Motor 1 Rechtslauf**
Das Kommando schaltet Motor 1 in Rechtslauf. Es entspricht dem Kommando CALL M1(RECHTS) der Interface-Anleitung.

m1v **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.

m1z **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.

m2(ri) **Steuerung des Motors 2**
Das Kommando m2 steuert den Motor 2. Der Parameter ri gibt die Drehrichtung des Motors an:

Rechtslauf	85
Linkslauf	170
Motorstopp	255

m2a **Motor 2 ausschalten**
Das Kommando schaltet Motor 2 ab. Es entspricht dem Kommando CALL M2(AUS) der Interface-Anleitung.

m2l **Motor 2 Linkslauf**
Das Kommando schaltet Motor 2 in Linkslauf. Es entspricht dem Kommando CALL M2(LINKS) der Interface-Anleitung.

m2r **Motor 2 Rechtslauf**
Das Kommando schaltet Motor 2 in Rechtslauf. Es entspricht dem Kommando CALL M2(RECHTS) der Interface-Anleitung.

m2v **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.

m2z **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.

m3(ri) **Steuerung des Motors 3**
Das Kommando m3 steuert den Motor 3. Der Parameter ri gibt die Drehrichtung des Motors an:

Rechtslauf	85
Linkslauf	170
Motorstopp	255

m3a **Motor 3 ausschalten**
Das Kommando schaltet Motor 3 ab. Es entspricht dem Kommando CALL M3(AUS) der Interface-Anleitung.

m3l **Motor 3 Linkslauf**
Das Kommando schaltet Motor 3 in Linkslauf. Es entspricht dem Kommando CALL M3(LINKS) der Interface-Anleitung.

m3r **Motor 3 Rechtslauf**
Das Kommando schaltet Motor 3 in Rechtslauf. Es entspricht dem Kommando CALL M3(RECHTS) der Interface-Anleitung.

m3v **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.

m3z **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.

m4(ri) **Steuerung des Motors 4**
Das Kommando m4 steuert den Motor 4. Der Parameter ri gibt die Drehrichtung des Motors an:

Rechtslauf	85
Linkslauf	170
Motorstopp	255

m4a **Motor 4 ausschalten**
Das Kommando schaltet Motor 4 ab. Es entspricht dem Kommando CALL M4(AUS) der Interface-Anleitung.

m4l **Motor 4 Linkslauf**
Das Kommando schaltet Motor 4 in Linkslauf. Es entspricht dem Kommando CALL M4(LINKS) der Interface-Anleitung.

m4r **Motor 4 Rechtslauf**
Das Kommando schaltet Motor 4 in Rechtslauf. Es entspricht dem Kommando CALL M4(RECHTS) der Interface-Anleitung.

m4v **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.

m4z **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.

motor(nr,ri) **Motorsteuerung**
Das Kommando motor steuert einen der Motoren M1 bis M4. Die Nummer des Motors wird durch den Parameter nr angegeben (Wertebereich 1 bis 4), die Drehrichtung durch den Parameter ri:

Rechtslauf	85
Linkslauf	170
Motorstopp	255

settb(b) **Schildkrötenbremse**
Das Kommando settb beeinflußt die Gegenstrombremse. Bei allen Schritt- und Schildkrötenkommandos m1v, m1z,...,tv, tz, tr, tl 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 settb eingestellt. Die Variable b ist ganzzahlig und liegt im Wertebereich1 bis 10. Das Kommando settb werden Sie normalerweise nicht benötigen, da der gewählte Fehlwert 4 bereits optimal eingestellt ist. Bei eigenen Modellen können jedoch abweichende Einstellungen je nach Lastverhältnissen besser sein.

tb **Schildkrötenbremse abfragen**
Die Funktion tb liefert den aktuell eingestellten Wert der Schildkrötenbremse (s. settb).

ti **Schildkröteninitialisierung**
Das Kommando ti 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.

tk **Schildkrötenkurs**
Die Funktion tk ermittelt den derzeitigen

Kurs der Schildkröte. Der Kurs ist 0° bei Blickrichtung in Startrichtung, 90° bei Blikkrichtung nach rechts, 180° bei Blickrichtung gegen die Startrichtung und 270° bei Blickrichtung nach links.

tl(d) 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. Nach Ausführung des Kommandos kann die Zahl der durchgeführten Schritte (nicht Winkelgrade!) mit der Funktion ts ermittelt werden. Der Eingang E5 wird - im Gegensatz zu dem Kommando tv - nicht geprüft.

tr(d) 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. Nach Ausführung des Kommandos kann die Zahl der durchgeführten Schritte (nicht Winkelgrade!) mit der Funktion ts ermittelt werden. Der Eingang E5 wird - im Gegensatz zu dem Kommando tv - nicht geprüft.

ts Schildkrötenschritte
Die Funktion ts ermittelt die Zahl der durchgeführten Schritte des letzten Schildkrötenkommandos. Bei Rückwärtsfahrt wird dies in aller Regel dem Wert des Parameters des tz-Kommandos entsprechen. Bei Drehungen wird die Schrittzahl, nicht der Drehwinkel, zurückgegeben. Bei Vorwärtsfahrt kann der Funktionswert von ts kleiner als der Parameterwert des tv-Kommandos sein, wenn die Schildkröte wegen eines Hindernisses ihre Fahrt nicht vollständig ausführen konnte.

tv(s) 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 tv prüft den Eingang E5 (bei der Schildkröte die Stoßstange). Ist E5 geöffnet (0), wird das Kommando abgebrochen. Bei Beendigung des Kommandos kann dies durch die Funktionen e5 und ts geprüft werden. Die Funktion e5 enthält den Zustand des Eingangs E5, die Funktion ts die Anzahl der erfolgreich ausgeführten Schritte. Die Wert von ts kann kleiner als s sein, wenn E5 vor Bahnende geöffnet wurde. Durch Kontrolle der Funktionen e5 und ts kann also ermittelt werden, ob und wann eine Kollision stattgefunden hat.

tx **Schildkröte X-Position**

Die Funktion ermittelt die X-Position der Schildkröte. Die Koordinate X ist 0 im Startpunkt der Schildkröte, nach rechts positiv und nach links negativ.

ty **Schildkröte Y-Position**

Die Funktion ermittelt die Y-Position der Schildkröte. Die Koordinate Y ist 0 im Startpunkt der Schildkröte, in Startrichtung positiv und gegen die Startrichtung negativ.

tz(s) **Schildkröte zurück**

Das Kommando bewegt die Schildkröte um s Schritte zurück. Nach Ausführung des Kommandos kann die Zahl der durchgeführten Schritte mit der Funktion ts ermittelt werden. Der Eingang E5 wird - im Gegensatz zu dem Kommando tv - nicht geprüft.

Bildnachweis

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

Bleichert Förderanlagen GmbH, Osterburken, S. 83 und 97,

Valvo Unternehmensbereich Bauelemente der Philips GmbH, Hamburg, S. 44,

Wagner Fördertechnik GmbH&Co. KG, Reutlingen, S. 116.

Art.-Nr: 39512 · 0389 · Printed in Germany · Technische Änderungen vorbehalten · RS