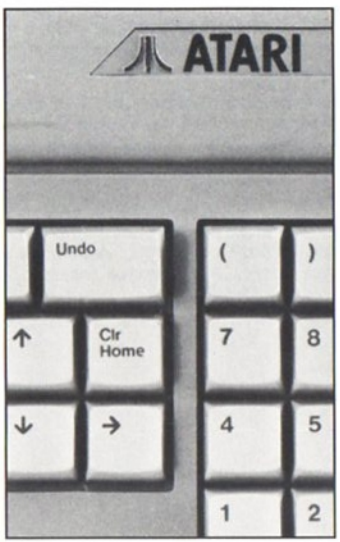
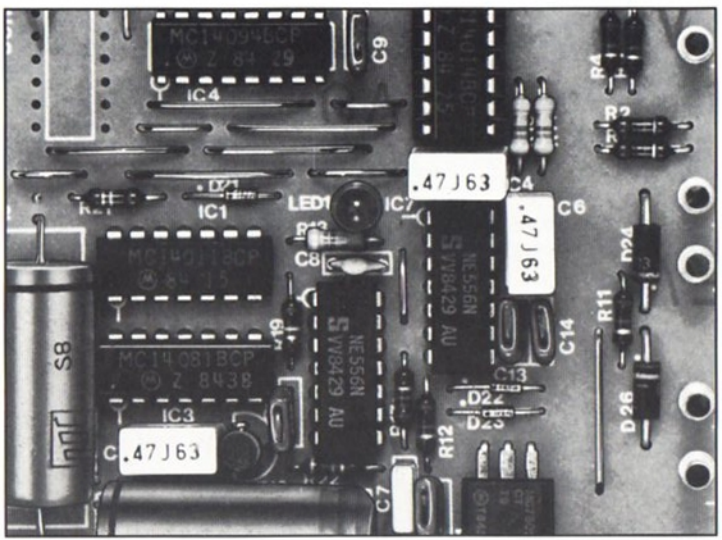
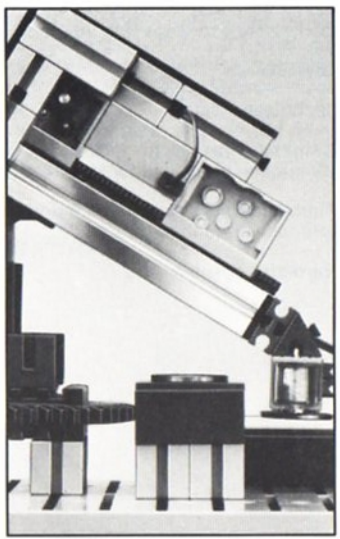
fischertechnik® COMPUTING

Interface für Atari ST Computer







Inhalt

Table of Contents

fischertechnik computing Interface

Lieber fischertechnik-Freund,

um mit einem Computer, in Erweiterung seiner Einsatzmöglichkeiten, auch technische Modelle ansteuern zu können, wurde fischertechnik computing entwickelt. Hierzu gehören sowohl der fischertechnik computing Baukasten und die fischertechnik computing Bausätze ebenso wie die fischertechnik computing Interfaces und die Software. Es ist jetzt möglich, technische Funktionen und Vorgänge zu simulieren, Aufgaben zu lösen und einfach viel Spaß an computergesteuerten Modellen zu haben.

Was braucht man zum Steuern der Modelle? Zunächst einmal das fischertechnik Modell zur Ausführung der Abläufe. Dann einen Personalcomputer, wie Sie ihn besitzen. Er dient der Steuerung und der Koordination. Und dann noch ein Interface als Bindeglied zwischen beiden.
Was Sie in den Händen halten, ist das fischertechnik computing Interface. Steuersignale, die von dem Computer kommen, z.B. „Motor einschalten!", werden von dem Interface in kräftige Ströme umgesetzt, die in der Lage sind, tatsächlich einen Motor zu bewegen. Wir sprechen in diesem Fall von einer Ausgabe. Die gedachte Blickrichtung verläuft von dem Computer nach außen. Aber auch der umgekehrte Weg ist denkbar und kommt vor. Die Modelle besitzen Taster, Potentiometer etc., um dem Computer Bericht zu erstatten, was an dem Modell draußen vorgeht. Auch hier greift das Interface wieder helfend ein und bereitet diese Signale dergestalt auf, daß sie eine für den Computer verständliche Eingabe darstellen.

Das fischertechnik Interface besitzt nun folgende Leistungsmerkmale:

- Mit ihm lassen sich vier fischertechnik Motoren, Lampen, Elektromagnete etc. steuern.
- Mit ihm kann man acht Taster oder Schalter abfragen.
- Darüber hinaus liefern zwei Eingänge die Werte von stufenlosen Signalgebern wie etwa Potentiometern.

Doch was würden alle elektrischen Verbindungen zwischen Computer und fischertechnik Modell mit Hilfe des Interface nutzen, wenn Sie keine Hilfsmittel hätten, jene zu aktivieren. Die Rede ist von der Software. Dieser Teil liegt in der Form einer Diskette vor. Auf ihr befindet sich ein Programm, das den Sprachschatz Ihres Computers derart erweitert, daß die Steuerung über das Interface tatsächlich erfolgen kann. Dieses Programm wird die Keimzelle Ihrer eigenen Programme sein. Doch damit nicht genug: Damit Sie den Einsatz dieser neuen Hilfsmittel studieren und lernen können, sind Beispielprogramme für alle fischertechnik computing Modelle auch noch untergebracht.
Sie sehen, es wartet eine ganze Menge von interessanten Aufgaben auf Sie. Ich wünsche Ihnen viel Spaß dabei. Ihr

Anschluß des Interface

Das fischertechnik Interface für Centronics Port wird an Paralleldrucker-Schnittstellen nach Centronics-Norm angeschlossen. Eine solche Schnittstelle ist auch an den Computern der Atari ST Reihe vorhanden: Atari 260 ST, Atari 520 ST, Atari 1040 ST sowie Atari Mega ST. Allerdings ist das Interface kein Drucker. Ein Drucker kann nur eine Ausgabe bewirken, d.h. die vom Computer gelieferten Signale so auswerten und umsetzen, daß ein Text auf dem Papier erscheint. Das fischertechnik Interface kann mehr: es kann Signale ausgeben, z.B. an den Antriebsmotor eines Roboters. Es kann aber auch Signale einlesen, z.B. die Menge des auf einen optischen Sensor einfallenden Lichtes registrieren. Dies ist, vereinfacht gesagt, der Grund, warum das fischertechnik Interface nicht wie ein Drucker (z.B. mit LPRINT) angesprochen werden kann. Vielmehr muß ein spezieller Interface-Treiber in die Software eingebunden werden. Der Interface-Treiber selbst ist wiederum ein Stück Software, das in der Maschinensprache des Mikroprozessors des Atari ST (Motorola 68000) geschrieben ist. Die Einbindung geschieht bei der mitgelieferten fischertechnik computing Software unter dem Betriebssystem TOS mit der Benutzeroberfläche GEM in die Programmiersprache GfA-BASIC. Darüber wird nachfolgend noch mehr gesagt.

Doch zurück zur Hardware. Das fischertechnik Interface wird, wie bereits oben erwähnt, an den Druckerausgang des Atari ST angeschlossen. Hierzu gehen Sie folgendermaßen vor:

- Vergewissern Sie sich, daß der Computer abgeschaltet ist.
- Drehen Sie den Computer herum, so daß die rückseitigen Anschlußstecker vor Ihnen liegen. Suchen Sie den Parallel-Druckeranschluß. Es muß ein 25poliger Buchsenstecker sein. Sie werden den Parallel-Druckeranschluß mit anderen 25poligen Schnittstellen, z.B. der Seriell-Schnittstelle, nicht verwechseln können, da diese Stifte und keine Buchsen besitzen.
- Durch Reibungselektrizität können Sie, ohne es zu merken und ohne daß es für Sie schädlich wäre, auf mehrere 1000 Volt aufgeladen sein. Diese Spannung ist jedoch schädlich für die Schaltkreise in dem Interface und dem Computer. Entladen Sie daher eine eventuell vorhandene elektrostatische Aufladung durch Berühren eines geerdeten Gegenstandes, z.B. eines Wasserhahns, eines Türgriffs oder einer Heizung.
- Legen Sie sich nun das Interface zurecht.
- Mit dieser Anleitung und der Diskette zum fischertechnik computing Modellbaukasten haben Sie auch den Steckadapter passend zu dem Druckeranschluß des Atari ST geschickt bekommen. Schließen Sie den Steckadapter an das Interface-Anschlußkabel an. Ein Verdrehen des Anschlußsteckers ist ausgeschlossen, da der Steckadapter eine Aussparung für die Nase des Steckers besitzt. Der Steckadapter wird auf den Parallel-Druckeranschluß des Atari ST gesteckt. Auch hier ist ein Verpolen wegen der Trapezform des Steckers nicht möglich.
- Schließen Sie das Interface an das Netzgerät an. Das Interface erwartet ungesiebte Gleichspannung zwischen 6 und 10 Volt. Beim Einsatz von stabilisierter Gleichspannung sind ca. 8 Volt (belastbar bis ca. 2 Ampere) einzustellen. Um ganz sicher zu gehen, sollten Sie jedoch das fischertechnik computing Netzgerät verwenden. Stecken Sie den Kabelausgang des Netzgeräts mit dem roten Flachstecker in die mit + gekennzeichnete Buchse des Interface und den Kabelausgang mit grünem Flachstecker in die mit − gekennzeichnete Buchse. Sie haben dabei die Auswahl zwischen zwei Buchsenpaaren; welches Sie verwenden, ist gleichgültig.
- Für Modelle mit bis zu zwei Motoren können auch die fischertechnik Netzgeräte mot4 eingesetzt werden, also insbesondere für die Modelle des fischertechnik computing Baukastens. Sollten Sie aber eigene Konstruktionen mit einer größeren Motorzahl oder die Bausätze Trainingsroboter oder Plotter/Scanner betreiben wollen, brauchen Sie Verstärkung. Entweder wechseln Sie dann doch zu dem kräftigeren fischertechnik computing Netzteil oder Sie speisen mit einem zweiten fischertechnik Netzgerät mot4 in die noch freien Anschlußbuchsen ein.
- Bei Einsatz der Netzgeräte mot4 verwenden Sie die den Baukästen beigelegte zweiadrige Leitung und Flachstecker zur Herstellung des Anschlußkabels.
- Verbinden Sie das fischertechnik computing Modell mit dem Interface. Hierzu dient das den Bausätzen und dem Baukasten fischertechnik computing beigefügte zwanzigadrige Flachbandkabel. Dieses Kabel ist auch als Einzelteil aus dem Service-Set fischertechnik erhältlich.
- Die Reihenfolge, in der Sie das Interface und den Computer nun einschalten, spielt keine Rolle. Wenn Sie das Interface mal nicht benutzen und mit anderen Porgrammen arbeiten, brauchen Sie das Interface dennoch nicht abkabeln. Lassen Sie in diesem Fall einfach das Interface ausgeschaltet.
- Nebeneffekte des Interface: Solange das Interface eingesteckt ist, können Sie selbstverständlich keinen Drucker benutzen, der auch an diesem Anschluß eingesteckt wird. Allerdings sind im Handel Umschalter erhältlich, mit denen Sie manuell zwischen mehreren Druckern umschalten können. Einen solchen Umschalter können Sie auch zwischen den Atari ST und das fischertechnik Interface schalten. Damit ersparen Sie sich, jedesmal die Anschlußkabel von Drucker und Interface auszutauschen.

Auch beim Arbeiten mit den Modellen sollten Sie sicherheitshalber immer zuerst eine eventuell vorhandene elektrostatische Aufladung ableiten, indem Sie einen geerdeten metallischen Gegenstand berühren. Doch nun genug von der Hardware,

fischertechnik computing Software

im nächsten Abschnitt wollen wir uns die fischertechnik computing Software vornehmen.

Wer sich schon einmal mit dem Gedanken befaßt hat, irgendwelche Geräte oder Modelle mit dem Computer zu steuern, wird aus eigener Erfahrung wissen oder von anderen Computerfreunden gehört haben, daß dies alles gar nicht so einfach sei. Man brauche eine genaue Kenntnis des Computers, des Mikroprozessors und der Ein- und Ausgabebausteine sowie der Maschinensprache für diese Aufgabe. Bislang stimmte diese Aussage, und dadurch wurde leider auch mancher von diesem interessanten Kapitel der Computerei abgehalten. Jetzt gibt es diese Schwierigkeit nicht mehr. In dem Lieferumfang des Interface sind die vorliegende Anleitung und Programme auf Diskette enthalten.

Um sicher zu sein, daß Sie nicht bei einem versehentlichen Löschen oder einer Beschädigung der Diskette ohne Software dastehen, sollten Sie sich von der fischertechnik computing Diskette eine Kopie anfertigen. Verfahren Sie hierzu genauso, wie in dem Handbuch Ihres Atari Computers die Anfertigung einer Sicherungskopie der Language Disk beschrieben ist. D.h. formatieren Sie zunächst eine einseitige Diskette (360 K). Wenn Sie schon dabei sind, formatieren Sie noch einige zusätzliche Disketten: Einige fischertechnik computing Programme legen Daten auf Diskette ab; da die fischertechnik computing Diskette nahezu vollständig gefüllt ist, benötigen Sie formatierte Leerdisketten.

Im nächsten Schritt legen Sie die fischertechnik Originaldiskette in das Laufwerk A ein. Wenn Sie zwei Diskettenlaufwerke angeschlossen haben, legen Sie die leere formatierte Diskette in Laufwerk B ein. Ziehen Sie mit dem Mauszeiger das Symbol des Laufwerks A herüber auf das Symbol des Laufwerks B und lassen es los. Darauf erscheint ein Hinweisfenster, das meldet, daß die Anfertigung der Kopie alle Daten auf Diskette B überschreibt. Das nächste Fenster gehört zu dem DISKCOPY Programm. Bestätigen Sie nochmals Ihre Absicht die Diskette zu kopieren. Danach läuft der Kopiervorgang an. Wenn Sie nur ein Diskettenlaufwerk besitzen, wird das Programm immer vermelden, wann die Original- und wann die Kopiediskette in das Laufwerk A einzulegen sind. Wenn der Kopiervorgang abgeschlossen ist, erscheint wieder das DISKCOPY Fenster. Sie können nun eine weitere Kopie anfertigen oder den Programmpunkt EXIT zum Verlassen des Programms anwählen. Beschriften Sie die Kopiediskette und verstauen Sie die fischertechnik computing Diskette an einem sicheren Platz, an den keine natürlichen Feinde der Disketten, wie Sand, Hitze, Katzen oder Magnetfelder hinkommen können. Arbeiten Sie fortan mit der Kopie und benutzen Sie die fischertechnik computing Diskette nur, um gegebenenfalls eine weitere Kopie zu ziehen. Kopien sind vor allen Dingen dann nützlich, wenn Sie die Programme noch abändern wollen und die Originaldiskette bestehen bleiben soll. Das Kopieren beschränkt sich jedoch wohlgemerkt nur auf den eigenen Gebrauch.

Wenn Sie das Interface noch nicht an Ihren Computer wie in dem vorigen Kapitel beschrieben angeschlossen haben, so sollten Sie es nun nachholen. Die fischertechnik computing Programme sind BASIC-Programme. Sie laufen unter dem BASIC-Interpreter der Firma GfA Systemtechnik. Dieser Interpreter verbindet eine relativ hohe Fehlerfreiheit mit einer hohen Arbeitsgeschwindigkeit, geringem Speicherbedarf und hohem Bedienkomfort.

Wenn Sie das Softwarepaket GfA-BASIC besitzen, so laden Sie nun GfA-BASIC. Legen Sie die GfA-Systemdiskette in das Diskettenlaufwerk A ein und schalten Sie, wenn noch nicht erfolgt, Bildschirm, evtl. das zweite Laufwerk und den Computer in dieser Reihenfolge ein. Wählen Sie mit der Maus das Laufwerk A an. Das Diskettenfenster zeigt unter anderem die Datei GFABASIC.PRG; wählen Sie dieses an. Anschließend erscheint der Editor von GfA-BASIC. Legen Sie nun die fischertechnik computing Diskette ein und wählen Sie den Menüpunkt LOAD. Es erscheint nun das Auswahlmenü der fischertechnik computing Diskette.

An dieser Stelle wollen wir einflechten, wie das Starten abläuft, wenn Sie kein GfA-BASIC besitzen. Legen Sie in diesem Fall gleich zu Beginn die fischertechnik computing Diskette ein und schalten Sie, wie oben beschrieben, Ihre Computeranlage ein. Wählen Sie das Symbol für Laufwerk A an. Das Auswahlfenster zeigt Ihnen nun die Datei GFABASRO.PRG zusammen mit den beiden Ordnern DEUTSCH und ENGLISH und einigen Hilfsprogrammen. Wählen Sie GFABASRO.PRG an. GFABASRO steht für GFABASIC run only Interpreter. Unter diesem Interpreter können Programme nur ausgeführt, nicht jedoch geschrieben oder geändert werden, da er keinen Editor besitzt. Dafür gestattet uns die Firma GfA Systemtechnik, den Interpreter lizenzfrei auf die fischertechnik computing Diskette zu kopieren. Im Gegensatz zu dem vollständigen GfA-BASIC Interpreter springt daher der run only Interpreter gleich in das Programmauswahlmenue. Die Auswahl fertiger Programme ist dann unter beiden Interpretern gleich, auch der Ablauf der Programme.
Da Sie als fischertechniker jedoch sich nicht damit begnügen werden, fertige Programme laufen zu lassen, sondern sicherlich mit fischertechnik und dem Computer experimentieren wollen, können wir die Anschaffung des GfA-BASIC Interpreters nur empfehlen.
Unter beiden Interpretern (dem vollständigen wie auch dem run-only-Interpreter) ist der folgende Ablauf der gleiche. Wählen Sie den Ordner DEUTSCH an, worauf das Inhaltsverzeichnis dieses Ordners erscheint. Das Inhaltsverzeichnis ist alphabetisch geordnet, unter den ersten Einträgen finden Sie das Programm FISCHER.BAS. Wählen Sie dieses aus. Unter dem run-only-Interpreter startet das Programm sofort. Unter dem vollständigen Interpreter müssen Sie noch das Kommando RUN geben (Shift-Taste festhalten und F10 drücken oder mit der Maus das Kommando RUN in der Kommandoleiste anklicken). Das Programm wird einen Begrüßungstext auf dem Bildschirm erscheinen lassen, den Sie sorgfältig studieren sollten. In diesem Text kann noch mancher nützliche Hinweis enthalten sein, der das spezielle Softwarepaket für den jeweiligen fischertechnik computing Bausatz betrifft.
Am Schluß des Begrüßungsprogramms steht Ihnen erneut die Auswahl der Programme zur Verfügung. Wählen Sie jetzt das sogenannte Grundprogramm, Datei GRUNDPR.BAS. Nach dem Starten des Grundprogramms (automatisch oder per Kommando RUN) wird das Diskettenlaufwerk nochmals kurz anlaufen, denn das Grundprogramm lädt das Maschinenprogramm INTERFAC.COM. Jenes enthält die oben erwähnten Detailkenntnisse über die Ein- und Ausgabebausteine des Computers. Anschließend erscheint auf dem Bildschirm die Titelmeldung von fischertechnik computing und gleich darauf wird das Programmende vermeldet.
Äußerlich scheint sich nichts geändert zu haben. Dennoch besitzt Ihr Atari ST nun einige neue Befehle, die vorher in dieser Form nicht im BASIC enthalten waren. Diese Befehle sind genau auf Ihren Atari ST und das fischertechnik computing Interface abgestimmt. Die Befehle benutzen das C:-Kommando. Das Grundprogramm selbst ist in BASIC geschrieben und erzeugt im Variablenspeicher des Atari ST die Parameter jener Maschinenprogrammaufrufe.
Sie brauchen daher nur noch die folgend beschriebenen Befehle zu beherrschen:

Der Motorausgang M1 wird angesteuert mit:

Dummy=C:M1(Ein) **Dummy=C:M1(Aus)**
Dummy=C:M1(Rechts) **Dummy=C:M1(Links)**

Die Variable Dummy wird hier notwendig, weil C: eine Funktion ist und einen Wert zurückliefert, der dann in einer Zuweisung aufgebraucht werden muß. Der eigentliche Zweck der Funktion ist jedoch die Steuerung des Motors, der zurückgegebene Wert selbst ist unerheblich. Infolgedessen wird er einer Variablen zugewiesen, deren Wert im Programm nicht mehr weiter verwendet wird. Für solche Variablen hat sich der Begriff „Dummy" eingebürgert (englisch: Strohmann, Attrappe).
Die Funktionsparameter bezeichnen den Motor, in diesem Fall Motor 1, und die Betriebsart.
Die entsprechenden Befehle mit M2, M3 und M4 steuern die übrigen drei Ausgänge. Außerdem sollten Sie sich merken, daß „Ein" ebenfalls immer Rechtslauf bewirkt.

Die 10 Eingänge des Interface werden ebenfalls mit Hilfe der C:-Funktion eingelesen. Die Funktion

C:In(E1)

ist 1, wenn der Eingang E1 des Interface mit +5 V verbunden ist. Sonst zeigt C:IN(E1) den Wert 0. Entsprechend erhält man mit C:In(E2)...C:In(E8) die Zustände der übrigen Digitaleingänge.
Die Analogeingänge EX und EY werden über je ein Potentiometer (4,7 kΩ) mit +5 V verbunden. Die Funktionen

C:In(Ex)
C:In(Ey)

führen dann einen Wert aus dem Bereich 20 bis 280, je nach Stellung der Potentiometer. Obige Grenzwerte können infolge von Exemplarstreuungen bei den Interfaces auch leicht anders ausfallen.
Die Analogeingänge dienen häufig der Positionsmessung. Wird. z.B. ein Roboterarm von einem Motor angetrieben und synchron mit der Bewegung des Arms das Potentiometer EX verstellt, so kann das Programm, indem es immer wieder die Funktion

C:In(Ex)

aufruft, die Bewegung des Roboters genau verfolgen.
Wenn kein Potentiometer an den Eingang EX bzw. EY angeschlossen ist, wird sich ein Überlauf des internen Auswertezählers ergeben. Der Funktionswert beträgt dann ca. 5000.

Der letzte der neuen Befehle ist

CALL Init

Dieser wird benutzt, um das Interface in einen wohldefinierten Anfangszustand zu versetzen. Er kann auch benutzt werden, wenn alle Motorkanäle mit einem Male abgeschaltet werden sollen.
Doch nun genug der langen Vorrede. Wenn Sie den vollständigen GfA-BASIC Interpreter besitzen, können sie die genannten Kommandos im Direktmodus ausprobieren. Den Direktmodus erhalten Sie, indem Sie das Kommando Direct in der Kommandoleiste anwählen oder einfach die Esc-Taste drücken. Schließen Sie einen fischertechnik Motor über das zwanzigpolige farbcodierte Flachbandkabel an M1 an. Dies sind die gelbe und orange Leitung in der oberen Hälfte des Flachbandkabels. Geben Sie ein:

Dummy=C:M1(Ein)

Der Motor wird kurz anlaufen und dann wieder stehenbleiben. Genießen Sie diesen Augenblick, er hat Ihnen das Gefühl gegeben, in kurzer Zeit die kompliziertesten fischertechnik Anlagen mit Ihrem Atari ST zu steuern.
Doch zunächst interessiert uns auch die Frage, wieso der Motor wieder stehenblieb. Hatten wir ihn nicht eingeschaltet? Gibt es zum Ausschalten nicht, wie oben beschrieben, einen eigenen Befehl? Nun, der Motor ist zwar stehengeblieben, aber in dem Interface ist nach wie vor gespeichert, daß er eigentlich laufen sollte. Das Interface hat sich selbst „schlafen gelegt", Dies tut es immer, wenn innerhalb einer halben Sekunde kein neuer Befehl kommt. Es geschieht aus Sicherheitsgründen. Stellen Sie sich vor, Sie erproben ein neues Programm. Die Wahrscheinlichkeit, daß noch irgendwo ein Fehler im Programm versteckt ist, grenzt an Gewißheit. Der Computer bleibt mit einer leidigen Meldung wie

STOP Division durch Null

stehen. Der Motor, der kurz vorher eingeschaltet wurde, bliebe jedoch nicht stehen und schicke sich an, das schöne Modell zu demolieren. Sie müßten zum Netzgerät hasten und schnell die Spannung abstellen.
Wie beruhigend ist es da zu wissen, daß der Motor von alleine stehenbleiben wird. Auch dann, wenn Sie mit dem Tastendruck Control-Shift-Alternate den Programmablauf unterbrechen. Und wenn es wieder weitergeht, so wird mit dem ersten Befehl das Interface wieder „aufgeweckt" und hat keinen der Motoren vergessen. Der Ablauf kann fortgesetzt werden, als sei nichts geschehen.
Daß das Interface mit dem Abschalten nicht sofort zur Hand ist, wurde mit Bedacht gewählt. Zwischen den Ein- und Ausgabebefehlen an das Interface werden sich immer wieder Pausen aufgrund von Berechnungen ergeben, die es zu überbrücken gilt. Ob das Interface durch Ein- oder Ausgabebefehle aktiviert wird, können Sie auch durch einen Blick auf die Leuchtdiode des Interface sagen. Sie dient nicht nur der Spannungsanzeige, sondern auch der Betriebsanzeige.
Nun wollen wir noch einen kurzen Blick auf die Eingabebefehle werfen. Schließen Sie zwischen E1 (der braunen Leitung am unteren Rand) und +5V (der roten Leitung in der Mitte des Flachbandkabels) einen Taster an.
Probieren Sie aus:

PRINT C:In(E1)

Je nachdem, ob der Taster zwischen E1 und +5V bei der Betätigung der Returntaste des Computers gedrückt war oder nicht, wird auf dem Bildschirm eine 1 oder eine 0 ausgegeben.
Wenn an dem Ausgang noch von vorhin der Motor angeschlossen war, wird er sich wieder rühren. Auch Eingabebefehle aktivieren wieder die Ausgänge des Interface! Nun schließen Sie bitte ein Potentiometer 4,7 kΩ zwischen EX und +5V an. Stellen Sie den Schleifer in eine mittlere Stellung und geben Sie ein

PRINT C:In(Ex)

Die Zahl, die jetzt auf dem Bildschirm erschienen ist, muß zwischen 0 und 280 liegen.
Das Grundprogramm, das Sie eben benutzt haben, werden Sie immer wieder brauchen. Jedes Programm, das mit dem fischertechnik computing Interface Modelle steuern soll, beginnt mit diesem Vorspann, der die neuen Befehle installiert. Bei den Beispielprogrammen auf der Diskette ist das Grundprogramm jeweils schon enthalten. Bei Programmen, die Sie selbst schreiben, beginnen Sie mit dem Grundprogramm und schreiben anschließend weiter.
Damit Sie das Potentiometer leichter beobachten können, wollen wir nun auf diese Weise das erste fischertechnik computing Programm schreiben. Das Grundprogramm befindet sich noch in dem Computer. Gehen Sie wieder in den Editiermodus, indem Sie die Tastenkombination Control-Shift-Alternate drücken. Setzen Sie den Eingabecursor an das Programmende hinter den Befehl **Call Init.** Geben Sie ein

Do
Print C:In(Ex)
Loop

Es dauert einen kurzen Moment, bis das Grundprogramm mit der Datei INTERFAC.COM geladen ist. Es erfolgt wieder die Frage nach der Einstellung der Bildschirmfarben. Und dann geht es los. Im Nu wird der Bildschirm mit Zahlen gefüllt, die ständig nach oben hinausgeschoben werden. Wenn Sie jetzt das Potentiometer in die Hand nehmen und den Schleifer drehen, werden Sie die Veränderung der Zahlen beobachten. Drehen Sie von einem Anschlag zum andern. Die eingelesenen Zahlen sollten zwischen 20 und 280 liegen. Zum Beenden des Programms müssen Sie Control-Shift-Alternate drücken.
Für diejenigen, die etwas genauer die Abläufe verstehen wollen und nicht nur die auf der Diskette vorliegenden Programme benutzen wollen, halten wir hier nun noch Detailinformationen bereit. Die Funk-

tion des Grundprogramms besteht darin, in einen Speicherbereich des Atari ST ein kurzes Maschinenprogramm einzuschreiben. Dieses liegt als Datei INTERFAC.COM auf der Diskette vor.
Das Maschinenprogramm wird in den Speicherplatz geschrieben, der durch das Variablenfeld **Interface%** freigehalten wird. Die Einsprungsadresse wird durch die Funktion **Arrptr** ermittelt. Damit ist die Funktionstüchtigkeit der Software auch unter den unterschiedlichsten Speicherausstattungen und Speicherbelegungen des Atari ST gewährleistet.
Neben dem Maschinenprogramm selbst werden auch noch die genormten Parameter M1, M2, M3, M4, In, E1, E2, E3, E4, E5, E6, E7, E8, Ex, Ey, Ein, Rechts, Links und Aus gesetzt.
Für selbstgeschriebene BASIC-Programme sind unbedingt folgende Einschränkungen zu beachten:
Da die genannten Parameter Werte tragen, die für die korrekte Funktion des Maschinenprogramms sehr wesentlich sind, dürfen diese Parameter nicht als Variablen für andere Zwecke benutzt werden. Das gleiche gilt für das Variablenfeld **Interface%**. Die Einschränkung betrifft allerdings nicht Variablen gleichen Namens, aber anderen Typs, so daß

Aus$, M1%, Interface . . .

durchaus vorkommen dürfen, wenngleich wir davon abraten.
Wer noch weiterforschen will, weil er z. B. die elektronischen Abläufe im Interface verstehen will oder eine Adaption an andere Programmiersprachen vornehmen will, wird auf die Beschreibung des Interface und des Quelltextes verwiesen.
Die letzte Funktion des Grundprogramms besteht in dem Einschalten des Interface und dem Ausschalten aller angeschlossenen Verbraucher. Dies wird durch den Befehl

Call Init

bewirkt. Damit wird das Interface sozusagen in betriebsbereitem Zustand an ein hier anschließendes Benutzerprogramm übergeben. Auch alle Beispielprogramme der Diskette sind nach diesem Muster aufgebaut. Studieren Sie diese, wenn Sie sich Anregungen holen wollen.
Je nach fischertechnik computing Modellbaukasten können gegebenenfalls weitere Befehle an das Interface nach diesem Muster verwendet werden. Sie dienen dann dazu, komplexere oder zeitkritische Abläufe ebenfalls in Maschinensprache zu erledigen. Ein gutes Beispiel hierfür sind die Befehle des fischertechnik computing Trainingsroboter.

```
Rem *************************
Rem *** Interface-Treiber ***
Rem ***       für       ***
Rem ***     ATARI ST    ***
Rem *************************
Rem
Res=Xbios(4)
If Res=0
  Let M$="Das Programm benötigt die|80-Zeichen Darstellung"
  Alert 0,M$,1,"Return",B
  End
Endif
Dim Interface%(90)
Let Init=Lpeek(Arrptr(Interface%()))+2
Bload "INTERFAC.COM",Init
Let M1=Init+&H8
Let M2=Init+&HC
Let M3=Init+&H10
Let M4=Init+&H14
Let E1=1
Let E2=2
Let E3=4
Let E4=8
Let E5=16
Let E6=32
Let E7=64
Let E8=128
Let Ex=160
Let Ey=144
Let Ein=85
Let Rechts=85
Let Links=170
Let Aus=255
Let In=Init+&H5A
Setcolor 0,7,7,7
Setcolor 1,0,0,7
Setcolor 2,7,0,0
Setcolor 3,0,0,0
Deftext 3,0,0,16*Res
Text 64,16*Res,112,"fischer" .
Deftext 2
Text 176,16*Res,112,"technik"
Print At(37,1);"/";
Deftext 1,16
Text 64,34*Res,224,"COMPUTING"
Deftext ,0,900,4*Res
Text 16,184*Res,"Copyright (c) fischerwerke 1987"
Deftext ,,0,9.5*Res
Call Init
```

Wenn Sie ein fischertechnik computing Modell aufgebaut haben, werden Sie vielleicht die Erfahrung machen, daß es nicht so läuft, wie Sie sich das vorgestellt haben. Wen wundert das bei dieser großen Zahl von Leitungen, die zwischen Modell und Interface hin- und herlaufen. Und wenn nur ein Taster vertauscht wäre, die verblüffendsten Effekte könnte dies zur Folge haben. Doppelt schwierig wird die Situation, wenn die Programme selbst geschrieben sind. Wo soll man da mit der Suche anfangen? In der Hardware oder der Software?
Damit Sie die Hardware eindeutig und komfortabel testen können, wurde das Diagnoseprogramm entwickelt. Es liegt auf der fischertechnik computing Diskette als DIAGNOSE.BAS vor. Laden Sie dieses Programm immer zum Austesten eines Modells. Sie können mit ihm sämtliche Eingänge beobachten und feststellen, ob ihr Verhalten mit Ihren Vorstellungen übereinstimmt.
Während die Digitaleingänge ständig angezeigt werden, müssen die Werte der beiden Analogeingänge bei Bedarf immer abgerufen werden. Betätigen Sie dazu die Leertaste. Der Grund für diese Vorgehensweise liegt in einer wechselseitigen Behinderung der Mausbedienung und der Potentiometerabfrage, so daß die Potentiometer nicht dauernd abgefragt werden sollten – Näheres dazu auch in dem Kapitel zur Benutzung von Maus und Interface.
Die Motorausgänge werden mit der Maus angewählt. Jedem Motor ist ein Kasten zugeordnet. In dem jeweiligen Kasten befindet sich das Symbol für Motorstop in der Mitte. Die Symbole rechts davon wählen Rechtslauf an, diejenigen links den Linkslauf. Sie haben dann noch die Wahl zwischen zwei Betriebsarten. Das gestrichelte Symbol bewirkt den Motorlauf nur solange die Maustaste gedrückt ist. In dieser Betriebsart kann präzise positioniert werden. Das durchgezogene Symbole wählt Dauerbetrieb an. Man wird es verwenden, wenn z. B. ein Greifmagnet eingeschaltet bleiben soll, während die Roboterachsen bewegt werden.

Zum gleichzeitigen Abschalten aller Motoren existiert ein großer Not-aus-Kasten. Mit einem weiteren Kasten kann das Ende des Programms angewählt werden.

```
Rem ***************************************
Rem *                                     *
Rem *      fischertechnik computing       *
Rem *                                     *
Rem *              Diagnose               *
Rem *                                     *
Rem *   Copyright (c) fischerwerke 1987   *
Rem *                                     *
Rem ***************************************
Rem
Rem Funktion:
Rem
Rem Mit diesem Programm werden sämtliche
Rem Funktionen der Modelle überprüft.
Rem Alle Eingänge werden angezeigt.
Rem Alle Ausgänge werden über die
Rem Computertastatur angesteuert.
Rem
Rem ****************************
Rem *** Steuerung vorbereiten ***
Rem ****************************
Rem
Gosub Bildaufbau
Rem alle Motoren aus
Rem Status der 4 Motoren
Dim Sta(4),X(4)
For I=1 To 4
  Let Sta(I)=Aus
Next I
Rem Adressen für Motor 1-4
Dim M(4)
Let M(1)=M1
Let M(2)=M2
Let M(3)=M3
Let M(4)=M4
Rem Start mit Motor 1
Let Mot=1
Rem
Rem ****************************
Rem *** Anzeige der Eingabe ***
Rem ****************************
Rem
Do
  Print At(12,13);C:In(E1)
  Print At(15,13);C:In(E2)
  Print At(18,13);C:In(E3)
  Print At(21,13);C:In(E4)
  Print At(24,13);C:In(E5)
  Print At(27,13);C:In(E6)
  Print At(30,13);C:In(E7)
  Print At(33,13);C:In(E8)
  A$=Inkey$
  If A$=Chr$(32)
    Print At(36,13);Using "###",C:In(Ex)
    Print At(41,13);Using "###",C:In(Ey)
  Endif
  Rem ****************************
  Rem *** Steuerung der Ausgabe ***
  Rem ****************************
  Rem
  Rem Mausabfrage
  Mouse X,Y,K
  If Res=1
    Let Y=2*Y
  Endif
  Rem Angewählter Motor
  Zu=Sta(Mot)
  If K
    If Y>240
      If Y>320 And X<316
        Let Mot=2
      Endif
      If Y>320 And X>328
        Let Mot=4
      Endif
      If Y<304 And X<316
        Let Mot=1
      Endif
      If Y<304 And X>328
        Let Mot=3
      Endif
      If Abs(X-188)<=20 Or Abs(X-452)<=20
        Let Sta(Mot)=Aus
        Let X(Mot)=0
        Let Zu=Aus
      Endif
      If Abs(X-92)<=20 Or Abs(X-356)<=20
        Let Sta(Mot)=Links
        Let X(Mot)=96
        Let Zu=Links
      Endif
      If Abs(X-288)<=20 Or Abs(X-548)<=20
        Let Sta(Mot)=Rechts
        Let X(Mot)=-96
        Let Zu=Rechts
      Endif
      If Abs(X-140)<=20 Or Abs(X-404)<=20
        Let Zu=Links
      Endif
      If Abs(X-236)<=20 Or Abs(X-500)<=20
        Let Zu=Rechts
      Endif
    Else
      If X>376
        If Abs(Y-208)<=16
          Call Init
          For I=1 To 4
            Let Sta(I)=Aus
            Let X(I)=0
          Next I
          Let Zu=Aus
        Endif
        If Abs(Y-160)<=16
          Call Init
          Stop
        Endif
      Endif
    Endif
  Endif
  Rem Ausgabe an gewählten Motor
  Let M=M(Mot)
  Dummy=C:M(Zu)
  Sprite M1$,180-X(1),138*Res-12
  Sprite M2$,180-X(2),178*Res-12
  Sprite M3$,444-X(3),138*Res-12
  Sprite M4$,444-X(4),178*Res-12
Loop
End
Rem
Rem ******************
Rem *** Bildaufbau ***
```

```
Rem ******************
Rem
Procedure Bildaufbau
  Deftext 1,0,0,9.5*Res
  Text 64,64*Res,"Diagnose"
  Deffill 1,0
  Prbox 64,72*Res,360,112*Res
  Deftext ,,,6.5*Res
  Print At(10,11);" E1 E2 E3 E4 E5 E6 E7 E8    Ex    Ey";
  Prbox 376,72*Res,576,88*Res
  Text 416,83*Res,"Programmende"
  Prbox 376,96*Res,576,112*Res
  Text 400,107*Res,"Alle Motoren aus"
  Prbox 64,160*Res,316,192*Res
  Prbox 64,120*Res,316,152*Res
  Prbox 328,160*Res,576,192*Res
  Prbox 328,120*Res,576,152*Res
  Defline 1,1,0,1
  Circle 356,178*Res,20,2250,1800
  Circle 356,138*Res,20,2250,1800
  Circle 92,178*Res,20,2250,1800
  Circle 92,138*Res,20,2250,1800
  Defline 3
  Circle 404,178*Res,20,2250,1800
  Circle 404,138*Res,20,2250,1800
  Circle 140,178*Res,20,2250,1800
  Circle 140,138*Res,20,2250,1800
  Defline 1,,,0
  Circle 452,178*Res,20
  Circle 452,138*Res,20
  Circle 188,178*Res,20
  Circle 188,138*Res,20
  Defline 3,,1
  Circle 500,178*Res,20,0,3150
  Circle 500,138*Res,20,0,3150
  Circle 236,178*Res,20,0,3150
  Circle 236,138*Res,20,0,3150
  Defline 1,,1
  Circle 548,178*Res,20,0,3150
  Circle 548,138*Res,20,0,3150
  Circle 284,178*Res,20,0,3150
  Circle 284,138*Res,20,0,3150
  Sprite M1$,180,138*Res-12
  Sprite M2$,180,178*Res-12
  Sprite M3$,444,138*Res-12
  Sprite M4$,444,178*Res-12
Return
Rem Sprite-Daten
Procedure Sprite_daten
  Let M1$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M1$=M1$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Hinten
    Let M1$=M1$+Mki$(Hinten)
  Next I%
  For I%=1 To 16
    Read Vorn
    Let M1$=M1$+Mki$(Vorn)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,49926,59150,65310,56118,49958,49926,49926
  Data 49926,49926,49926,49926,49926,49926,49926,49926
  Let M2$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M2$=M2$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Hinten
    Let M2$=M2$+Mki$(Hinten)
  Next I%
  For I%=1 To 16
    Read Vorn
    Let M2$=M2$+Mki$(Vorn)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,49948,59198,65379,56129,49921,49921,49923
  Data 49926,49932,49944,49968,50016,50016,50047,50047
  Let M3$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M3$=M3$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Hinten
    Let M3$=M3$+Mki$(Hinten)
  Next I%
  For I%=1 To 16
    Read Vorn
    Let M3$=M3$+Mki$(Vorn)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,50047,59263,65283,56070,49932,49944,49982
  Data 49927,49923,49923,49923,50019,50019,50047,49982
  Let M4$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M4$=M4$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Hinten
    Let M4$=M4$+Mki$(Hinten)
  Next I%
  For I%=1 To 16
    Read Vorn
    Let M4$=M4$+Mki$(Vorn)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,49926,59142,65294,56078,49950,49942,49974
  Data 49958,50022,50022,50047,50047,49926,49926,49926
Return
```

GfA-BASIC

Die fischertechnik computing Programme der beigefügten Diskette sind, wie bereits erwähnt, in GfA-BASIC geschrieben. Die in den fischertechnik computing Modellanleitungen dokumentierten Programme sind jedoch in dem BASIC eines anderen Computers formuliert. Die Unterschiede in den Programmlisten sind schon beträchtlich, insbesondere weil GfA-BASIC keine Zeilennummern als Strukturierungsmittel verwendet. Stattdessen stehen eine Reihe von leistungsfähigen Strukturbefehlen zur Verfügung, die in der vorliegenden Software auch optimal genutzt wurden. Das BASIC-typische GOTO wurde in keinem Fall mehr verwendet.
Auch die Gestaltungsmöglichkeiten auf dem Bildschirm reichen weit über das hinaus, was in einem „klassischen" BASIC möglich ist. Hinzu kommt die Möglichkeit auf die Routinen des Betriebssystems GEM mit dessen Unterstützung von Bildschirmfenstern und der Maus zurückzugreifen.
Aus all diesen Gründen lehnt sich die Gestaltung der Atari ST Software für die fischertechnik computing Baukästen nicht sehr eng an die Vorlage an, sondern schöpft aus diesen Möglichkeiten. Der eigentliche Ablauf der Steuerung der Modelle hält sich jedoch immer noch an die ausgedruckte Vorlage. Wir stellen hier noch einmal die wichtigsten unterschiedlichen Kommandos gegenüber:

fischertechnik computing Programmieranleitung	GfA-BASIC
SYS M1,EIN	**Dummy=C:M1(Ein)**
PRINT USR(E1)	**Print C:In(E1)**
SYS INIT	**Call Init**
PRINT CHR$(147)	**Cls**
GET A$	**A$=Inkey$**

Die Steuerung der Ausdruckposition vereinfacht sich unter GfA-BASIC, da das Print-Kommando eine Positionierung in der Form

Print At(Spalte,Zeile)

erlaubt. Auch kann das Ausdruckformat durch die Angabe eines Formats

Print Using(. . .)

festgelegt werden. Die Überschrifttexte wurden mit den Kommandos

Deftext und **Text**

gestaltet, mit deren Hilfe sich verschiedene Schriftarten und -größen realisieren lassen.
Studieren Sie die Programme auf der fischertechnik computing Diskette, um die Programmierung unter GfA-BASIC zu verstehen.

Bei der gleichzeitigen Benutzung der Maus und der Analogwerteingabe des Interface kommt es zu leichten Behinderungen der Abläufe im Computer, die sich jedoch bei geschickter Programmierung weitgehend vermeiden lassen. Bei der Ermittlung eines Analogwertes benötigt das Interface die ungeteilte Aufmerksamkeit des Mikroprozessors. Dies erfordert unter anderem auch, daß der Informationsfluß von der Tastatur zu dem Mikroprozessor unterbunden werden muß. Während der Ermittlung des Analogwertes kann der Computer somit keine Tastaturbetätigung und keine Mausbewegung erkennen. Für die Tastatur ist das nicht so schlimm, da eine Reihe von Zeichen in der Tastatur zwischengelagert werden können, bis die Verbindung nach Ende der Analogeingabe wieder hergestellt wird. Bei schnellen Mausbewegungen kann aber das Zwischenlager in der Tastatur (die Maus ist elektrisch an die Tastatur angeschlossen) überfüllt werden. Dies führt zu Fehlermeldungen, die sich durch ein Klicken des Lautsprechers äußern. Die Fehlermeldungen beeinträchtigen aber nicht den Programmlauf.
Wie empfehlen trotzdem, bei den fischertechnik computing Programmen die Maus nur zu bewegen, wenn eine entsprechende Eingabe durch das Programm angefordert wird. Bei selbstgeschriebenen Programmen achten Sie bitte darauf, daß Maus- und Analogwertabfragen nicht im schnellen Wechsel sondern zu unterschiedlichen Phasen des Programms erfolgen. Die fischertechnik computing Programme können als Beispiel dienen. Bei dem Diagnoseprogramm wird z.B. die Maus ständig abgefragt, der Analogwert jedoch nur bei Betätigung der Leertaste. Bei dem Programm TIR_MAUS.BAS wird die nächste Roboterposition mit der Maus angefordert und erst anschließend der Roboter unter Benutzung der Positionsmessung am Analogkanal positioniert.

Weiterhin empfehlen wir, an die beiden Analogeingänge **immer** einen Analogwertgeber anzuschließen. Sollte das betreffende Modell keinen Analogwertgeber (z.B. Potentiometer, Fotowiderstand oder dergleichen) besitzen, so schließen Sie die Eingänge EX und EY einfach mit der Referenzspannung +5 V zusammen. Dies entspricht einem Widerstandswert von Null Ohm.
Sollte jetzt eine Analogwerteingabe versehentlich durch einen Störimpuls angestoßen werden, so wird die Datenleitung zwischen Atari ST und Interface nicht unnötig lange blockiert.

Checkliste

Sollte das fischertechnik computing Interface sich einmal widerwillig zeigen und nicht so arbeiten, wie Sie es erwarten, so überprüfen Sie bitte folgende Punkte mit dem Diagnoseprogramm:
Das Diagnoseprogramm zeigt bei E1 bis E8 überall 0 an, obwohl ein Modell angeschlossen ist. – Das Interface ist nicht an den Computer oder nicht an das Netzgerät angeschlossen.
Einer der Eingänge E1 bis E8 zeigt bei Betätigung des Tasters gerade das umgekehrte Ergebnis. – Öffner- und Schließerfunktion des Tasters sind vertauscht.
Einer der Eingänge E1 bis E8 zeigt immer das Ergebnis 0, obwohl er angeschlossen ist und betätigt wird. – Prüfen Sie die Verkabelung.
Einer der Eingänge E1 bis E8 zeigt immer das Ergebnis 1, selbst wenn kein Modell angeschlossen ist. – Vermutlich das Eingangsgatter IC 4014 durch Überspannung (elektrostatische Aufladung) beschädigt.
Ein Motorausgang arbeitet nicht. – Bitte Verkabelung überprüfen.
Ein Motorausgang arbeitet nur in einer Richtung. – Leistungsstufe des Motors defekt.
Motor läuft sehr langsam oder setzt aus. – Entweder Netzgerät durch zu viele Motoren überlastet (zwei Netzgeräte mot4 oder das stärkere computing Netzgerät verwenden) oder Netzgerät bei Verwendung des regelbaren Ausgangs nicht voll aufgedreht.
Bei Defekten schicken Sie das Gerät bitte an die fischerwerke, Abt. Service, ein.

Technische Daten

fischertechnik computing Interface zum Anschluß an Druckerschnittstellen nach der Centronics-Norm, Art.-Nr. 30566.
Das Interface kann unter anderem an den Computern Atari 260 ST, Atari 520 ST, Atari 520 ST+, Atari 1040 ST, Atari Mega ST sowie deren jeweiligen Varianten verwendet werden.
4 Ausgänge zum Anschluß von Motoren, Lampen, Elektromagneten.. (M1 bis M4).
Polarität des Ausgangs steuerbar.
Belastbarkeit: 1A Dauerstrom, 1,5A Spitzenstrom.
8 Eingänge für digitale Signale (E1 bis E8).
Durch interne Beschaltung sowohl Anschluß von elektromagnetischen Artikeln (Taster, Schalter, Relais) in positiver Logik als auch Anschluß von TTL-Ausgängen möglich. Schutz gegen Überspannung eingebaut.
2 Eingänge für analoge Signale (EX und EY).
Anschließbar sind Geber mit Widerstandswerten zwischen 0 und 5 kΩ, z.B. Potentiometer, Fotowiderstände...
Überwachungsschaltung des Datenstroms. Bei Ausbleiben von Datensignalen des Atari ST schaltet das Interface nach 0,5 Sekunden alle Ausgänge inaktiv. Die Signale bleiben jedoch gespeichert.
Überwachungsschaltung der Software. Bei gravierenden Syntaxfehlern spricht ebenfalls die Überwachungsschaltung, jedoch ohne Verzögerung, an.
Die Überwachungsschaltung reagiert auch auf Unterversorgung des Interface, sei es durch Überlastung oder zu niedrige Spannung des Netzgeräts.
Die zu dem Atari ST und dem Modellbaukasten passende Software und der an die Druckerschnittstelle passende Adapter wird als getrenntes Set geliefert.

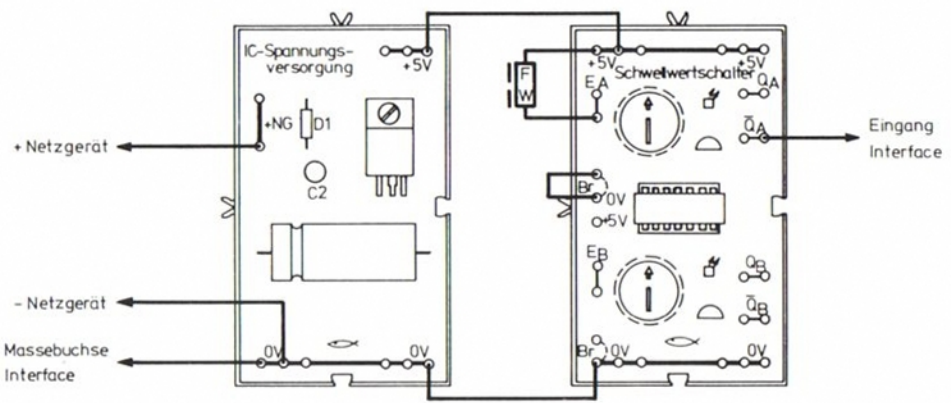
Benutzung von fischertechnik Elektromechanik und Elektronik

Das fischertechnik computing Interface ist kompatibel zu den Bauteilen und Elektronikbausteinen der obengenannten Baukästen. Anstelle der bei den fischertechnik computing Modellen verwendeten mini-Taster können Sie genauso gut Taster und Schalter anderer Bauart anschließen. Z.B. den großen Taster oder den Polwendeschalter, aber auch den Reedkontakt oder den Schaltkontakt eines Relais. Aufpassen müssen Sie jedoch bei der Verwendung von selbstgebauten Tastern und Schaltern aus Gelenkbausteinen und Federn. Hier könnten eventuell Prellerscheinungen auftreten. Wir empfehlen, in diesen Fällen den Taster mehrmals abzufragen und den Wert nur dann als gültig zu erachten, wenn zweimal hintereinander der gleiche Wert erschien.

Die Analogeingänge des Interface können mit Sensoren beschaltet werden, die einen Widerstandswert zwischen 0 und 5 kΩ als Ausgang liefern. Zunächst bieten sich die Potentiometer aus dem Baukasten fischertechnik computing an. Genauso können aber auch andere Bauelemente, wie z.B. der Fotowiderstand, verwendet werden.

Die Motorausgänge des Interface sind kräftig belastbar. Nicht nur die mini-Motoren, auch der S-Motor und der N-Motor lassen sich mit dem Interface ansteuern, wobei noch eine Lampe zur Funktionsanzeige parallelgeschaltet sein darf. Außer Motoren eignen sich noch der Elektromagnet und das Relais RBII.

Die Signale der Elektronikbausteine mit integriertem Schaltkreis aus der TTL-Familie (z.B. Schwellwertschalter) können ebenfalls in die Eingänge des Interface eingespeist werden. Als gemeinsamer Bezugspunkt ist jedoch zuvor die Masseschiene des Elektronikbausteins mit der Massebuchse des Interface zu verbinden. In der Abbildung zeigen wir, wie eine Lichtschranke aufgebaut wird. Der Schwellwertschalter dient dazu, die Ansprechschwelle der Lichtschranke einzustellen.



Funktionsweise des Interface und des Interface-Treiberprogramms

Wenn Sie die fischertechnik computing Software benutzen oder selbst Programme entsprechend der Hinweise in den vorigen Kapiteln erstellen, werden Sie kaum die nun folgende Information benötigen. Wenn Sie aber die Programme in anderen Sprachen als BASIC formulieren wollen, die Programme durch komplexe Abläufe in Maschinensprache beschleunigen wollen, die Funktionen des Interface erweitern wollen oder auch nur einfach einen Blick hinter die Kulissen werfen wollen, so wird Ihnen das Nachfolgende sicherlich hilfreich sein. Allerdings sollten Sie dann auch ein paar Kenntnisse der Maschinensprache und der Digitalelektronik mitbringen, denn hier geht es an die „bits and pieces".
Das fischertechnik Interface erfüllt eine Reihe von Aufgaben, die wir anhand des Blockdiagramms besprechen wollen. Am linken Rand sind die Signale von und zu dem Computer aufgeführt. Es fällt auf, daß diese recht wenig mit den Ausgängen M1 bis M4 und Eingängen E1 bis E8 sowie EX und EY gemein haben. Der Grund ist darin zu suchen, daß am Computeranschluß wesentlich weniger Datenleitungen zur Verfügung stehen, als auf der Modellseite des Interface benötigt werden. Diese wenigen Datenleitungen müssen deshalb so eingesetzt werden, daß alle Signale auf der Modellseite gesteuert werden können. Das Konzept sieht eine Mehrfachverwendung der Datenleitungen mit Hilfe von Schieberegistern vor. Auf diese Weise werden z.B. nur drei Datenleitungen für die Steuerung der Ausgabe notwendig. Eine parallele Anschlußweise hätte acht Datenleitungen benötigt.

Schauen wir uns gleich die Ausgabe an den Anschlüssen M1 bis M4 genauer an. Die dafür benötigten Datenleitungen werden mit DATA-OUT, CLOCK und LOAD-OUT bezeichnet. Bei einer Ausgabe werden immer die Daten für alle vier Motoren übertragen, d.h. ein ganzes Byte. (Ein Byte deswegen, weil jeder der vier Motoren zwei Bits zur Steuerung der Drehrichtung benötigt). Die von dem Kommando nicht betroffenen Motorausgänge erhalten somit den derzeitigen Stand, der im Computer als Ausgabewort zwischengespeichert ist, erneut eingeschrieben.
Bei der Ausgabe werden der Reihe nach die Bits des Ausgabeworts an die Leitung DATA-OUT angelegt, das höchstwertige zuerst. Mit einem Übergang von low nach high am Ausgang CLOCK wird das Bit in ein Schieberegister übernommen. Danach folgt das nächste Bit an DATA-OUT, das ebenfalls in das Schieberegister mit dem nächsten CLOCK-Impuls übernommen wird. Das vorangegangene Bit ist dabei aber auch um eine Position im Schieberegister nach rechts gerutscht, um dem nachfolgenden Platz zu machen. Nach insgesamt acht solchen Datenübertragungen ist das ganze Ausgabewort im Schieberegister abgelegt. Das zuerst übertragene Bit ist im Verlaufe des Datentransfers ganz nach rechts durchgeschoben worden. Von der Aktivität im Schieberegister ist aber bislang an seinen Ausgängen noch nichts spürbar. Die Ausgangsverstärker werden nicht direkt über das Schieberegister angesteuert, sondern über ein zwischengeschaltetes Speicherregister, das auch noch im Schieberegister-Baustein integriert ist. Erst mit dem Übergang von low nach high am Ausgang LOAD-OUT erfolgt die Übernahme in das Speicherregister. Die zeitliche Abfolge der Signale können Sie dem Impulsdiagramm entnehmen.

Ob die Daten allerdings auch die Leistungsverstärker durchsteuern, hängt wiederum von der Freigabesteuerung des Speicherbausteins ab. Die Freigabesteuerung erfolgt durch ein Monoflop. Diese Schaltung erzeugt ein Freigabesignal von einer halben Sekunde Dauer, wenn ein Impuls auf der CLOCK-Leitung vorliegt. Wir können davon ausgehen, daß zunächst die Leistungsverstärker angesteuert werden, da zuvor gerade die Daten mit Hilfe der CLOCK-Leitung übertragen wurden. Sollte aber innerhalb der nächsten halben Sekunde kein weiterer Datentransfer erfolgen, so kippt das Monoflop wieder in seinen stabilen Zustand zurück und das Freigabesignal wird zurückgenommen. Das Monoflop ist übrigens nachtriggerbar, d.h. die Zeitdauer von einer halben Sekunde rechnet sich jeweils vom Zeitpunkt des letzten CLOCK-Impulses an.

Auch das Monoflop besitzt einen Freigabeeingang. Über jenen kann letztlich die Ausgabe an die Verstärker sofort unterbunden werden. Beim fischertechnik Interface erfolgt dies, wenn ein ungültiges Datenmuster am Ausgang des Speicherregisters anliegen würde, das einen angeschlossenen Motor quasi in Rechts- und Linkslauf gleichermaßen steuern würde.

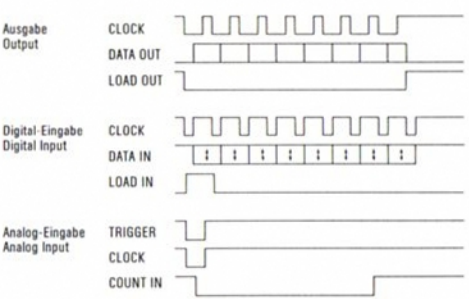
Nun zu der Übertragung der digitalen Signale an E1 bis E8. Im Prinzip findet bei der Eingabe eine Umkehrung des oben Beschriebenen statt. Durch das Ausgabe-Signal LOAD-IN werden die an den Eingängen anstehenden Signale in das Eingaberegister übernommen. Dies erfolgt wiederum für alle acht Eingänge, auch wenn nur ein einziger abgefragt werden soll. In dem Schieberegister angelangt, bringt jeder Impuls auf der CLOCK-Leitung ein Bit auf der Eingabeleitung DATA-IN zum Vorschein, jenes von E8 zuerst und das von E1 zuletzt. Durch Testen dieser Leitung kann der Computer die Bits „aufsammeln" und wieder ein Datenwort bilden. Das gewünschte Bit wird anschließend herausgefiltert und dem BASIC-Programm übergeben.

Da zur Übertragung der Daten dieselbe CLOCK-Leitung wie bei der Ausgabe benutzt wird, wird auch bei der digitalen Eingabe das Monoflop aktiviert, das das Freigabesignal für die Ausgabedaten steuert. Eine Fehlfunktion des Ausgabeschieberegisters durch die Mehrfachfunktion der CLOCK-Leitung steht nicht zu befürchten, denn die aktuellen Ausgabedaten stehen ja nicht im Ausgabeschieberegister, sondern im Speicherregister. Ersteres wird zwar wohl durch die CLOCK-Impulse beeinflußt,

nicht aber letzteres, das ja nur auf das Signal LOAD-OUT reagiert.
Bleiben zum Schluß noch die Analogeingänge EX und EY. Die Potentiometer oder sonstigen veränderlichen Widerstände dienen als zeitbestimmendes Bauelement in zwei weiteren Monoflop-Schaltungen. Ein niedriger Widerstandswert wird in einen Impuls kurzer Dauer, ein hoher Widerstandswert in einen Impuls langer Dauer umgesetzt. Der Impuls selbst wird durch Startsignal TRIGGER-X bzw. TRIGGER-Y (mit negativer Logik) ausgelöst und erscheint dann auf der Leitung COUNT-IN. Ein Maschinenprogramm stellt die Impulsdauer anhand der Zahl der Schleifendurchläufe fest, die während der Impulsdauer durchgeführt werden können. Diese Zahl wird in das aufrufende BASIC-Programm zurückgegeben. Sie sehen also, daß der Analogwert weder die Winkelstellung noch den Widerstandswert der Potentiometer darstellt. Dagegen geht die Arbeitsgeschwindigkeit des Prozessors ein. Zwischen der letztlich ermittelten Zahl und dem Widerstandswert besteht ein linearer Zusammenhang. Dieser muß gegebenenfalls im BASIC-Programm noch anhand einer Eichung in Winkelgrade oder Widerstandswerte umgerechnet werden.
Wir wollen jetzt noch kurz zusammenfassen, wie das Interface mit dem Atari ST verbunden wird. Es wird, wie Sie ja schon von dem Kapitel über den Anschluß des Interface wissen, die Parallel-Druckerschnittstelle verwendet. Von deren acht Datenleitungen werden die untersten sechs für die oben besprochenen Ausgabesignale verwendet, siehe auch die nachstehende Tabelle.
Ein kleines Problem gilt es für die Eingabeleitungen zu lösen. Die Parallel-Druckerschnittstelle beinhaltet nur eine für alle PC verbindliche Eingabeleitung, das Aktiv-Signal des Druckers, BUSY. Es entsteht jedoch kein Konflikt, wenn alle Eingabeleitungen mit Hilfe einer ODER-Schaltung zusammengefaßt werden. Da das Maschinenprogramm ja „weiß", welche Eingabefunktion es angefordert hat, kann es die Signale an dieser einzigen Eingabeleitung auch korrekt interpretieren.

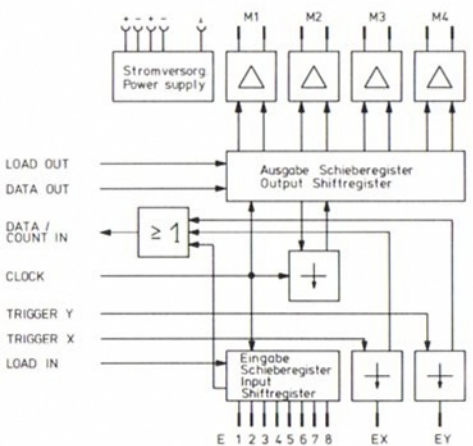
Anschluß des fischertechnik Interface an die Parallel-Druckerschnittstelle.

Interface-Signal	Drucker-Signal	Stiftnummer
LOAD-OUT	**Datenbit 1**	**2**
LOAD-IN	**Datenbit 2**	**3**
DATA-OUT	**Datenbit 3**	**4**
CLOCK	**Datenbit 4**	**5**
TRIGGER-X	**Datenbit 5**	**6**
TRIGGER-Y	**Datenbit 6**	**7**
DATA/COUNT-IN	**Busy**	**11**



Impulsdiagramme des fischertechnik Interface
Pulse Diagrams of the fischertechnik Interface

Das nachstehende Maschinenprogramm steuert die zuvor beschriebenen Abläufe. Zu der Übergabe der Parameter in das Maschinenprogramm und wieder zurück studieren Sie die Beschreibung des CALL-Befehls bzw. der C:-Funktion in der GfA-BASIC-Anleitung.



```
;****************************************
;* ATARI ST Interface Treiber           *
;* Version 1                            *
;* Copyright (C) fischerwerke 1987      *
;* Steuerung des fischertechnik Interface *
;* für Aus- und Eingabe mithilfe des    *
;* C: Kommandos.                        *
;****************************************
;* Dieses Programm benötigt eine        *
;* Centronics-Parallel Schnittstelle    *
;*                                      *
;* Das Maschinenprogramm steht in einem *
;* Integerfeld (wird durch NEW und CLEAR *
;* gelöscht!).                          *
;* Erster Aufruf mit CALL init.         *
;*                                      *
;* Ausgabe Steuerung:                   *
;*                                      *
;* dummy=C:Motornummer(Betriebsart)     *
;* Motornummern sind M1,M2,M3 und M4    *
;* Betriebsarten sind RECHTS,LINKS,EIN,AUS*
;* EIN entspricht rechts                *
;*                                      *
;* CALL init                            *
;* Initialisiert das Interface und      *
;* schaltet alle Motoren ab.            *
;*                                      *
;* Eingabe Steuerung:                   *
;* Digital-Eingabe Kommandos:           *
;* X=C:In(Digital-Eingang)              *
;* Digital-Eingänge sind E1 bis E8 .    *
;* X enthält danach den Zustand des     *
;* Einganges (0 oder 1).                *
;*                                      *
;* Analog-Eingabe Kommandos:            *
;* X=C:In(Analog-Eingang)               *
;* Analog-Eingänge sind EX und EY .     *
;* X enthält danach den Wert.           *
;*                                      *
;****************************************
;* Initialisierungs-Routine             *
;* Aufruf mit CALL init                 *
;****************************************

        section status

init:   clr.l   d7              ;d7 loeschen
        lea     avar,a4         ;Adresse Ausgabevariable
        bra.s   stvar
m1:     moveq   #$03,d6         ;Motor 1
        bra.s   bout
m2:     moveq   #$0c,d6         ;Motor 2
        bra.s   bout
m3:     moveq   #$30,d6         ;Motor 3
        bra.s   bout
m4:     moveq   #-64,d6         ;Motor 4
;*********************************
;*      Einzelbit Ausgabe        *
;*********************************
bout:   clr.l   d7              ;d7 loeschen
        move.w  $04(sp),d5      ;Drehrichtung nach d5
        lea     avar,a4         ;Adresse Ausgabevariable
        move.b  (a4),d7
        or.b    d6,d7
        and.b   d6,d5
        eor.b   d5,d7           ;setze Drehrichtung

stvar:  move.b  d7,(a4)         ;setze Ausgabevariable
        bsr     disabl          ;Interrupt sperren
        bsr.s   shout           ;Ausgabe an Interface
        bsr     enable          ;Interrupt freigeben
        rts                     ;zurueck zu BASIC
;*********************************
;* Routine zur Interface-Steuerung    *
;* Ausgabe                            *
;* Ausgabemuster wird in d7 uebergeben*
;* benutzt d5,d6,d7.                  *
;*********************************
shout:  moveq   #8,d6           ;Schleifenzaehler
loop:   moveq   #$30,d5         ;Ruhepegel Interface
        rol.b   #1,d7           ;naechstes Bit
        bcc.s   null            ;=0?
        or.b    #4,d5           ;setze DATA OUT
null:   move.b  d5,$02(a6)      ;Ausgabe
        or.b    #8,d5           ;setze CLOCK
        move.b  d5,$02(a6)      ;Ausgabe
        subq.b  #1,d6           ;
        bne.s   loop            ;Schleifenende
        move.b  #$39,$02(a6)    ;setze LOAD OUT
        rts
;*********************************
;* Eingaberoutine                *
;*********************************
inp:    move.w  $04(sp),d4      ;hole Maske
        bsr.s   disabl          ;Interrupt sperren
        cmp.w   #$a0,d4         ;Eingang Ex?
        beq.s   poti            ;dann Analogeingabe
        cmp.w   #$90,d4         ;Eingang Ey?
        beq.s   poti            ;dann Analogeingabe
        bsr.s   shin            ;Digitaleingabe
        and.w   d4,d7           ;vergleiche Masken
        beq.s   basret
        move.w  #1,d7           ;Eingang High
basret: bsr     enable          ;Interrupt freigeben
        clr.l   d0
        move.w  d7,d0           ;Rueckgabevariable nach d0
        rts                     ;zurueck zu BASIC
;*********************************
;* Routine zur Interface-Steuerung    *
;* Eingabe                            *
;* Eingabemaske wird in d7 uebergeben*
;* benutzt d5,d6,d7                   *
;*********************************
shin:   moveq   #$32,d6         ;setze LOAD In
        move.b  d6,$02(a6)      ;Ausgabe
        or.b    #8,d6           ;setze CLOCK
        moveq   #8,d5           ;Schleifenzaehler
loop2:  move.b  d6,$02(a6)      ;Ausgabe
        asl.b   #1,d7           ;Datenwort hochschieben
        btst    #0,(a5)         ;teste DATA IN
        bne.s   null2           ;Eingang Low?
        or.b    #1,d7           ;Eingang High
null2:  move.b  #$30,$02(a6)    ;loesche CLOCK
        move.b  #$38,$02(a6)    ;setze CLOCK
        subq.b  #1,d5           ;Schleifenzaehler
        bne.s   loop2           ;Schleifenende
        rts
;*********************************
;* Analogeingabe                 *
;*********************************
poti:   clr.l   d7              ;Zaehler auf 0
        move.b  d4,$02(a6)      ;setze TRIGGER
        move.b  #$38,$02(a6)    ;setze CLOCK

loop3:  btst.b  #0,(a5)         ;Puls zu Ende?
        bne.s   stop
        addq.w  #1,d7           ;Zaehler erhoehen
        bne.s   loop3           ;weiterzaehlen
stop:   bsr.s   enable          ;Interrupt freigeben
        clr.l   d0
        lsr.w   #1,d7           ;durch 2 teilen
        move.w  d7,d0           ;Rueckgabevariable nach d0
        rts                     ;zurueck zu BASIC
;*********************************
;* Interrupt sperren             *
;benutzt d0                     *
;*********************************
disabl: clr.l   -(sp)
        move.w  #$20,-(sp)      ;Supervisormode setzen
        trap    #1
        add.l   #6,sp
        ori.w   #$0700,sr       ;alle Interrupts sperren
        movea.l sound,a6        ;Adresse Soundchip
        move.b  #7,(a6)         ;Register 7 anwaehlen
        bset.b  #7,$02(a6)      ;Datenrichtung Ausgabe
        move.b  #15,(a6)        ;Register 15 anwaehlen
        movea.l mfp,a5          ;Adresse MFP
        bclr.b  #5,$08(a5)      ;BUSY-Interrupt sperren
        bclr.b  #0,$04(a5)      ;Datenrichtung BUSY-Ausgabe
        rts
;*********************************
;* Interrupt freigeben           *
;* benutzt d0                    *
;*********************************
enable: bset    #5,$08(a5)      ;BUSY-Interrupt freigeben
        andi.w  #$3300,sr       ;alle Interrupts freigeben
        move.l  d0,-(sp)        ;User-Statusregister restaurieren
        move.w  #$20,-(sp)
        trap    #1              ;in Usermode zurueckkehren
        add.l   #6,sp
        rts

        section data

avar:   ds.b    1               ;Ausgabevariable
sound:  dc.l    $ff8800         ;Adresse Soundchip
mfp:    dc.l    $fffa01         ;Adresse MFP

        end
```

The fischertechnik computing Interface

Dear friend of fischertechnik:

fischertechnik computing has been developed to allow computer owners to control technical models with a computer, thereby extending the computer's capabilities. fischertechnik computing includes the fischertechnik computing Kit, the fischertechnik computing construction sets, the fischertechnik computing interfaces and software. Computer owners can now simulate technical functions and processes, solve problems and have a lot of fun controlling models with the computer.

What is needed for controlling the models? First, the fischertechnik model which will perform the processes. Second, a personal computer such as the one you own to provide for control and coordination. Third, an interface which will link the two. The item in your hands is the fischertechnik computing interface. Control signals from the computer, e.g. "Turn on motor!" are converted by the interface into currents strong enough to make a motor move. This is called "output". The point of view is from the computer outward.
The opposite direction is also possible and occurs in practice. The models are equipped with pushbutton switches, potentiometers, etc., in order to inform the computer about what is happening on the model. Here again, the interface provides assistance by modifying the signals in such a way that they represent an input which is meaningful to the computer.

The fischertechnik interface provides the following functions:

- With it you can control fischertechnik motors, lamps, electromagnets, etc.
- With it you can interrogate eight pushbuttons or switches.
- Additionally, two inputs provide values from variable signal transmitters such as potentiometers.

What would be the use of all the electrical connections between the computer and the fischertechnik model provided by the interface without a means of activating them? We are referring to software, of course. The software is located on a diskette. The diskette contains a program which expands the language of your computer in such a way that it can control the interface. This program will be the core of your own programs. The diskette contains a lot more, though: in order for you to study and learn the use of these new resources, sample programs for all fischertechnik computing models are also included on it.
As you can see, a lot of challenging problems are waiting to be solved. I hope you will enjoy solving them! Yours sincerely,

Connecting the Interface

The fischertechnik interface Centronics Port will be hooked up to parallel printer-ports in Centronics norm. Such a printer port is also available for the computers of the Atari ST line: Atari 260 ST, Atari 520 ST, Atari 1040 ST, and Atari Mega ST. Nevertheless the interface is not a printer. Printers can only show an output, i.e. to evaluate and transpose signals from the computer so that a text appears on the paper. The fischertechnik interface is able to do more: it can give output signals very much like a printer, but this time to move e.g. the drive motor of a robot. On the other hand, it can also read input signals, e.g. register the quantity of light measured in an optical sensor. Simply said, this is the reason why the fischertechnik interface cannot be addressed as a printer (e.g. with LPRINT). Rather a special interface driver routine has to be integrated into the software. The driver-routine itself is a software-component, written in the machine-language of the microprocessor of the Atari ST (Motorola 68000). The enclosed fischertechnik computing software will integrate the driver-routine into the programming language GfA-BASIC under the operating system TOS with GEM.

Let's get back to the hardware. The fischertechnik interface is connected to the Atari ST printer port, as we have already mentioned above. Use the following procedure:

- Make sure the computer is turned off.
- Turn the computer around so that you are facing the ports on the rear panel. Locate the parallel printer port. This port is a jack connector which will accept 25 pins. You cannot mistake it for the other interfaces, for example, the serial interface, because they have pins, not jacks.
- You may be electrostatically charged to several thousand volts by frictional electricity without noticing it and without it doing you any harm. However, these voltages are harmful to the circuits contained in the interface and the computer. It is therefore necessary to discharge any electrostatic charges by touching a grounded object.
- Place the interface in front of you. Together with this instruction manual and the diskette for the fischertechnik computing model kit you got the adaptor connector fitting to the printer port of the Atari ST. Insert the cable of the interface into the adaptor connector. A reversed insertion is prohibited by the shape of the connector. Now plug the adaptor onto the parallel printer port of your Atari ST. Also in this case plugging in the wrong way is impossible due to the trapezoidal shape of the connector.
- Connect the interface to the power supply unit. The interface expects an unfiltered direct current between 6 and 10 volts. If a stabilized direct voltage source is used, the power supply must be adjusted to approximately 8 volts (loadable to approximately 2 amps). Using the fischertechnik computing power supply unit will guarantee the correct voltage. Plug the output cable of the power supply unit (red flat plug) into the jack marked with a "+" on the interface, and the output cable with the green flat plug into the jack marked with a "—". Two pairs of jacks are provided; it does not matter which pair you use.
- For models with up to two motors, the fischertechnik power supply units mot 4 may also be used, i.e. especially for the models of the fischertechnik computing kit. If you intend to use your own constructions with a greater number of motors or the training robot or Plotter/Scanner kits, you will need a more powerful power source. In this case, you can either switch to the more powerful fischertechnik computing power supply unit or plug a second fischertechnik power supply unit mot 4 into the empty jacks.
- When using the mot 4 power supply units, you use the two-wire cable and flat plugs supplied with the kits to make the power cord.
- Connect the fischertechnik computing model to the interface. Use the twenty-wire ribbon cable supplied with the kits and the fischertechnik computing kit. This cable can also be ordered separately from the fischertechnik Service Set.
- The turn-on sequence of the interface and the computer does not matter. If you are not using the interface while working with other programs, it is not necessary to unplug the interface. Just leave it turned off.
- Side effect caused by the interface: as long as the interface is plugged in, you cannot use a printer which is to be connected to the same port. Electronic switches to direct the data flow to a number of printers are also available in computer stores. Such a switch may be placed between the Atari ST and the fischertechnik interface. In this way you need not exchange the connecting cables of the printer and the interface each time.

Prior to working with the models you should also always discharge possible electrostatic charges by touching a grounded metal object.
Enough about hardware – in the next section we'll deal with the fischertechnik computing software.

fischertechnik computing Software

Anyone who has toyed with the idea of controlling some sort of device or model with a computer may know from experience or may have heard from friends that this is not the easiest of tasks. That it requires a thorough knowledge of the computer, the microprocessor, the input/output devices and machine language.
These difficulties, which were considerable, were a good reason for many not to experiment with this fascinating field of computing – up to now, that is. fischertechnik computing provides the proper interface for your computer, and with it, the software you need to make it all work smoothly and easily.
In order to avoid losing the software through accidental deletion or damage to the diskette, you should make a copy of the fischertechnik computing diskette. To do so proceed in a similar way as described in the instruction manual of your Atari ST to generate a backup diskette. I.e. first of all format a blank diskette. Select single-sided format (360 K). While you are at it, format a few additional diskettes. Some of the fischertechnik computing programs will write data to a diskette. Since the fischertechnik computing diskette is almost completely filled with programs, you will need formatted blank diskettes.
The next step is to put the fischertechnik master diskette into disk drive A. If you have a second disk drive B connected, put the formatted disk into it. Using the mouse drag the icon of disk drive A over the icon of drive B and release the mouse-button. There upon a window appears. It informs you that through the copy process all data on diskette B will be overwritten. The proceding window is part of the DISKCOPY program. Confirm again your intention to copy the diskette. Hereafter the copying process will be started.
If you have one diskdrive only, the program will tell you when to put the master diskette and when to put the back up diskette into disk drive A. When the copying process is finished the DISKCOPY window appears again. Now you can generate additional copies or you chose the menu item EXIT to leave the program. Store the fischertechnik computing diskette in a safe place inaccessible to the natural enemies of diskettes such as sand, heat, cats or magnetic fields. Work with the copy and only use the original fischertechnik computing diskette to make another copy if necessary. Additional copies are especially useful if you want to change the programs while retaining the contents of the original diskette. You are restricted to making copies for your own use.
If you have not already connected the interface to your computer as described in the previous section, you should do so now. The fischertechnik computing programs are BASIC programs. They run under the BASIC-Interpreter of the company GfA Systemtechnik. This interpreter combines a relatively low number of bugs, a moderate memory useage, a high execution speed and an easy handling.
In case you do have the software package GfA-BASIC, load the GfA-BASIC interpreter. To do so insert the GfA system diskette in disk drive A. Now switch on in that order: the screen, eventually the second disk drive and the computer. Using the mouse select disk drive A. One of the programs shown in the diskette window is GFABASIC.PRG; choose that. Subsequently the editor of GfA-BASIC appears. Now put in the fischertechnik computing diskette and select menue item LOAD. The file selection window of the fischertechnik computing diskette appears.
At this place we will explain, how the starting procedure works, if you do not own GfA-BASIC. In this case insert the fischertechnik diskette right away and switch on your computersystem in the way described before. Select the icon for disk-drive A. The diskette window shows now the program GFABASRO.PRG besides the two folders ENGLISH and DEUTSCH and a couple of other datafiles. Select GFABASRO.PRG. GFABASRO stands for GfA-BASIC run-only interpreter. This interpreter only can execute programs; but it is not possible to alter or write programs, since it contains no editor. Instead the company GfA-Systemtechnik allows us to copy the run-only interpreter onto the fischertechnik computing diskette without any charges. In contrast to the complete GfA-BASIC interpreter the run-only interpreter jumps immediately into the program selection menu. The range of programs available among the two interpreters is the same, as well the behaviour of the programs themselves.
Since you as a fischertechnik-user certainly won't be satisfied just to run the programs but rather like to experiment with fischertechnik and the computer we recommend to purchase the GfA-BASIC interpreter.
Among the two interpreters (the complete on and the run-only interpreter) the following procedure is alike. Select the folder ENGLISH where upon the directory of this folder appears. The directory is in alphabetical order. Among the first entries you find the program FISCHER.BAS. Select that. Using the run-only interpreter the program starts immediately. Using the complete interpreter you have to give the command RUN (Hold down the shiftkey and press F10 or move the mouse cursor to the command RUN in the command list and press the mouse-button. A welcome text will be generated on the screen by the program; you should carefully study this text since it will contain some helpful advice to the specific software-packages and the respective fischertechnik computing kit.
At the end of the welcome program a new file selection is available. Select now the interface driver program DRIVER.BAS. After having started the driver program (either automatically or using the command RUN) the diskette drive will spin again briefly, since the driver program is loading the machine language program INTERFAC.COM which contains the above-mentioned detail knowledge about the input/output devices of the computer. Subsequently the headline of fischertechnik computing will be displayed on the screen and the end of the program will be announced.

No apparent changes have taken place. However, your Atari ST now has a number of new commands available which were not previously a part of BASIC in this form. These commands are taylored exactly to your Atari ST and the fischertechnik computing interface. They will use the C: command. The driver program itself is written in BASIC and generates the parameters of the machine language program calls in the variable storage memory of the Atari ST.
The only commands you need to be able to use are the following ones:

The motor output M1 is controlled with the following commands:

Dummy=C:M1(Cw)
(clockwise rotation)

Dummy=C:M1(Ccw)
(counter-clockwise rotation)

Dummy=C:M1(Off)
(turn off)

The variable Dummy is necessary here because C: is a function and delivers a value which has to be used up in an assignement. The proper purpose of the this function is the motor-control, the delivered value itself is not of importance. Therefore it is assigned to a variable, the value of which will never be used later in the program. For such variables the notion "dummy" has become common.

The command parameters designate the motor, motor 1 in this case, and the mode of operation.
The same command is used with the parameters M2, M3 and M4 to control the other three outputs.
The 10 inputs of the interface are also read with the aid of the C: function. The function

C:In(E1)

is 1 if +5 volts apply at input E1 of the interface. Otherwise C:In(E1) will output 0. Similarly, C:In(E2)...C:In(E8) will display the status of the other digital inputs.

The analog inputs EX and EY are each connected to +5 V via a potentiometer (4.7 kΩ). The functions

C:In(Ex)
C:In(Ey)

will then produce a value within the range between 20 and 280 according to the position of the potentiometer. The actual limits may slightly vary due to deviations within the scatter of the production of the interfaces. Analog inputs are mostly used for measuring positions. If, for example, a robot arm is driven by a motor, and potentiometer EX is adjusted synchronously with the motion of the arm, then the program, by repeatedly calling the function

C:In(Ex)

can trace the movement of the robot exactly.
If no potentiometer is connected to inputs EX or EY, there will be an overflow of the internal evaluation counter. The value of the function is then appr. 5000.

The last of the new commands is

CALL Init

This command is used to switch the interface to a well-defined initial state. The command can also be used to turn off all motor channels simultaneously. Enough of the long introduction. In case you have the complete GfA-BASIC interpreter you can try the commands in the direct mode. You switch to the direct mode by choosing the command Direct in the command list or by hitting the Esc-key. Connect a fischertechnik motor to M1 using the twenty-wire, color-coded ribbon cable. Use the yellow and the orange cables in the upper half of the ribbon cable. Type:

Dummy=C:M1(Cw)

The motor will briefly start up and then stop again. Congratulations! You've just learned how to control the most complicated fischertechnik models with your Atari ST.
However, you would probably like to find out why the motor has stopped. Didn't you turn it on? Isn't there a separate command for turning it off, as described above? Well – the motor was stopped, but the value stored in the interface says that it is still running. The interface itself has "gone to sleep". This always happens when another command is not received within half a second and is programmed this way for safety reasons. Let's say you are testing a new program. The probability that there is a mistake hidden away somewhere in the program borders on certainty. The program run is interrupted with a message such as

STOP Division by zero

The motor, which was turned on a short time ago, however, continues running and is about to demolish that beautiful model you worked so hard to build. You would have to run to the power supply unit and turn the power off.
How good it is to know, then, that the motor will stop by itself. Even if you have interrupted program execution by pressing Control-Shift-Alternate. And if the program is resumed, the first command "wakes up" the interface, and none of the motors has been forgotten. The procedure can go on as if nothing had ever happened.
There is a reason why the interface is turned off with a delay. Due to calculations, there will always be delays between input and output commands to the interface. These delays must be bridged. You can tell by looking at the light emitting diode (LED) on the interface whether the interface is being activated by input or output commands. The LED not only indicates voltage but also shows if the interface is operative.
Next we will take a brief look at the input commands. Connect a pushbutton switch between E1 (the brown wire on the lower edge) and +5 V (the red wire in the center of the ribbon cable).

Type in the following command:

PRINT C:In(E1)

Depending on whether or not the pushbutton switch between E1 and +5V was depressed when the return key on the computer was pressed, the screen displays a 1 or a 0.
Since the motor is still connected to the output from before, it will start up once more. Input commands will also reactivate the outputs of the interface! Connect a 4.7 kΩ potentiometer between EX and +5V. Adjust the slider to a medium position and type in

PRINT C:In(Ex)

The number which is displayed on the screen must be in the range between 0 and 280.
You will regularly need the driver program which you were using just now. Every program written to control models via the fischertechnik computing interface begins with this routine which installs the new commands. The driver program is already contained in all the sample programs on the diskette. All programs that you write yourself must start with the driver program and continue thereafter.
In order to be able to observe the potentiometer more easily, we will now write the first fischertechnik computing program in this manner. The driver program is still in the computer. Go back to the edit mode by pushing the key combination Control-Shift-Alternate. As well you may also type in the command Edit. Put the cursor at the end of the program behind the line reading:

Call Init

Type in:

Do
Print C:In(Ex)
Loop

Now start the program (press the keys Shift-F10 or click Run of the command list).

There may be a brief delay while the INTERFAC.COM file is being loaded. The question concerning the color screen settings will be displayed. And then it will begin: all at once the screen will be filled with numbers which will continuously scroll up. If you then take the potentiometer in your hand and move the slider, you will be able to observe the change in the numbers. Turn the potentiometer from one stop to the other. The numbers displayed should be between 20 and 280. Press Control-Shift-Alternate to interrupt the program.
For those who are interested in more detailed information about these processes and who do not merely wish to use the programs included on the diskette, we will provide some detailed information here. The function of the driver program is to write a short machine language program to a memory range of the Atari ST. This program is contained on the diskette as the INTERFAC.COM file. The machine language routine will be written in a range of memory which is reserved through the array **Interface%.** The starting address will be ascertained through the function **Arrptr.** In this way the functioning of the software in different memory equipment and different memory usage of the Atari ST can be guaranteed.
Besides the machine language program, the standardized parameters M1, M2, M3, M4, In, E1, E2, E3, E4, E5, E6, E7, E8, Ex, Ey, Cw, Ccw and Off are set. The following limitations apply when you write your own BASIC programs:
Since the mentioned parameters contain values, which are important for the correct functioning of the machine language program, those parameters may not be used as variables for other purposes. The same applies for the array **Interface%.** The restriction does not apply for variables of the same name but of different type. So

Off$, M1%, Interface . . .

could be used but we do not recommend it.

Those interested in further details, e.g. in the electronic processes in the interface or adaptations to other programming languages, should refer to the description of the interface and the source code.

The last function of the base program is to turn on the interface and turn off all connected loads. This is caused by issuing the following command:

500 CALL Init

This switches the interface to the ready state for a subsequent user program. All sample programs on the diskette are built according to this pattern. We suggest you study them for hints and ideas to use in designing your own programs.
Depending on the type of the fischertechnik computing kit, additional commands of that kind may be introduced. They are being used to run more complex and time critical sequences in machine code. A good example for this are the commands for the fischertechnik training robot.

The Diagnostic Program

```
Rem *************************
Rem *** Interface-Driver  ***
Rem ***          for      ***
Rem ***      ATARI ST     ***
Rem *************************
Rem
Res=Xbios(4)
If Res=0
  Let M$="The Program needs the|80-Character-Display"
  Alert 0,M$,1,"Return",B
  End
Endif
Dim Interface%(90)
Let Init=Lpeek(Arrptr(Interface%()))+2
Bload "INTERFAC.COM",Init
Let M1=Init+&H8
Let M2=Init+&HC
Let M3=Init+&H10
Let M4=Init+&H14
Let E1=1
Let E2=2
Let E3=4
Let E4=8
Let E5=16
Let E6=32
Let E7=64
Let E8=128
Let Ex=160
Let Ey=144
Let Cw=85
Let Ccw=170
Let Off=255
Let In=Init+&H5A
Setcolor 0,7,7,7
Setcolor 1,0,0,7
Setcolor 2,7,0,0
Setcolor 3,0,0,0
Deftext 3,0,0,16*Res
Text 64,16*Res,112,"fischer"
Deftext 2
Text 176,16*Res,112,"technik"
Print At(37,1);"®";
Deftext 1,16
Text 64,34*Res,224,"COMPUTING"
Deftext ,0,900,4*Res
Text 16,184*Res,"Copyright (c) fischerwerke 1987"
Deftext ,,0,9.5*Res
Call Init
```

If you have constructed a fischertechnik computing model, you may find that things do not work as you had expected. No wonder considering the great number of lines running between the model and the interface. If only one pushbutton switch were incorrectly wired, this could produce the most astounding results. Matters become twice as complicated when you write your own programs. Where do you start looking for errors? Is it a hardware problem? Is the software to blame?

In order to help you test the hardware clearly and conveniently, we have developed the diagnostic program. It is included on the fischertechnik computing diskette as the DIAGNOST.BAS file. Always load this program for testing a model. It allows you to monitor all the inputs and to verify whether their behavior corresponds to your expectations.

While the digital inputs are displayed permanently, the values of the two analogue inputs have to be interrogated when required. To do so push the space bar. The reason for this proceeding is the mutual obstacle of mouse use and analogue reading. Therefor the analogue reading shouldn't be recalled permanently. More to that in the chapter which explains the use of mouse and interface.

The motor control outputs are selected using the mouse. A box is assigned for each motor. In the respective box is the symbol for motor stop in the middle, the symbols to the right initiate clockwise, the symbols to the left initiate counter-clockwise rotation. Additionally you may choose between two running conditions. The dotted symbol controls motor operation only as long as the mouse button is pressed. The solid symbol controls continous running. You may select it if e.g. a magnet should be switched on, while the robot axis will be moved.

To switch off all motors simultaneously there exists a big emergency-off-box. With a final box you may select the end of the program.

```
Rem ***************************************
Rem *                                     *
Rem *      fischertechnik computing       *
Rem *                                     *
Rem *         Diagnostic program          *
Rem *                                     *
Rem *   Copyright (c) fischerwerke 1987   *
Rem *                                     *
Rem ***************************************
Rem
Rem Function:
Rem
Rem Using this program you may control
Rem al functions of a model.
Rem All inputs are displayed on screen
Rem (for analog values you must press the space bar).
Rem Alle outputs are controlled via the mouse.
Rem
Rem *************************
Rem *** Initialize control ***
Rem *************************
Rem
Gosub Sprite_data
Gosub Screen_display
Rem switch all motors off
Rem Status of the motors
Dim Sta(4),X(4)
For I=1 To 4
  Let Sta(I)=Off
Next I
Rem Adresses for Motor 1 thru 4
Dim M(4)
Let M(1)=M1
Let M(2)=M2
Let M(3)=M3
Let M(4)=M4
Rem Start with motor 1
Let Mot=1
Rem
Rem ************************
Rem *** Display of input ***
Rem ************************
Rem
Do
  Print At(12,13);C:In(E1)
  Print At(15,13);C:In(E2)
  Print At(18,13);C:In(E3)
  Print At(21,13);C:In(E4)
  Print At(24,13);C:In(E5)
  Print At(27,13);C:In(E6)
  Print At(30,13);C:In(E7)
  Print At(33,13);C:In(E8)
  A$=Inkey$
  If A$=Chr$(32)
    Print At(36,13);Using "###",C:In(Ex)
    Print At(41,13);Using "###",C:In(Ey)
  Endif
  Rem *************************
  Rem *** Control of output ***
  Rem *************************
  Rem
  Rem Status of the mouse
  Mouse X,Y,K
  If Res=1
    Let Y=Y*2
  Endif
```

```
Rem Selected motor
Zu=Sta(Mot)
If K
  If Y>240
    If Y>320 And X<316
      Let Mot=2
    Endif
    If Y>320 And X>328
      Let Mot=4
    Endif
    If Y<304 And X<316
      Let Mot=1
    Endif
    If Y<304 And X>328
      Let Mot=3
    Endif
    If Abs(X-188)<=20 Or Abs(X-452)<=20
      Let Sta(Mot)=Off
      Let X(Mot)=0
      Let Zu=Off
    Endif
    If Abs(X-92)<=20 Or Abs(X-356)<=20
      Let Sta(Mot)=Ccw
      Let X(Mot)=96
      Let Zu=Ccw
    Endif
    If Abs(X-288)<=20 Or Abs(X-548)<=20
      Let Sta(Mot)=Cw
      Let X(Mot)=-96
      Let Zu=Cw
    Endif
    If Abs(X-140)<=20 Or Abs(X-404)<=20
      Let Zu=Ccw
    Endif
    If Abs(X-236)<=20 Or Abs(X-500)<=20
      Let Zu=Cw
    Endif
  Else
    If X>376
      If Abs(Y-208)<=16
        Call Init
        For I=1 To 4
          Let Sta(I)=Off
          Let X(I)=0
        Next I
        Let Zu=Off
      Endif
      If Abs(Y-160)<=16
        Call Init
        Stop
      Endif
    Endif
  Endif
Endif
Rem output to selected motor
Let M=M(Mot)
Dummy=C:M(Zu)
Sprite M1$,180-X(1),138*Res-12
Sprite M2$,180-X(2),178*Res-12
Sprite M3$,444-X(3),138*Res-12
Sprite M4$,444-X(4),178*Res-12
Loop
End
Rem
Rem **********************
Rem *** Screen display ***
Rem **********************
Rem
Procedure Screen_display
  Deftext 1,0,0,9.5*Res
  Text 64,64*Res,"Diagnostic program"
  Deffill 1,0
  Prbox 64,72*Res,360,112*Res
  Deftext ,,,6.5*Res
  Print At(10,11);" E1 E2 E3 E4 E5 E6 E7 E8    Ex    Ey";
  Prbox 376,72*Res,576,88*Res
  Text 416,83*Res,"End of program"
  Prbox 376,96*Res,576,112*Res
  Text 400,107*Res,"All motors off"
  Prbox 64,160*Res,316,192*Res
  Prbox 64,120*Res,316,152*Res
  Prbox 328,160*Res,576,192*Res
  Prbox 328,120*Res,576,152*Res
  Defline 1,1,0,1
  Circle 356,178*Res,20,2250,1800
  Circle 356,138*Res,20,2250,1800
  Circle 92,178*Res,20,2250,1800
  Circle 92,138*Res,20,2250,1800
  Defline 3
  Circle 404,178*Res,20,2250,1800
  Circle 404,138*Res,20,2250,1800
  Circle 140,178*Res,20,2250,1800
  Circle 140,138*Res,20,2250,1800
  Defline 1,,,0
  Circle 452,178*Res,20
  Circle 452,138*Res,20
  Circle 188,178*Res,20
  Circle 188,138*Res,20
  Defline 3,,1
  Circle 500,178*Res,20,0,3150
  Circle 500,138*Res,20,0,3150
  Circle 236,178*Res,20,0,3150
  Circle 236,138*Res,20,0,3150
  Defline 1,,1
  Circle 548,178*Res,20,0,3150
  Circle 548,138*Res,20,0,3150
  Circle 284,178*Res,20,0,3150
  Circle 284,138*Res,20,0,3150
  Sprite M1$,180,138*Res-12
  Sprite M2$,180,178*Res-12
  Sprite M3$,444,138*Res-12
  Sprite M4$,444,178*Res-12
Return
Rem Sprite-Data
Procedure Sprite_data
  Let M1$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M1$=M1$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Bottom
    Let M1$=M1$+Mki$(Bottom)
  Next I%
  For I%=1 To 16
    Read Top
    Let M1$=M1$+Mki$(Top)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,49926,59150,65310,56118,49958,49926,49926
  Data 49926,49926,49926,49926,49926,49926,49926,49926
  Let M2$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M2$=M2$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Bottom
    Let M2$=M2$+Mki$(Bottom)
  Next I%
  For I%=1 To 16
    Read Top
    Let M2$=M2$+Mki$(Top)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,49948,59198,65379,56129,49921,49921,49923
  Data 49926,49932,49944,49968,50016,50016,50047,50047
  Let M3$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M3$=M3$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Bottom
    Let M3$=M3$+Mki$(Bottom)
  Next I%
  For I%=1 To 16
    Read Top
    Let M3$=M3$+Mki$(Top)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,50047,59263,65283,56070,49932,49944,49982
  Data 49927,49923,49923,49923,50019,50019,50047,49982
  Let M4$=Mki$(0)+Mki$(0)+Mki$(1)
  Let M4$=M4$+Mki$(0)+Mki$(1)
  For I%=1 To 16
    Read Bottom
    Let M4$=M4$+Mki$(Bottom)
  Next I%
  For I%=1 To 16
    Read Top
    Let M4$=M4$+Mki$(Top)
  Next I%
  Data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  Data 0,49926,59142,65294,56078,49950,49942,49974
  Data 49958,50022,50022,50047,50047,49926,49926,49926
Return
```

GfA-BASIC

The fischertechnik computing programs on the diskette enclosed are, as mentioned before, written in GfA-BASIC. The programs documented in the fischertechnik computing instruction manuals, however, have been written in a BASIC of another computer. The difference in the program listings are considerable, especially because GfA-BASIC does not use line numbers as a mean for structuring the programs. Instead a number of efficient structure commands are available, which have been used optimally in the present software. The BASIC-typical GOTO has not been used in any case.
Also the possibilities to manipulate the screen display are much wider than in a "classical" BASIC. In addition it is possible to refer to the routines of the GEM operating system with its support of mouse and windows.

For those reasons the conception of the Atari ST software of the fischertechnik computing kits is not modelled on the original, but takes advantages of the above-mentioned possibilities. The proper sequence of controlling of the models is still following the printed programs.

Here again we confront the most important differing commands:

SYS M1, CW	**Dummy=C:M1(Cw)**
SYS INIT	**Call Init**
PRINT USR(E1)	**Print C:In(E1)**
PRINT CHR$(147)	**Cls**
GET A$	**A$=Inkey$**

GfA-BASIC simplifies the controlling of the print position, since the print command allows to specify the print position in the way:

Print At(Column,Line)

Also the printing format can be fixed through a format statement like:

Print Using(. . . .)

The headlines can be designed with the commands

Deftext and **Text**

With its help you can create different fonts and letter sizes.
Study the programs on the fischertechnik computing diskette to understand the programming in GfA-BASIC.

The use of the mouse and the interface

Using the mouse and the analogue input of the interface at the same time obstructs the processes in the computer a little bit. Skillful programming, however, can avoid it to a good part. Recording an analogue input value, the interface needs the full attention of the microprocessor. Therefore the flow of information from the keyboard to the microprocessor has to be stopped. During the evaluation of the analogue input value the computer cannot recognize any pushed key or movement of the mouse. For the keyboard this is not that restrictive, since a number of characters can be memorized there until the connection after finishing the analogue input is reinstalled. Fast mouse-movements, however, may exceed the capacity of the keyboard memory (the mouse is electrically connected to the keyboard). This leads to errror messages, which are indicated by a clicking sound of the loudspeaker. Those error messages do not hinder the program run.

Nevertheless we recommend to move the mouse only when the fischertechnik program requests a corresponding input. Writing your own programs avoid to request mouse and analogue input in fast changes but use them rather at different phases of the program. The fischertechnik computing programs may serve as example. In the diagnostic program e.g. the mouse is constantly requested, the analogue input, however, only when pushing the space-bar. In the program TIR_MOUS.BAS the next robot position will be requested using the mouse. Only thereafter starts the position control routine measuring the potentiometer setting using the analogue input channels.

Furthermore we recommend to have a device connected to the analogue input channels at any time. In case the corresponding fischertechnik computing model does not use analogue sensors (i.e. potentiometer, light-dependent resistors etc.) so connect the inputs EX and EY just with the reference voltage

+5V. This corresponds to a resistance value of zero Ohms. In case the analogue input gets a voltage spike the data lines between the Atari ST and the interface won't be blocked for an unnecessary length of time.

Check List

Should you ever have a problem with the fischertechnik computing interface, and it does not work as you expect it to, then check the following points with the diagnostic program:

The diagnostic program displays 0 for E1 through E8, although a model is connected. – The interface has not been connected to the computer or to the power supply unit.

One of the inputs E1 through E8 shows the reverse effect from what you would expect when the pushbutton switch is actuated. – Opening and closing functions of the pushbutton switch have been reversed.

One of the inputs E1 through E8 always shows 0, although it is connected and has been actuated. – Check for a broken or improperly connected wire.

One of the inputs E1 through E8 always shows 1, even though no model is connected. – The input gate IC 4014 has probably been damaged by overvoltage (electrostatic discharge).

A motor output does not work. – Check for a broken or improperly connected wire.

A motor output only works in one direction of rotation. – The power stage of the motor output is defective.

The motor rotates slowly or works intermittently. – Either the power supply unit is overloaded by too many motors (use two mot 4 power supply units or the more powerful computing power supply unit) or the power supply unit has not been turned up all the way when using the adjustable output.

In case of defect, send the unit to your dealer or the company representing and importing fischertechnik.

Technical Data

fischertechnik computing interface to be connected to parallel printer points in Centronics standard, Part No. 30566. Amongst other computers the interface can be used with the Atari 260 ST, Atari 520 ST, Atari 1040 ST, Atari Mega ST and respective variants.

Four outputs for connection of motors, lamps, electromagnets, etc. (M1 to M4). Polarity of output (direction of rotation) controllable.

Load capacity: 1 A continuous, 1.5 A peak.

Eight inputs for digital signals (E1 through E8). Due to internal wiring, the connection of electromagnetic devices (pushbutton switches, switches, relays) in positive logic and the connection of TTL outputs is also possible. Built-in overvoltage protection.

Two inputs for analog signals (EX and EY). Transmitters with resistances between 0 and 5 kΩ, e.g. potentiometers, photoresistors, etc., can be connected.

Circuitry for monitoring the stream of data. If no data signals are received from the Atari ST computer, the interface will disable all outputs after 0.5 seconds. The signals are stored in memory.

Circuitry for monitoring the software. Grave errors will trigger the guard circuit, in this case without delay.

The guard circuitry will also react to an insufficient supply of power, be it through overload or insufficient voltage from the power supply unit.

Software suitable for the respective fischertechnik computing kit and the Atari ST, as well as a suitable adaptor for the printer port will be delivered as a separate set without any extra charges.

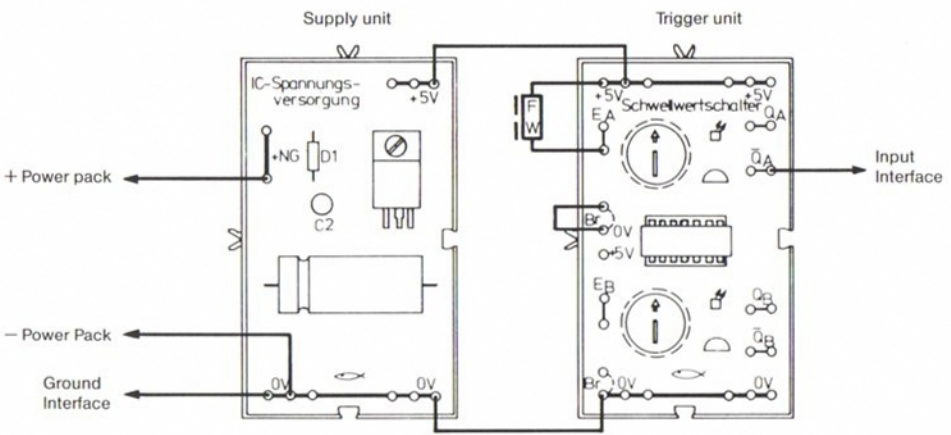
Using the fischertechnik Electromechanics and Electronics Kits

The fischertechnik computing interface is compatible with the components and electronic devices of the above-mentioned kits. Other types of pushbutton switches and switches may be used in place of the mini pushbuttons used with the fischertechnik computing models, e.g. the large pushbutton or the pole changing switch, but also the reed contact or the switching contact of a relay. Be careful when using pushbuttons and switches which you have constructed yourself from articulated elements and springs – they could cause contact bounce. In such cases, we recommend having the program check the input several times and consider the result valid only if the same value appears twice in a row.

The analog inputs of the interface can be connected to sensors with an output resistance between 0 and 5 kΩ. The potentiometers from the fischertechnik computing kit may be used, of course. Other devices, such as the photoresistor, may be used as well.

The motor outputs of the interface feature high load capacity. Not only the mini motors, but also the S-motor and the N-motor can be controlled via the interface, whereby one lamp may be connected in parallel as a function indicator. Besides motors, the electromagnet and the relay RBII are also suitable.

Signals from the electronic modules with integrated circuits of the TTL family (e.g. trigger unit) may also be fed to the interface inputs. First, however, the common point of reference must be established by connecting the ground circuit of the electronic module to the ground jack of the interface. The figure shows the design of a light barrier. The trigger unit is used to adjust the reaction threshold of the light barrier.



Operation of the Interface and the Interface Driver Program

If you use the fischertechnik computing software or write programs yourself according to the notes in the previous sections, most likely you will not need the information that follows. If, however, you intend to write the programs in a language other than BASIC, would like to speed them up through complex procedures in machine language, wish to extend the functions of the interface or simply want to glimpse behind the scenes, then the following information will most certainly be helpful. In this case, however, you should have a basic knowledge of machine language and digital electronics, since this is about the "bits and pieces".

The fischertechnik interface handles a number of tasks which we would like to discuss with the aid of the block diagram. On the left side you see the signals from and to the computer. Note how little they have in common with outputs M1 to M4 and inputs E1 through E8 and EX and EY. The reason for this is that the number of data lines available at the computer port is significantly lower than the number of lines required on the model side of the interface. This limited number of data lines must therefore be employed in such a way as to control all signals on the model side. The concept employed is that of multiple use of the data lines with the aid of shift registers. In this way, for example, only three data lines are required for controlling the output. A parallel connection scheme would have required eight data lines. Let's take a closer look at the output at connections M1 to M4. The data lines required are designated DATA OUT, CLOCK and LOAD OUT. If there is an output, the data for all four motors are transmitted in each case, i.e. a whole byte (a byte because each of the four motors requires two bits for controlling the direction of rotation). The motor outputs to which the signal does not apply are thus once again supplied with the current state which is buffered in the computer as an output word.
For output, the bits of the output word are sequentially (with the most significant bit first) fed to the DATA OUT line. When the signal at the CLOCK output goes from low to high, the bit is transferred to a shift register. Then the next bit at DATA OUT follows, and is likewise transferred to the shift register with the next CLOCK pulse. The previous bit has been shifted one position to the right in the shift register in order to make room for the subsequent bit. After a total of eight such data transfers, the whole output word has been transferred to the shift register. The bit first transferred has been shifted all the way to the right in the course of the data transfer. Thus far, the activity in the shift register has not had any effect on its outputs. The output amplifiers are not controlled directly by the shift register, but rather via an in-line storage register which is integrated in the shift register module. Only when the LOAD OUT output goes from low to high are the data transferred to the storage register. The timing of the signals is shown in the pulse diagram.

Whether the data are fed to the power amplifiers, however, depends on the enabling control of the memory module. The enabling circuit is controlled by a monostable multivibrator. This circuit generates an enabling signal with a duration of half a second if there is a pulse on the CLOCK line. We may assume that the power amplifiers receive a signal first since the data were just transferred with the aid of the CLOCK line. If no more data are transmitted within the next half second, however, the monostable multivibrator will flip back to the stable state and the enabling signal is removed. The monostable multivibrator, by the way, can be retriggered, i.e. the time of half a second is always calculated from the time of the last CLOCK pulse.
The monostable multivibrator also has an enabling input. The output to the amplifiers can be immediately inhibited via this input. On the fischertechnik interface this occurs when an invalid data pattern, which would command the connected motor to simultaneously turn clockwise and counterclockwise, applies at the output of the storage register.
We will proceed with the transfer of the digital signals to inputs E1 through E8. Basically the input is a reversal of the output process described above. The output signal LOAD IN causes the transfer of the data applying at the inputs to the input shift register. This always involves all eight inputs, even though only one of them is to be interrogated. When applying to the shift register, each pulse of the CLOCK line will cause the transfer of one bit on the input line DATA IN, the bit from E8 first and the one from E1 last. By testing this line, the computer can "collect" the bits and reassemble them into a data word. The desired bit is subsequently filtered out and transferred to the BASIC program.
Since the same CLOCK line is used for data transmission as for output, the digital input will also activate the monostable multivibrator, which controls the enabling signal for the output data. Malfunctioning of the output shift register caused by the multiple function of the CLOCK line is not to be expected since the current output data are not contained in the output shift register, but in the storage register. The former is controlled by the CLOCK pulses, unlike the latter, which only reacts to the LOAD OUT signal.

That leaves the analog inputs EX and EY. The potentiometers or other variable resistors are used as the timing element in two additional monostable multivibrator circuits. A low resistance value is converted to a short pulse, a high resistance value to a pulse with a long duration. The pulse itself is triggered by the starting signals TRIGGER-X and TRIGGER-Y (with negative logic), respectively, and then appears on the COUNT IN line. A machine language program determines the pulse duration by means of the number of loops which can be executed during the duration of the pulse. This number is fed back to the BASIC program which calls this function. You can see that there is no direct relationship between the

analog value and the angle position of the resistance of the potentiometer. The clock rate of the processor, however, is involved. There is a linear relationship between the number determined in the end and the resistance. If required, this value must be converted into angular degrees or resistance values by means of calibration.

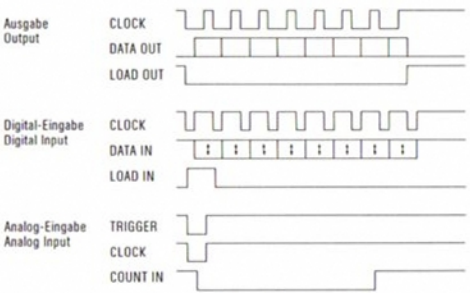
At this point, we will briefly review the connection between the interface and the Atari ST. As you know from the section about connecting the interface, the computer's parallel printer port is used for this purpose. Out of the eight data lines of this interface, the lower six are used for the output signals discussed above. (Also refer to the following table.)

The input lines pose a slight problem in that the parallel printer interface has only one input line available applying to all PC's. This is the active signal from the printer, BUSY. There is no conflict, however, if all input lines are combined using an OR circuit. Since the machine language program "knows" which input function it has requested, it has the capability of interpreting the signals on this one input line correctly.

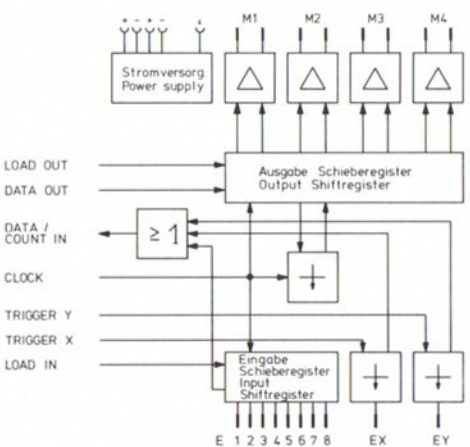
Connecting the fischertechnik Interface to the Parallel Printer Port

Interface Signal	Printer Signal	Pin Number
LOAD OUT	**Data bit 1**	**2**
LOAD IN	**Data bit 2**	**3**
DATA OUT	**Data bit 3**	**4**
CLOCK	**Data bit 4**	**5**
TRIGGER-X	**Data bit 5**	**6**
TRIGGER-Y	**Data bit 6**	**7**
DATA/COUNT IN	**Busy**	**11**

The following machine language program controls the processes described above. For the parameter transfer to the machine language program and back refer to the description of the CALL command or the C: function, respectively, in the GfA-BASIC handbook of your PC.



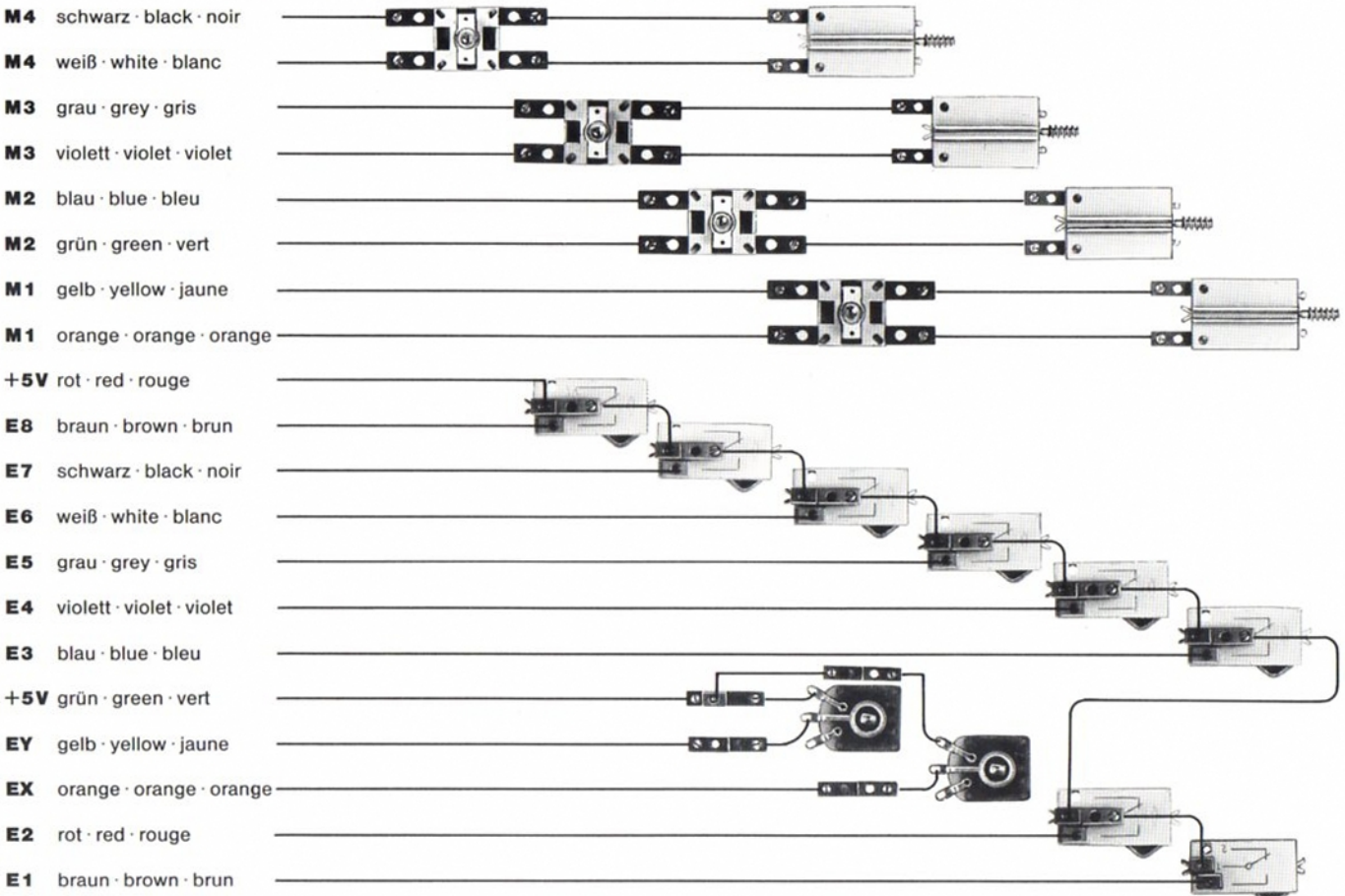
Impulsdiagramme des fischertechnik Interface
Pulse Diagrams of the fischertechnik Interface

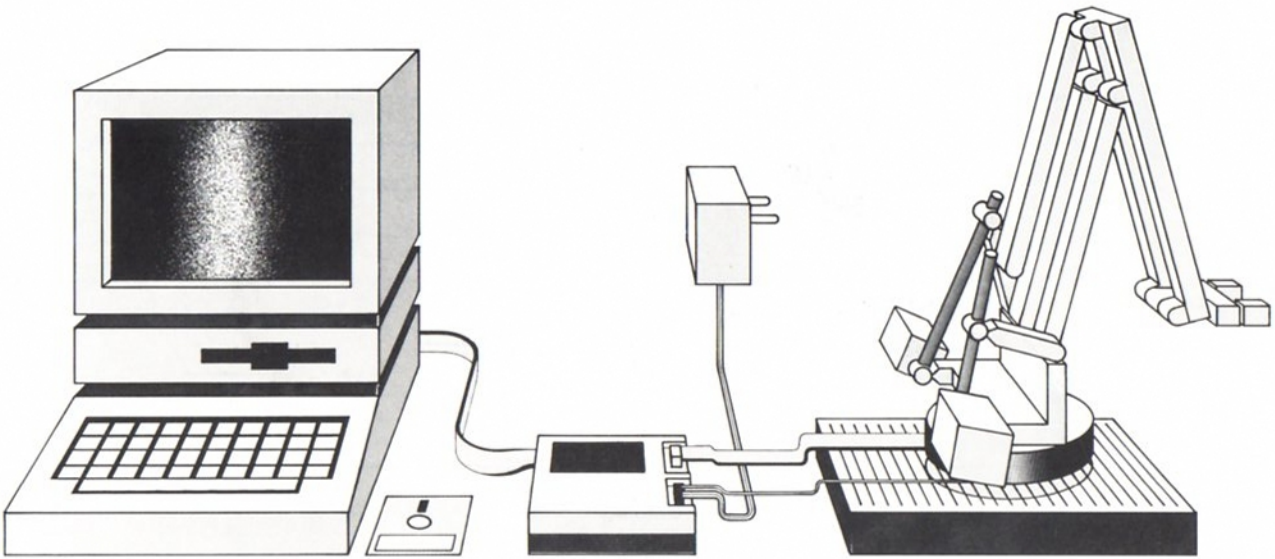



```
;****************************************
;* ATARI ST Interface Driver Routines   *
;* Version 1                            *
;* Copyright (C) fischerwerke 1987      *
;* Control of the fischertechnik Interface*
;* output and input using the C: function.*
;*                                      *
;****************************************
;* This program requires a Centronics   *
;* parallel printer interface as I/O-port.*
;*                                      *
;* The machine language program is located*
;* in an integer-array (will be erased by *
;* NEW and CLEAR!).                     *
;* First command must be CALL init.     *
;*                                      *
;* Output control:                      *
;*                                      *
;* dummy=C:motornumber(operation)       *
;* Motornumbers are  M1,M2,M3 and M4    *
;* Operation is cw (clockwise), ccw     *
;* (Counter-clockwise)  and off.        *
;*                                      *
;* CALL init                            *
;* Initializes the interface and switches *
;* off all motors.                      *
;*                                      *
;* Input control:                       *
;* Digital input commands:              *
;* X=C:In(Digital-input-line)           *
;* Digital-input-lines are E1 to E8 .   *
;* X contains the level of the input line *
;* (either 0 or 1).                    *
;*                                      *
;* Analog input commands:               *
;* X=C:In(Analog-input-line)            *
;* Analog-input-lines are EX and EY.    *
;* After the use of the analog-input    *
;* commands X contains the measured value.*
;****************************************
;* Initializing-routine                 *
;* Start with CALL init                 *
;****************************************

        section status

init:   clr.l   d7
        lea     avar,a4
        bra.s   stvar
m1:     moveq   #$03,d6
        bra.s   bout
m2:     moveq   #$0c,d6
        bra.s   bout
m3:     moveq   #$30,d6
        bra.s   bout
m4:     moveq   #-64,d6
;*************************************
;*      singlebit output             *
;*************************************
bout:   clr.l   d7
        move.w  $04(sp),d5
        lea     avar,a4
        move.b  (a4),d7
        or.b    d6,d7
        and.b   d6,d5
        eor.b   d5,d7
```

```
stvar:  move.b  d7,(a4)
        bsr     disabl
        bsr.s   shout
        bsr     enable
        rts
;**************************************
;* Routine for interface control      *
;* output control                     *
;* output bit pattern in d7           *
;* Routine uses d5,d6,d7.             *
;**************************************
shout:  moveq   #8,d6
loop:   moveq   #$30,d5
        rol.b   #1,d7
        bcc.s   null
        or.b    #4,d5
null:   move.b  d5,$02(a6)
        or.b    #8,d5
        move.b  d5,$02(a6)
        subq.b  #1,d6
        bne.s   loop
        move.b  #$39,$02(a6)
        rts
;**********************************
;* input routine                  *
;**********************************
inp:    move.w  $04(sp),d4
        bsr.s   disabl
        cmp.w   #$a0,d4
        beq.s   poti
        cmp.w   #$90,d4
        beq.s   poti
        bsr.s   shin
        and.w   d4,d7
        beq.s   basret
        move.w  #1,d7
basret: bsr     enable
        clr.l   d0
        move.w  d7,d0
        rts
;**************************************
;* Routine for interface control      *
;* input control                      *
;* input-mask in d7                   *
;* routine uses d5,d6,d7              *
;**************************************
shin:   moveq   #$32,d6
        move.b  d6,$02(a6)
        or.b    #8,d6
        moveq   #8,d5
loop2:  move.b  d6,$02(a6)
        asl.b   #1,d7
        btst    #0,(a5)
        bne.s   null2
        or.b    #1,d7
null2:  move.b  #$30,$02(a6)
        move.b  #$38,$02(a6)
        subq.b  #1,d5
        bne.s   loop2
        rts
;**********************************
;* Analog input                   *
;**********************************
poti:   clr.l   d7
        move.b  d4,$02(a6)
        move.b  #$38,$02(a6)
loop3:  btst.b  #0,(a5)
        bne.s   stop
        addq.w  #1,d7
        bne.s   loop3
stop:   bsr.s   enable
        clr.l   d0
        lsr.w   #1,d7
        move.w  d7,d0
        rts
;**********************************
;* disable all interrupts         *
;* Routine uses d0                *
;**********************************
disabl: clr.l   -(sp)
        move.w  #$20,-(sp)
        trap    #1
        add.l   #6,sp
        ori.w   #$0700,sr
        movea.l sound,a6
        move.b  #7,(a6)
        bset.b  #7,$02(a6)
        move.b  #15,(a6)
        movea.l mfp,a5
        bclr.b  #5,$08(a5)
        bclr.b  #0,$04(a5)
        rts
;**********************************
;* Enable all interrupts          *
;* Routine uses d0                *
;**********************************
enable: bset    #5,$08(a5)
        andi.w  #$3300,sr
        move.l  d0,-(sp)
        move.w  #$20,-(sp)
        trap    #1
        add.l   #6,sp
        rts

        section data

avar:   ds.b    1
sound:  dc.l    $ff8800
mfp:    dc.l    $fffa01

        end
```

Verdrahtungsplan der Interface Ein- und Ausgänge · Circuit layout of the Interface Inputs and Outputs





39483 · 0787 ST · Printed in Germany · Technische Änderungen vorbehalten