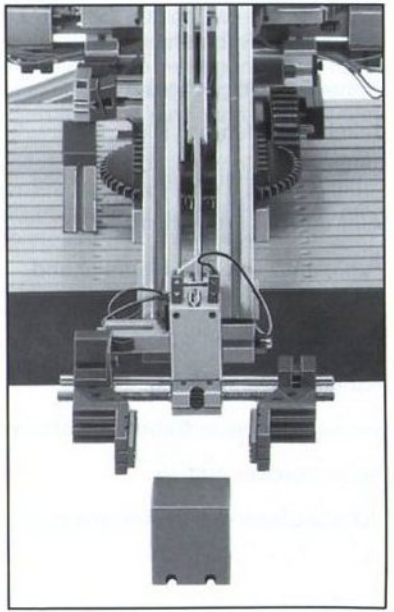
fischertechnik®
COMPUTING

Bauanleitung Trainingsroboter · Instructions Training Robot · Mode d'emploi du robot d'entraînement



Inhalt

Einführung	3
Was ist ein Roboter?	4
Antriebs- und Positioniersystem	5
Interface und Software	6
Roboter-Systemprogramm	8
Erste Experimente	9
Aufbau des Roboters	10
Steuerung des Roboters	11
Teach-In Verfahren	14
Weitere Experimente	15
Abdruck der Programme	16
Funktionsweise des Interface und Roboter-Systemprogramms	21
Bebilderte Bauanleitung	61
Kabelkonfektionierung	62
Verdrahtungsplan Gabellichtschranke	63
Mechanischer Aufbau	64
Verdrahtungsplan Trainingsroboter	91

Table of Contents

Introduction	33
What is a Robot?	34
The Driving and Positioning System	35
Interface and Software	36
The Robot System Program	38
First Experiments	39
Assembly of the Robot	40
Control of the Robot	41
Teach-in Procedure	44
Further Experiments	45
Print out of the Programs	46
Operation of the Interface and the Robot System Program	51
Illustrated Assembly Instructions	61
Ribbon Cable Configuration	62
Circuit Layout Photo-Interrupter	63
Mechanical Assembly	64
Circuit Layout Training Robot	91

fischertechnik Trainingsroboter

Lieber fischertechnik-Freund,

kaum ein technisches Instrument läßt sich so vielfältig einsetzen, wie ein Computer. Eines der reizvollsten Gebiete der Computertechnik ist jedoch die Steuerung technischer Modelle. Mit dem fischertechnik computing Bausatz Trainingsroboter haben Sie ein Modell erworben, das Sie in das anspruchvollste Gebiet der Steuerungstechnik, die Robotik, einführt.
Der Bausatz verbindet verschiedene Forderungen in einem leistungsfähigen Instrument. Zu allererst soll der Roboter realistisch sein. Aus diesem Grund hat der Trainingsroboter keinerlei Ähnlichkeit mit Science-fiction-Robotern, jenen menschenähnlichen Gestalten in Metall. Vielmehr lagen Konstruktionsskizzen heutiger Industrieroboter dem Entwurf zugrunde. Sie können aus dem Bausatz das Modell eines dreiachsigen Industrieroboters mit rotatorischen Bewegungsachsen aufbauen. Was dies genau bedeutet, werden Sie in dem Anleitungsbuch noch lesen.
Zum zweiten sollen Sie nicht nur den Roboter aufbauen und mit Ihrem Heim- oder Personal-Computer steuern können. Sie sollen auch die Möglichkeit haben, die Abläufe zu verstehen. Daher sind die Programme so ausgelegt, daß sie zum überwiegenden Teil in BASIC geschrieben und reichlich dokumentiert wurden. Das Verständnis der Programme ist auch die Voraussetzung, eigene Ideen und Vorstellungen zu realisieren. Ich möchte behaupten, daß der Nutzen des Roboters an dieser Stelle zuvörderst zum Tragen kommt.
Ändern, Experimentieren, Ausbauen, Programmieren ... hier spielt Ihre eigene Kreativität die entscheidende Rolle.
Dieser dritten Anforderung an den Roboter kommt vom mechanischen Aufbau her das fischertechnik-System in idealer Weise entgegen. Gleichgültig, ob Sie die Funktionen der Greifhand ausbauen, den Roboter mit Sensoren versehen oder eine Umgebung zur Computer-integrierten Fertigung aufbauen – die Vielfalt und hohe Präzision der fischertechnik Bauelemente wird Ihnen in allen Bereichen weiterhelfen.
Zur hohen Funktionszuverlässigkeit des Trainingsroboters tragen auch neue Bauelemente bei. An erster Stelle ist hier das Antriebs- und Positioniersystem zu erwähnen. Neue kompakte Gleichstrommotoren mit hoher Leistung verhelfen dem Roboter zu schneller Bewegung. Doch ist es mit der Steuerung des Roboters alleine nicht getan. Das Programm im Computer muß die Möglichkeit haben, die einmal erreichte Stellung abzufragen. Hierzu dienen die eigens entwickelten Gabellichtschranken. Sie durchleuchten mit Infrarotlicht das mit der Antriebsachse verbundene Rad. Die dort angebrachte Segmentierung erzeugt bei Bewegung letztlich Impulse, die über das fischertechnik Interface dem Computer zugeführt werden. Um diese schnelle Impulsfolge verwerten zu können, wird ein speziell für den Trainingsroboter entwickeltes Softwarepaket eingesetzt. Dieser Teil ist in der Maschinensprache Ihres Computers geschrieben und kann ohne Zählverluste selbst die gleichzeitige Bewegung aller Robotergelenke überwachen. Diese Programme werden von BASIC aufgerufen und sind somit problemlos zu benutzen.
Ich bin sicher, daß der fischertechnik computing Trainingsroboter Sie zu einer Reihe eigener Experimente anregen und Ihr Wissen und Ihre Erfahrung auf diesem Gebiete erheblich erweitern wird.

Ihr

Was ist ein Roboter?

Der Wunsch Roboter zu bauen, ist schier so alt wie die Menschheit. Wie so oft in der Geschichte unserer Kultur finden wir die Wurzeln des Roboters auch schon im antiken Griechenland. Mechanische Tempeldiener wurden dort erstellt, die vorgegebene Bewegungen ausführen konnten. Gesteuert wurden diese durch das Gewicht des herabrieselnden Sandes einer großen, im Roboter eingebauten, Sanduhr. Mit der Blütezeit der Handwerkskunst in der Neuzeit erregten auch immer wieder mechanisch gesteuerte Puppen die Bewunderung der Zeitgenossen und lassen auch uns Zeuge des Geschicks ihrer Erbauer werden. Ein schachspielender Automat zeigte jedoch die Grenzen der damaligen Möglichkeiten: In ihm verbarg sich ein kleinwüchsiger Mensch, der geschickt über Hebel und Gestänge die schachspielende Puppe steuerte. Unser Jahrhundert hat erst die Wortschöpfung „Roboter" erlebt. In dem Bühnenstück R.U.R. (Rossum's Universal Robot) des tschechischen Autors Karel Capek wurden Maschinensklaven „Roboter" genannt; abgeleitet von dem slawischen Wort „robota" für „schwer arbeiten". In dieser gedanklichen Linie bleibt auch häufig unsere heutige Vorstellung eines Roboters, von einer Maschine mit menschenähnlicher Gestalt, mittlerweile nicht mehr mit Sanduhr oder Federwerk sondern mit einem Computer als Gehirn ausgestattet. Genährt wird diese Vorstellung durch eine vielfältige Science-fiction-Literatur.

Daneben erleben wir in der Praxis unserer Arbeitswelt eine andere Art von Roboter. Es handelt sich hier um „... universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegung hinsichtlich Bewegungsfolge und -wegen bzw. -winkeln frei programmierbar und gegebenenfalls sensorgeführt sind" (VDI-Richtlinie). Roboter können je nach Ausstattung z.B. mit Greifern Werkstücke umsetzen, mit Punktschweißzange oder Lackierpistole Fertigungsaufgaben durchführen, aber auch eine Vielzahl anderer Handreichungen vornehmen.

Trotz der nüchternen Sichtweise der industriellen Praxis kommt doch auch bei diesen Instrumenten die Menschenähnlichkeit wieder ins Spiel. Gerade die universellsten der Roboter lehnen sich an das Vorbild der Natur an. Ihr Bewegungsapparat gleicht häufig einem menschlichen Arm. Deutlich sind vier Teilstücke des Roboters auszumachen: Die Grundplatte entspricht dem Körper, darauf folgt der Oberarm, an den sich nach dem Ellenbogengelenk der Unterarm anschließt. Den Abschluß bildet die Greifhand. Wir werden in der Folge diese menschenbezogenen Begriffe benutzen, um Ihnen eine schnelle Orientierung zu ermöglichen.
Wir wollen nicht verhehlen, daß auch andere Bauformen von Industrierobotern üblich sind. In beliebiger Kombination können die Drehachsen des Roboters auch durch Verschiebungen längs einer Führung ersetzt werden. Erfolgen im Extremfall alle Bewegungen längs einer Bahn, so erhalten wir ein Gerät, das mehr einem Portalkran als einem Arm entspricht. Dieser Robotertyp heißt dann auch Portalroboter und ist meist besser für hohe Lasten geeignet. Auch bei diesem Robotertyp werden die Verschiebemöglichkeiten als Bewegungsachsen bezeichnet, obwohl eine Drehachse im strengen Sinne nicht vorliegt.
Den Robotern ist gemeinsam, daß zur Positionierung der Greifhand immer drei Bewegungsachsen erforderlich sind. Bei einem Knickarmroboter ist dies erst nach längerem Ausprobieren einzusehen. Dagegen leuchtet diese Tatsache bei dem Portalroboter sofort ein. Er kann sich längs der drei Raumdimensionen Höhe, Breite und Tiefe bewegen.

Sie werden vielleicht schon gehört haben, daß Industrieroboter fünf und mehr Achsen haben. In seltenen Fällen kommt diese größere Zahl von Achsen der Bewegungsapparatur zugute, z.B. um in die Ecke einer Autokarosserie hineingreifen zu können. Meist dienen die vierte und fünfte Achse dem Schwenken des Handgelenks. Auf diese Weise kann die Griffrichtung verändert werden. Diese Möglichkeiten finden Sie beim fischertechnik Trainingsroboter nicht eingebaut, er enthält nur die drei Hauptachsen zur Bewegung. Allerdings stellt dies kein Nachteil dar, da alle Studien zur Robotergeometrie auch an den drei Hauptbewegungsachsen angestellt werden können. Zudem ist die Greifhand mit einem mechanischen Lageausgleich ausgestattet. Vielleicht ist es aber ein für Sie interessantes Unterfangen, die zusätzlichen Achsen zur Orientierung der Greifhand noch anzubauen? Doch zunächst soll das Grundmodell erstellt werden.

Das Antriebs- und Positioniersystem

Zum Antrieb des Roboters dienen Gleichstrommotoren. Die drei größeren Motoren bewirken die Roboterbewegung, der kleinere Motor das Öffnen und Schließen der Greifzange. Gleichstrommotoren haben den Vorteil, daß sie einfach anzusteuern sind und ein hohes Drehmoment bei geringem Eigengewicht und -volumen erbringen. Sie tragen letztlich zu der schnellen Bewegung des Roboters bei. Wir können die Motoren vor dem Einbau testen, indem wir sie direkt an das Netzgerät anschließen. Da im nachfolgenden Betrieb des Roboters bis zu vier Motoren gleichzeitig im Betrieb sein können, muß ein entsprechend belastbares Netzgerät verwendet werden. Wir empfehlen das fischertechnik computing Netzgerät oder aber die Verwendung von zwei fischertechnik Netzgeräten mot 4.

Gleichstrommotoren teilen mit den meisten anderen Motoren aber auch einen Nachteil. Sie können zwar über ein Getriebe den Roboter bewegen, jedoch ist nach Durchführung der Bewegung die Stellung des Roboters nur ungefähr bekannt. Lediglich die Laufdauer, während der der Motor angeschaltet ist, kann man über den Computer steuern. Andere Faktoren, wie die genaue Netzspannung und damit die Spannung des Netzgeräts, die Leichtgängigkeit des Roboters und der Lastwechsel an der Greifhand sind Faktoren, die zu einer Änderung der Motorgeschwindigkeit führen. Daher müssen gleiche Schaltdauern nicht unbedingt immer zu gleichen Bewegungen des Roboters führen. Ein solcher Zustand ist jedoch für ein Präzisionsinstrument wie einen Roboter untragbar. Daher muß unabhängig vom Motor die Position des Roboters noch einmal gemessen werden. Das hierzu dienende Positioniersystem besteht bei dem Trainingsroboter aus Gabellichtschranken. Bild 1 zeigt, wie eine solche Gabellichtschranke mit dem Motor und dem Getriebe verbunden ist. Die Abtriebswelle des Getriebes trägt ein becherförmiges Rad, auf dessen Umfang in regelmäßigem Abstand schwarze Linien aufgedruckt sind. Insgesamt sind es 32 Linien.

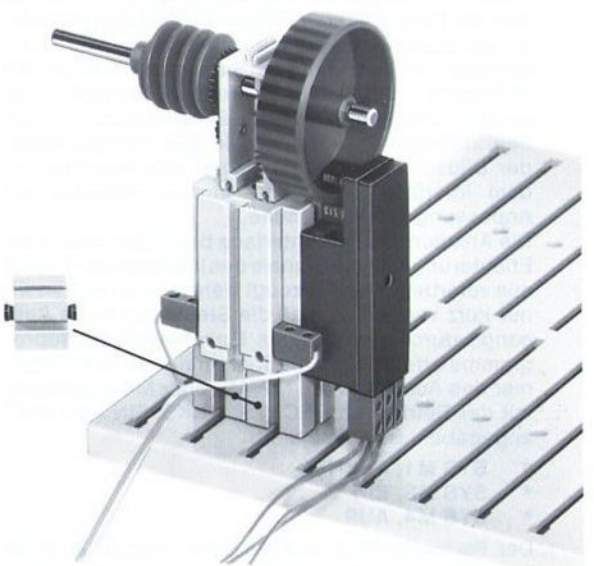
Die Gabellichtschranke greift nun über den Rand des Bechers. Auf der einen Seite der Gabel befindet sich eine Leuchtdiode, die Infrarotlicht sendet. Auf der anderen Seite der Gabel ist ein Fototransistor eingebaut, der auf Infrarotlicht empfindlich ist. Befindet sich kein Gegenstand zwischen der Gabel, so erscheint, vorausgesetzt daß die Betriebsspannung der Lichtschranke richtig angeschlossen ist (s.u.), an dem mit ⎍⎍ bezeichneten Ausgang ein High-Signal. Wird dagegen ein lichtundurchlässiger Gegenstand zwischen die Zinken der Gabel eingeschoben, so ist der Lichtstrahl unterbrochen. An dem Ausgang erscheint ein Low-Signal. Genauso reagiert die Lichtschranke auch auf die verschiedenen Zonen des Rades. Die schwarz bedruckte Stelle unterbricht den Lichtstrahl, die nicht bedruckten Stellen schwächen zwar das Infrarotlicht, lassen aber eine ausreichende Menge durch.

Wenn der Motor läuft, so erscheint an dem Ausgang der Lichtschranke eine Impulsfolge, die genau den dunklen und hellen Stellen des Rades entspricht. Nun sind wir in der Lage, den Roboter mit solch einem Aggregat präzise zu steuern. Wir messen nicht die Laufdauer des Motors, sondern zählen die Impulswechsel am Ausgang der Lichtschranke. Diese Zahl ist ein exaktes Maß für die Umdrehung der Abtriebsachse und damit für die Stellung des Roboters. Allerdings erfordert das Zählen der Impulse eine hohe Arbeitsgeschwindigkeit des Computers. Eine solche Aufgabe muß direkt in der Maschinensprache des Computers programmiert werden, wie weiter unten noch erläutert wird.

Im Moment wollen wir erste Erfahrungen mit dem Positioniersystem sammeln. Wie oben geschildert, muß der Infrarotstrahl das Plastikmaterial durchleuchten, nicht aber den schwarzen Aufdruck. Auch Sonnenlicht und das Licht von Neonröhren enthält Infrarotanteile, die natürlich störend wirken. Wir müssen daher den optimalen Arbeitspunkt der Lichtschranke suchen. Hierzu kann ihre Empfindlichkeit eingestellt werden. Mit dem beigefügten Schraubendreher können Sie durch das Loch in der Gehäuseoberseite ein Potentiometer verstellen. Gehen Sie dabei aber mit Vorsicht und Gefühl voran, um die elektronischen Bauteile nicht zu beschädigen.

Als Einstellhilfe finden Sie auf der Diskette bzw. Kassette das Programm ROBOT.JUST. Zu seiner Benutzung ist das fischertechnik Interface notwendig, so daß wir an dieser Stelle uns zunächst mit diesem wichtigen Instrument vertraut machen wollen.

Bild 1



Interface und Software

An dieser Stelle wollen wir eine kurze Bemerkung zu der Dokumentation der Programme bei fischertechnik computing einfließen lassen. Die Programme sind in dem Anleitungsheft in der Schreibweise des Commodore 64 abgedruckt. Mit dem Interface, das zu Ihrem Computer paßt, wird eine Diskette oder Kassette mitgeliefert, auf der die Programme auch vorliegen. Die BASIC-Schreibweisen der verschiedenen Computer unterscheiden sich leicht. Wenn Sie keinen Commodore 64, sondern einen anderen Computer haben, wird das Programm auf der Diskette oder der Kassette nicht ganz identisch mit dem hier abgedruckten Programm sein. Es ist schon an den entsprechenden Computertyp angepaßt. Die Stellen, wo sich auf jeden Fall Abweichungen ergeben, sind in dem Abdruck des Programms mit einem Sternchen vor der Zeile gekennzeichnet. Wenn Sie also die abgedruckten Programme mit den eingelesenen vergleichen oder das Programm von Hand eingeben wollen, müssen Sie an den Stellen mit Sternchen aufpassen. Außerdem können sich leichte Verschiebungen der Zeilennummern ergeben, die meist ihre Ursache in den Unterschieden der Bildschirmsteuerung haben. Die Anleitung zu dem Interface gibt Ihnen weitere Hinweise zur Anpassung der Programme.

Die Anleitung zu dem Interface beinhaltet auch eine Erläuterung, wie die Signale des Interface von BASIC aus verarbeitet bzw. erzeugt werden. Hier wollen wir nur kurz vermerken, daß die Steuerung eines Ausgangs durch Aufruf eines Maschinenspracheprogramms erfolgt. Als Aufrufparameter wird die Nummer des Ausgangs M1, M2, M3 oder M4 zusammen mit der Betriebsart RECHTS, LINKS, EIN oder AUS angegeben. Beispiele sind:

```
* SYS M1, RECHTS
* SYS M3, EIN
* SYS M4, AUS
```

Der Parameter EIN in der obigen zweiten Zeile ist übrigens gleichbedeutend mit dem Parameter RECHTS. Als allererstes muß jedoch immer der Befehl

```
* SYS INIT
```

erfolgen, der das Interface in einen Anfangszustand versetzt. Dabei werden alle Motoren ausgeschaltet, so daß dieser Befehl auch zum gleichzeitigen Abschalten der Motoren dient.

Die Eingänge des Interface werden mit der USR-Funktion erfaßt. Mit den Parametern E1, E2 bis E8 werden die acht Eingänge abgefragt, an die die mini-Taster angeschlossen werden. Auch andere Ein-Aus-Signale können dort eingespeist werden. Die Funktionen USR(EX) und USR(EY) hingegen dienen der Eingabe stufenlos veränderlicher elektrischer Werte. Diese Eingänge werden bei dem hier beschriebenen Roboter nicht benötigt. Sie können aber wichtig werden, wenn Sie Sensoren anschließen wollen, z.B. Fotowiderstände zur Objekterkennung.

Wichtig zu wissen ist auch, daß das Interface eine Überwachungsschaltung des Datenverkehrs besitzt. Immer wenn innerhalb einer halben Sekunde kein neuer Befehl, sei es ein Ausgabe- oder Eingabebefehl, kommt, schaltet es alle Motoren ab. Beim Stoppen des Computerprogramms brauchen Sie daher nicht eigens die Stromversorgung der Motoren abzustellen. Setzt der Datenaustausch wieder ein, nimmt das Interface alle Motoren wie zuletzt wieder in Betrieb.

Das Maschinenspracheprogramm, das den Datenaustausch zwischen Computer und Interface bewirkt, muß natürlich auch in dem Computer abgespeichert sein. Hierzu dient das sogenannte Grundprogramm, das sich ebenfalls auf der Diskette oder Kassette befindet. Gleichzeitig ist es Bestandteil eines jeden weiteren fischertechnik computing Programms und belegt die Zeilennummer 1 bis 500. In den Programmlisten dieses Anleitungsbuches erscheint dieser Teil jedoch nicht, da er für jeden Computertyp anders ausschaut. Das Maschinenprogramm muß ganz detailliert auf den Hard- und Softwareaufbau des Computers eingehen. Sie finden das Grundprogramm in der Anleitung zu Ihrem Interface dokumentiert.

Der Trainingsroboter verwendet aufgrund des oben beschriebenen Positioniersystems ein erweitertes Grundprogramm, das Roboter-Systemprogramm. Es liegt auf der Diskette bzw. Kassette als ROBOT. SYSTEM vor. Es kommen acht weitere Kommandos hinzu: Ganz ähnlich zu den Ausgabekommandos SYS M1, RECHTS oder SYS M4, AUS verhält sich das Kommando

```
* SYS P1, nnnn
```

Dieses Kommando läßt den Motor M1 solange laufen, bis der Roboter die Position nnnn erreicht hat. nnnn ist dabei eine positive Ganzzahl und gibt den Stand des oben erwähnten Impulszählers wieder. Das Roboter-Systemprogramm „weiß" also immer den jeweiligen Zählerstand. Wenn mit dem Kommando eine neue Position angefordert wird, so berechnet das Roboter-Systemprogramm die Differenz der Positionen. Aus dem absoluten Zahlenwert ergibt sich die Anzahl der abzuwartenden Impulse von der Lichtschranke, aus dem Vorzeichen die Laufrichtung des Motors. Allerdings rührt sich der Motor noch nicht. Als Programmierer haben Sie nämlich die Gelegenheit, für weitere Motoren die neue Position anzufordern. Erst wenn alle Aufträge an das Roboter-Systemprogramm ausgeteilt sind, wird das Kommando

```
* SYS ROBOT
```

aufgerufen. Nun laufen alle Motoren, die eine neue Position erreichen sollen, gleichzeitig los. Von allen in Betrieb befindlichen Motoren werden die Impulseingänge überwacht und ausgezählt. Jeder Motor wird ausgeschaltet, wenn er die neue Position erreicht hat. Sind alle Positionen erreicht, gibt das Kommando die Kontrolle wieder an das BASIC-Programm zurück.

Noch einige weitere Bemerkungen zu dem Roboter-Systemprogramm sind angebracht. Ein Motor wird auch schon vor Erreichen der Position abgeschaltet, wenn der dem Motor zugeordnete Endtaster angesprochen hat. Diese Endtaster haben einerseits die Funktion, eine „Heimposition" des Roboters festzulegen, andererseits unzulässige Bewegungen des Roboters zu vermeiden. Die Endtaster müssen im nicht betätigten Zustand geschlossen sein. Sie öffnen den Kontakt bei Betätigung. Bei den dem Modell beigefügten mini-Tastern sind somit meist die Kontaktnummern 1 und 2 anzuschließen.

Weiter hatten wir geschildert, daß bei Erreichen der Zielposition der Motor einer jeder Bewegungsachse abgeschaltet wird. Wer sich jedoch schon genauer mit der motorischen Steuerung befaßt hat, wird wissen, daß der Motor nach dem Abschalten noch etwas nachläuft. Aus diesem Grund überwacht das Roboter-Systemprogramm den Impulseingang noch für eine festgelegte Zeit nach Abschalten des Motors. Jeder dann noch eintreffende Impuls wird noch gezählt. Letztlich wird sich also nicht die Zielposition ergeben, sondern eine klein wenig danebenliegende Position. Diese tatsächlich erreichte Position kann durch die Funktion

*** USR(P1) (dto. für P2, P3 und P4)**

abgefragt werden.
Die den Motoren zugeordneten Positionszähler können auch auf Null gestellt werden. Dies erfolgt mit dem Kommando

*** SYS INIT**

das Sie schon aus dem einfacheren Grundprogramm kennen. Da es im Rahmen des Roboter-Systemprogramms noch diese Nebenwirkung hat, kann es nicht mehr bedenkenlos zum gleichzeitigen Abschalten aller Motoren verwendet werden. Allerdings wird sich diese Notwendigkeit auch gar nicht mehr ergeben, da die Motoren der zuverlässigen Kontrolle des Roboter-Systemprogramms unterliegen.
Im Gegensatz zu den einfacheren Kommandos des Grundprogramms sind in den Kommandos des Roboter-Systemprogramms Ein- und Ausgaben miteinander verwoben, um so die hohe Leistungsfähigkeit zu erzielen. Dies erfordert aber auch eine feste Zuordnung von Ein- und Ausgängen des Interface, wie die nachstehende Tabelle zeigt:

Motor	End-taster	Impuls-eingang	Kommandos	
M1	E1	E2	SYS P1, nnnn	USR(P1)
M2	E3	E4	SYS P2, nnnn	USR(P2)
M3	E5	E6	SYS P3, nnnn	USR(P3)
M4	E7	E8	SYS P4, nnnn	USR(P4)

Wie aus der Tabelle zu sehen ist, sind die Roboter-Systemkommandos für alle vier Ausgänge des Interface vorhanden, wenn auch der Trainingsroboter das Positioniersystem nur für drei Motoren benutzt. In diesem Fall werden Motor 1 bis 3 über die Positionierkommandos gesteuert. Der Motor 4 treibt die Greifzange an und wird mit den einfacheren Kommandos des Grundprogramms gesteuert.
Sollten Sie nicht mit dem fischertechnik computing Interface arbeiten, sondern mit einer anderen Interfaceschaltung, gilt das bisher Gesagte natürlich nicht in jedem Detail. Dennoch können Sie die hier skizzierten Ideen auch auf jeder anderen Hardware realisieren.

Roboter-Systemprogramm

Nachfolgend ist das BASIC-Programm wiedergegeben, das das Roboter-Systemprogramm für den Commodore 64 erzeugt. Dieses Programm sowie die ab Seite 16 noch abgedruckten Programme können auch von der fischertechnik Diskette Trainingsroboter/Plotter/Scanner geladen werden. Dies gilt auch für die entsprechenden Programme für andere Computer. Fordern Sie die Diskette unter Angabe des Typs Ihres Computers und Laufwerks bitte bei

fischerwerke Artur Fischer GmbH & Co. KG
Abt. fischertechnik
7244 Tumlingen/Waldachtal

an. Sie müssen hierzu den beigefügten Gutschein verwenden.

```
*1 PRINT CHR$(147):POKE 53280,3:POKE 53281,1
*2 FOR I=1 TO 7
*3 PRINT
*4 NEXT
*5 PRINT TAB(6);"BITTE EINEN MOMENT WARTEN"
*6 PRINT
*7 PRINT TAB(11);"TRAININGSROBOTER"
*8 PRINT
*9 PRINT TAB(5);"SYSTEMPROGRAMM WIRD GELADEN"
*10 REM ROBOTER SYSTEMPROGRAMM FUER COMMODORE 64
*13 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1985
*15 REM AUFRUF DES PROGRAMMS MIT
*20 REM SYS M1,EIN     SYS M1,AUS
*25 REM SYS M1,LINKS  SYS M1,RECHTS
*30 REM USR(E1)  USR(EX)  USR(EY)
*35 REM SPEZIELLE ROBOTERBEFEHLE
*40 REM SYS P1,NNNN  SOLLPOSITION ABLEGEN
*45 REM USR(P1)  ISTPOSITION ABFRAGEN
*50 REM SYS ROBOT  START DES ROBOTERS
*55 DATA 52656,0,0,169,0,162,23,157,208,53375
*60 DATA 207,202,16,250,48,46,169,3,54316
*65 DATA 208,10,169,12,208,6,169,48,55146
*70 DATA 208,2,169,192,120,141,177,205,56360
*75 DATA 32,253,174,173,176,205,13,177,57563
*80 DATA 205,141,176,205,32,158,183,138,58801
*85 DATA 45,177,205,141,177,205,173,176,60100
*90 DATA 205,77,177,205,32,241,205,88,61330
*95 DATA 96,141,176,205,72,169,63,141,62393
*100 DATA 3,221,162,8,169,48,14,176,63194
*105 DATA 205,144,2,9,4,141,1,221,63921
*110 DATA 9,8,141,1,221,202,208,236,64947
*115 DATA 169,57,141,1,221,104,141,176,65957
*120 DATA 205,96,120,32,247,183,201,0,67041
*125 DATA 208,118,192,162,240,57,192,146,68356
*130 DATA 240,53,140,177,205,32,61,206,69470
*135 DATA 45,177,205,168,240,2,160,1,70468
*140 DATA 32,162,179,88,96,169,50,141,71385
*145 DATA 1,221,9,8,141,1,221,162,72149
*150 DATA 8,10,44,1,221,16,2,9,72460
*155 DATA 1,160,48,140,1,221,160,56,73247
*160 DATA 140,1,221,202,208,235,96,169 ,74519
*165 DATA 255,141,4,221,141,5,221,169,75676
*170 DATA 185,141,14,221,140,1,221,160,76759
*175 DATA 58,140,1,221,173,4,221,162,77739
*180 DATA 3,202,208,253,56,237,4,221,78923
*185 DATA 208,242,162,56,142,1,221,56,80011
*190 DATA 169,255,237,4,221,168,169,255,81489
*195 DATA 237,5,221,32,145,179,88,96,82492
*200 DATA 162,0,192,194,240,18,162,2,83462
*205 DATA 192,198,240,12,162,4,192,202,84684
*210 DATA 240,6,162,6,192,206,208,11,85695
*215 DATA 188,208,207,189,209,207,32,145,87080
*220 DATA 179,88,96,140,176,205,141,177,88282
*225 DATA 205,96,169,0,240,10,169,2,89173
*230 DATA 208,6,169,4,208,2,169,6,89945
*235 DATA 141,177,205,32,253,174,32,138,91097
*240 DATA 173,32,247,183,174,177,205,157,92445
*245 DATA 217,207,152,157,216,207,96,162,93859
*250 DATA 6,56,189,208,207,253,216,207,95201
*255 DATA 141,240,207,189,209,207,253,217,96864
*260 DATA 207,141,241,207,16,5,189,201,98071
*265 DATA 207,208,11,173,240,207,13,241,99371
*270 DATA 207,240,3,189,200,207,157,224,100798
*275 DATA 207,157,232,207,202,202,16,209,102230
*280 DATA 32,61,206,141,241,207,32,61,103211
*285 DATA 206,168,77,241,207,141,240,207,104698
*290 DATA 140,241,207,169,0,141,176,205,105977
*295 DATA 162,6,173,241,207,61,201,207,107235
*300 DATA 208,6,157,224,207,157,225,207,108626
*305 DATA 173,240,207,61,200,207,240,76,110030
*310 DATA 189,232,207,221,200,207,208,20,111514
*315 DATA 56,189,208,207,233,1,157,208,112773
*320 DATA 207,189,209,207,233,0,157,209,114184
*325 DATA 207,56,176,17,24,189,208,207,115268
*330 DATA 105,1,157,208,207,189,209,207,116551
*335 DATA 105,0,157,209,207,189,208,207,117833
*340 DATA 221,216,207,208,23,189,209,207,119313
*345 DATA 221,217,207,208,15,169,0,221,120571
*350 DATA 224,207,240,8,157,224,207,169,122007
*355 DATA 255,157,225,207,169,0,221,225,123466
*360 DATA 207,240,3,222,225,207,173,176,124919
*365 DATA 205,29,224,207,141,176,205,202,126308
*370 DATA 202,16,135,173,176,205,32,241,127488
*375 DATA 205,240,3,76,30,207,162,6,128417
*380 DATA 29,225,207,202,202,16,249,201,129748
*385 DATA 0,240,3,76,30,207,88,96,130488
*390 DATA 2,1,8,4,32,16,128,64,130743
*395 DATA 1,2,4,8,16,32,64,128,130998
*400 DATA 162,146,255,170,85,85,26,206,132133
*405 DATA 52930,52934,52938,52942,52967,396844
*410 READ INIT : M1=INIT
*415 FOR M3=0 TO 67 : FOR M2=0 TO 7
*420 READ M4 : POKE INIT+M3*8+M2,M4
*425 M1=M1+M4 : NEXT
*430 READ M4 : IF M1<>M4 THEN PRINT"DATAFEHLER
     IN ZEILE";M3*5+55:PRINT M1:END
*435 NEXT
*440 READ E1,E2,E3,E4,E5,E6,E7,E8
*445 M1=M1+E1+E2+E3+E4+E5+E6+E7+E8
*450 READ M4 : IF M1<>M4 THEN PRINT"DATAFEHLER
     IN ZEILE 395":PRINT M1:END
*455 READ EX,EY,AUS,LINKS,RECHTS,EIN,M2,M3
*460 M1=M1+EX+EY+AUS+LINKS+RECHTS+EIN+M2+M3
*465 READ M4 : IF M1<>M4 THEN PRINT"DATAFEHLER
     IN ZEILE 400":PRINT M1:END
*470 POKE 785,M2 :POKE786,M3
*475 READ P1,P2,P3,P4,ROBOT
*480 M1=M1+P1+P2+P3+P4+ROBOT
*485 READ M4 : IF M1<>M4 THEN PRINT"DATAFEHLER
     IN ZEILE 405":PRINT M1:END
*490 INIT=INIT+2
*495 M1=INIT+12: M2=M1+4 : M3=M2+4 : M4=M3+4
*500 SYS INIT
```

Erste Experimente

Nach soviel Theorie zur Programmierung wollen wir nun die ersten Experimente durchführen. Wir nehmen den zuvor erstellten Versuchsaufbau zur Hand. Motor und Lichtschranke müssen mit Hilfe des beigefügten 20-adrigen Kabels mit dem Interface verbunden werden. Das Kabel sollten wir jedoch gleich so herrichten, daß wir es nachher in Form eines Kabelbaums in den Roboter einziehen können. Der Zuschnitt ist auf S. 62 gezeigt. Aus dem abgeschnittenen Teil wird einerseits ein weiterer Kabelbaum angefertigt. Zum anderen werden die nun noch verbleibenden Kabelreste in Einzeladern auseinandergezogen. Diese Kabel dienen zum weiteren Verdrahten des Modells, insbesondere um die gemeinsame Masse- und +5 V-Leitung an alle Taster und Lichtschranken zu bringen.
Insgesamt vier Leitungen werden bei dem Aufbau des Trainingsroboters nicht benutzt. Sie sind den Eingängen EX, EY und E8 zugeordnet. Wenn Sie sich jetzt schon gedanklich mit Ausbauplänen für Ihren Roboter befassen, so sollten Sie diese Leitungen nicht abschneiden, sondern in voller Länge belassen, aufwickeln und z. B. mit Klebestreifen unter dem Hauptkabelbaum fixieren. Die Kabelenden werden nun vorsichtig auf eine Länge von ca. 3 bis 5 mm abisoliert, ohne die feinen Adern der Litze zu beschädigen. Anschließend werden die Adern verdrillt. Mit dem Anschrauben der Stecker sollten Sie noch warten, bis Sie das Kabel in den Roboter eingezogen haben.

Für unseren ersten Test genügt es, wenn der Ausgang M1, der Eingang E1 und E2 sowie die Leitung +5 V mit Steckern versorgt wird. Welche Kabeladern dies sind, geht aus dem Verdrahtungsplan sowie der Deckelbeschriftung des Interface hervor. Biegen Sie die Litze auf die Isolation um. Lösen Sie das Schräubchen des Steckers und führen Sie das Kabelende in die Hülse ein. Danach wird die Schraube wieder angezogen, aber nicht so fest, daß das Kabel abgequetscht wird. Außerdem richten Sie sich aus den Kabelabschnitten zwei Kabel, die an beiden Enden mit fischertechnik-Steckern versehen werden.

Verbinden Sie nun den Motor mit den beiden zugehörigen Kabeln orange und gelb. Die Buchse mit dem ⊕-Symbol der Lichtschranke wird mit der roten +5 V-Leitung verbunden. Außerdem wird die +5 V-Leitung weitergeführt an den Kontakt 2 eines mini-Tasters. Die Buchse mit dem ⊖-Symbol der Lichtschranke wird mit der getrennten Massebuchse des Interface verbunden. Zum Schluß verbinden Sie noch die Leitung E1 (braun) mit dem Kontakt 1 des mini-Tasters und die Leitung E2 (rot) mit dem Impulsausgang der Lichtschranke ⎍⎍.

Wir kommen nun zurück auf die zuvor erwähnte Empfindlichkeitseinstellung der Lichtschranke. Wenn alle Verbindungen hergestellt sind und das Interface an den ausgeschalteten (!) Computer angeschlossen ist, schalten Sie diesen ein und laden das Programm ROBOT.JUST und starten es. Auf die Frage, welche Lichtschranke einjustiert werden soll, antworten Sie mit 1. Nun muß der Motor laufen und auf dem Bildschirm ein Meßinstrument erscheinen. Der Zeiger des Meßinstruments muß sich in dem grünen Bereich bewegen, optimalerweise auf 0,5 zeigen. Sollte das nicht der Fall sein, so können Sie wie zuvor beschrieben die Empfindlichkeit an der Lichtschranke einjustieren.

Wenn die Einstellung korrekt ist, verfahren Sie mit den anderen Lichtschranken auf die gleiche Weise. Sie können auch später die Lichtschranke im eingebauten Zustand am Roboter einstellen, müssen dazu aber das Gestänge des Roboters ausklinken. Sind diese Vorarbeiten zur Zufriedenheit abgeschlossen, so können Sie sich von der Funktion der Roboter-Systemkommandos überzeugen. Laden Sie hierzu das Programm ROBOT.SYSTEM und starten es. Ähnlich wie bei dem Grundprogramm meldet sich das Roboter-Systemprogramm zurück als wäre nichts geschehen. Doch nun können Sie die zuvor besprochenen Kommandos im Direktmodus ausprobieren. Denken Sie daran, daß die Sternchen **nicht** Bestandteil des Kommandos sind, sondern Sie an die unterschiedliche Schreibweise der Computer erinnern sollen.

Geben Sie ein:

```
* SYS P1, 64
```

und anschließend:

```
* SYS ROBOT
```

Der Motor muß für eine kurze Zeit laufen und das Rad um etwas mehr als eine Umdrehung drehen. 64 Pegelwechsel entsprechend den 32 schwarzen und 32 hellen Sektoren machen gerade eine Umdrehung aus. Wegen des Motornachlaufs wurde eine höhere Positionszahl erreicht. Ermitteln Sie diese durch das Kommando

```
* PRINT USR(P1)
```

Vielleicht werden Sie über die Größe des Nachlaufs erstaunt sein. Jedoch können Sie davon ausgehen, daß der Nachlauf später im Roboter aufgrund der Last- und Reibungsverhältnisse geringer sein wird. Im nächsten Versuch geben Sie ein:

```
* SYS P1, 10000
```

und

```
* SYS ROBOT
```

Der Motor läuft nun längere Zeit und Sie haben Gelegenheit, den angeschlossenen mini-Taster zu drücken. Sofort bleibt der Motor stehen und das Kommando ist beendet. Mit

```
* PRINT USR(P1)
```

können Sie sehen, wie weit der Positionszähler kam, bevor Sie die Taste drückten.

Aufbau des Roboters

Nun wollen wir den Motor wieder zurücklaufen lassen:

*** SYS P1,0 : SYS ROBOT**

Der Motor wird sich nun in der anderen Richtung drehen und der Zähler über die Nullposition in den negativen Zahlenbereich überschwingen. Sie können sich davon überzeugen:

*** PRINT USR(P1)**

Zum Abschluß noch ein Hinweis zur genaueren Positionierung. Hierzu müssen Sie die Getriebewelle bremsen. Sie können eine Bremse aufbauen oder aber einfach von Hand bremsen. Geben Sie nun ein:

*** SYS P1, 64 : SYS ROBOT : SYS ROBOT : SYS ROBOT**

Da das Roboter-Kommando nun dreimal mit dem gleichen Sollwert aufgerufen wird, wird die Position noch zweimal nachkorrigiert. Aufgrund der Bremse kommt aber der Motor nicht mehr auf volle Drehzahl und stoppt jedesmal genauer. In dieser Richtung können Sie noch weitere Experimente unternehmen, wenn der Roboter aufgebaut ist und die Motoren unter der Last des Roboterarms stehen.

Zum Aufbau des Roboters verfahren Sie nach der anschließend folgenden bebilderten Bauanleitung. Wenn der Roboter aufgebaut ist, prüfen Sie noch einmal, ob alle Teile exakt ausgerichtet sind und die Antriebsgestänge leichtgängig arbeiten. Hierzu können Sie die Motoren ein klein wenig aus dem Eingriff in die Getriebeverzahnung ausklinken.
Danach beginnt die Verkabelung des Roboters. Verfahren Sie nach dem Verdrahtungsplan auf Seite 91.

Beginnen Sie damit, daß Sie den Hauptkabelbaum in der Zugentlastung arretieren. Alle Motorausgänge werden zunächst mit der Lampenreihe verbunden. Danach werden die mit * gekennzeichneten Leitungen, insbesondere der kleine Kabelbaum, durch die zentrale Öffnung des Drehkranzes und entlang dem Aufbau des Roboters geführt. Eine Pinzette ist zum Durchziehen der Kabel hilfreich. Mit Hilfe der grauen Bauplatten können Sie die Kabel in den Nuten der Metallbaustäbe arretieren.
Danach kommt der elektrische Test des Trainingsroboters. Laden Sie hierzu das Diagnoseprogramm DIAGNOSE. Überprüfen Sie die Endtaster, Eingänge E1, E3 und E5. Im nicht betätigten Zustand sollte eine Eins am Bildschirm erscheinen. Bei dem Taster zur Überwachung des Greifers ist es umgekehrt, hier erscheint die Eins, wenn der Taster gedrückt ist. Allen Endtastern ist aber gemeinsam, daß die Eins in dem zulässigen Arbeitsbereich des Roboters erscheint. Warum wurde gerade diese Polarität? Wenn während des Betriebs des Roboters ein Kabel reißt, so ergibt sich die gleiche Situation, wie wenn der Endtaster öffnet. Das Roboter-Systemprogramm reagiert auf Kabelriß daher auch mit Abschalten des Motors.
In die Plusleitung, die zu allen Endtastern führt, ist noch einmal ein Taster eingeführt. Wird dieser Kontakt geöffnet, so gehen sämtliche Eingänge auf Null und das Robotersystemprogramm schaltet alle Motoren ab. Der Taster hat also eine Not-Aus-Funktion. Auch ihn sollten Sie überprüfen.

Von der Funktionsfähigkeit der Gabellichtschranken können Sie sich nochmals durch leichtes Drehen des bedruckten Rades überzeugen. Die zugeordnete Anzeige muß zwischen Eins und Null hin- und herspringen.
Nun zu den Motoren. Für erste Versuche ist es vielleicht besser, wenn das Getriebe des Roboters noch ausgeklinkt ist. Mit den Zahlentasten wird ein Motor angewählt. Nun kann für diesen Motor mit den Tasten R und L Rechts- bzw. Linkslauf gewählt werden. Mit der Taste A wird der Motor abgeschaltet. Auf diese Weise können Sie den Roboter bewegen. Überzeugen Sie sich, daß bei Rechtslauf die Spindelmutter des Oberarm- und Unterarmantriebs nach unten wandert. Ist dies nicht der Fall, müssen die Anschlüsse am Motor umgepolt werden. Desgleichen muß die Drehbewegung des Roboters von oben gesehen im Uhrzeigersinn erfolgen, wenn Rechtslauf gewählt wird. Die Greifzange muß schließlich bei Rechtslauf öffnen.
Sind alle Kontrollen abgeschlossen können wir zur Robotersteuerung übergehen.

Steuerung des Roboters

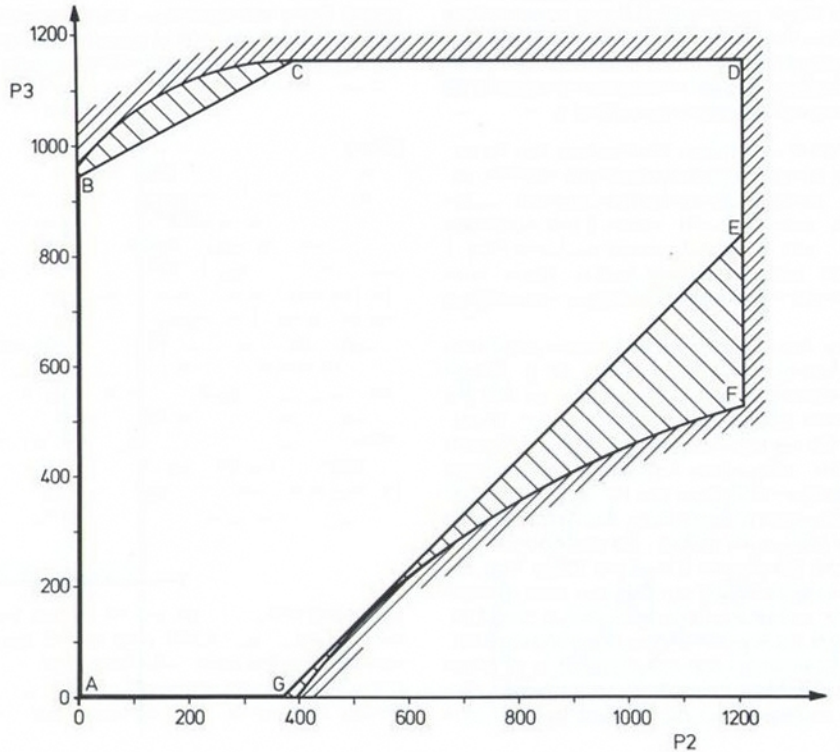
Um den Roboter programmieren zu können, sollten wir uns zunächst mit den Bewegungsformen des Roboters vertraut machen. Besser als das Diagnoseprogramm ist hierzu das Programm ROBOT.HAND geeignet. In diesem Programm steht der Roboter unter der Kontrolle des Roboter-Systemprogramms, so daß z.B. die Endtaster und die Not-Aus-Schaltung automatisch überwacht werden. Das Programm zeigt Ihnen am Bildschirm die Bedienung des Roboters an und steuert den Roboter in seine Heimposition. Die Heimposition des Roboters wird durch das Ansprechen der Endtaster gekennzeichnet. In dieser Position ist der Arm des Roboters erhoben, die Zange geöffnet und der Roboter zeigt auf die dem Anschlußkabel gegenüberliegende Seite. Allerdings sind in der eigentlichen Heimposition die Endtaster gerade eben wieder freigegeben, da im Arbeitsbereich des Roboter-Systemprogramms die Endtaster nicht ansprechen dürfen. Nach Erreichen der Heimposition wird der Befehl

* **SYS INIT**

durchlaufen, so daß die Heimposition den Positionswert Null für alle drei Bewegungsachsen erhält.

Die Bewegungen des Roboters werden mit den Zahlentasten angesteuert. Die Wegstrecke pro Tastendruck kann mit Hilfe der Funktionstasten gewählt werden, so daß Sie auf einfache Weise grob und fein positionieren können. Der Greifzangenmotor wird beim Schließen im Gegensatz zu den Roboterachsen immer über feste Zeitintervalle angesteuert. Die Dauer des Zeitintervalls können Sie per Kommando ändern. Wählen Sie sie so, daß der Gegenstand sicher ergriffen wird, jedoch das Zangengetriebe noch nicht durch Verkanten blockiert. Beim Öffnen der Greifzange wartet das Programm auf ein 0-Signal an E7 (Taster nicht betätigt). Die Betätigung der HOME-Taste bringt den Roboter immer wieder in seine Heimposition. Aus der Heimposition kann der Roboter nur in Richtung positiver Positionsdaten bewegt werden. Dies ist bei den beiden Armachsen einleuchtend, denn ein weiteres Zurückfahren ist nicht möglich. Anders bei der Drehbewegung. Die Heimposition kann freizügig auf jeden Drehwinkel des Roboterkörpers relativ zur Grundplatte festgelegt werden, da der Roboter in seiner Drehbewegung nicht eingeschränkt ist. In den Baustufen ist sie so angelegt, daß der Roboter von den Anzeigelampen wegzeigt. Durch Versetzen des Betätigungsnockens sind aber auch andere Positionen möglich. Um ein Verdrehen des Kabelbaums zu verhindern, gilt jedoch die gleiche Einschränkung wie bei den Armachsen auch: es können nur positive Positionsdaten angefahren werden. Wenn Sie den Roboter wie gezeigt aufbauen, empfiehlt sich daher, den Arbeitsbereich hauptsächlich vor der Breitseite der Grundplatte anzuordnen, um dann freizügig nach rechts und links schwenken zu können.

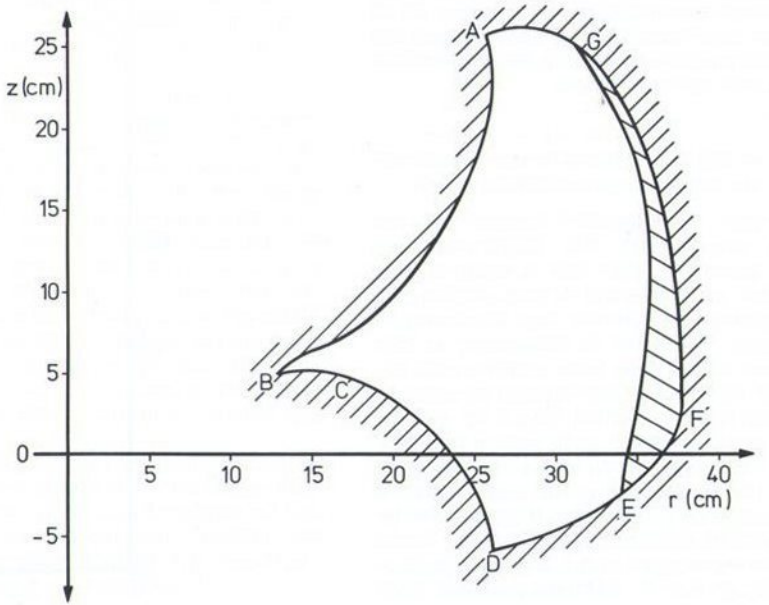
Bild 2



Experimentieren Sie mit dem Roboter und versuchen Sie, die beigefügten Werkstücke zu greifen und umzusetzen. Sie werden feststellen, daß Sie hierzu sorgfältig und mit Bedacht vorgehen müssen. Sie werden auch die Wirkung der beiden Bewegungsachsen von Ober- und Unterarm einzuschätzen lernen. Der Oberarm bewirkt hauptsächlich das Vorstrecken und das Zurückziehen der Greifhand, während der Unterarm mehr für eine Auf-/Ab-Bewegung zuständig ist.
Sie werden weiter feststellen, daß die beiden Bewegungsachsen nicht ganz unabhängig voneinander bedient werden können. In bestimmten Zuständen stößt das Gestänge an der einen oder anderen Stelle an. Diesen Gesichtspunkt wollen wir etwas näher erforschen und verwenden hierzu Bild 2.

Da das Programm uns immer die Position der Bewegungsachsen auf dem Bildschirm angibt, können wir untersuchen, welche Zahlenkombinationen zulässig sind und welche nicht. Jede Zahlenposition ergibt einen Punkt in dem Achsenkreuz von Bild 2. Alle erlaubten Kombinationen bilden dann eine Fläche, die wir als internen Arbeitsraum bezeichnen werden.
Zur Ermittlung des internen Arbeitsraums beginnen wir bei der Heimposition, Koordinate (0,0). Weiter zurück kann keine der Antriebsspindeln, so daß die negativen Werte beider Bewegungsachsen entfallen. Erhöhen Sie nun die Position der Achse 2 Schritt für Schritt, ohne die Achse 3 zu verändern. In dem Diagramm wandern Sie längs der P2-Achse. Irgendwann wird eine weitere Verstellung der Achse 2 nicht mehr möglich sein, ohne daß ein Teil des Antriebsgestänges aneckt. Die Achse 3 muß ein Stück von der Nullposition weggesteuert werden, um eine weitere Bewegung der Achse 2 zu ermöglichen. In dem Diagramm löst sich die Randlinie des Arbeitsraums von der Koordinatenachse. Mit viel Sorgfalt und unter ständiger Beobachtung des Roboters können Sie die Randlinie verfolgen, bis die Spindel der Achse 2 ganz ausgefahren ist. Nun wird die Bewegungsachse 3 verstellt, bis auch diese ganz ausgefahren ist. In dem Diagramm verläuft die rechte Randbegrenzung senkrecht nach oben. An dem oberen Rand entlang geht es nun durch Verringerung der Positionswerte der Bewegungsachse 2 wieder nach links. Aber auch hier kann Bewegungsachse 2 nicht ganz bis Null zurückgenommen werden, ohne Bewegungsachse 3 ebenfalls zurückzunehmen. Am linken Rand des internen Arbeitsraums angekommen, kann durch Zurückfahren der Bewegungsachse 3 die Heimposition wieder erreicht werden.
Nachdem wir nun ein Diagramm besitzen, das uns das Gebiet der zulässigen Werte angibt, interessiert uns auch, welche Bewegungen des Roboters diesem Gebiet zuzuordnen sind. Die Position des Roboters beschreibt man am leichtesten durch die Position eines idealisierten Punktes, dem „tool-center-point", TCP. Dieser Punkt ist die Mitte zwischen den Backen der Greifzange. Wird die Lage dieses Punktes im Raum angegeben, so sind die Positionen aller drei Bewegungsachsen daraus ableitbar. Der Zusammenhang zwischen der Höhe des TCP über der Grundplatte, z, und der Entfernung vom Drehmittelpunkt, r, einerseits und den Achspositionen P2 und P3 andererseits ergibt sich aus den nachfolgenden Formeln, die wir hier nicht weiter ableiten wollen:

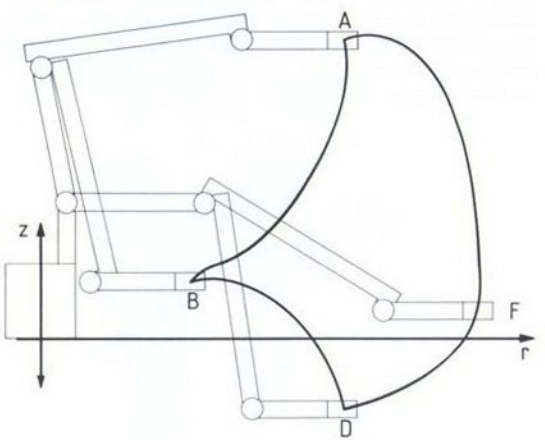
Bild 3



$r = 120*\cos\alpha + 180*\cos\beta + 110$
$z = 120*\sin\alpha + 180*\sin\beta + 117{,}5$
$\alpha = 126° - \delta$
$\beta = 36° - \varepsilon$
$\delta = \cos^{-1}\,[(14004-(P2*0{,}07363+60)^2)/12240]$
$\varepsilon = \cos^{-1}\,[(14004-(P3*0{,}07363+60)^2)/12240]$
(alle Maße in mm)

Bei der Anwendung dieser Gleichungen müssen Sie berücksichtigen, daß diese den idealisierten Zusammenhang wiedergeben. In der Praxis wird sich eine Abweichung durch das notwendige Lagerspiel ergeben. Auch aus dem Vergleich von Bild 2 und Bild 3 können Sie den Zusammenhang zwischen internem und realem Arbeitsraum ablesen. Die charakteristischen Eckpunkte sind in beiden Bildern mit den gleichen Buchstaben bezeichnet. Den realen Arbeitsraum können Sie auch selbst ermitteln. Sie verfolgen gemäß Bild 2 die Randlinie des internen Arbeitsraums und messen die Position des TCP.

Bild 4



Damit Sie eine Vorstellung von der Stellung des Roboters gewinnen, ist in Bild 4 der reale Arbeitsraum zusammen mit schematischen Skizzen des Roboters aufgetragen.
Wir wollen nun das Programm ROBOT.HAND so abändern, daß wir den Roboter nicht versehentlich beschädigen können. Die Randlinien des internen Arbeitsraums müssen abgefragt werden und jede Steuerung außerhalb dessen vermieden werden. Dazu wird die Randlinie durch Geraden angenähert. Während dies parallel zu den Koordinatenachsen und in der linken oberen Ecke des internen Arbeitsraums problemlos möglich ist, verdient der rechte untere Eckabschnitt besondere Beachtung. Wir müssen bei der Definition auch die Arbeitsweise des Positioniersystems beachten.

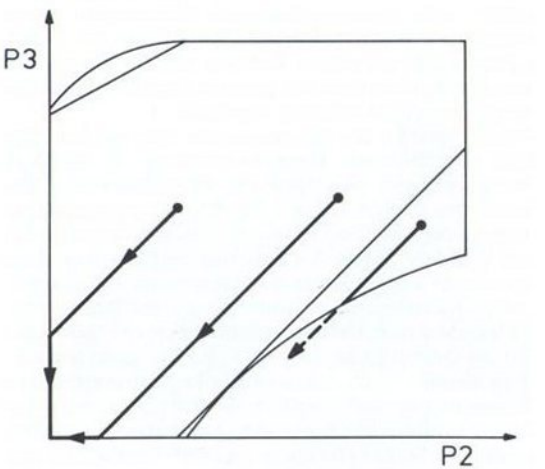
Wenn das Roboterprogramm gestartet wird, kann das Programm keinerlei Kenntnisse über die vorliegende Position des Roboters haben (in Fachsprache: er besitzt kein absolutes Positioniersystem). Alle Positionskenntnisse werden ja nur über das Zählen von Impulsen, beginnend mit einer bekannten Position, gewonnen (inkrementales Positioniersystem). Diese unabhängig bekannte Position ist die Heimposition, wie sie sich aus dem Ansprechen der Endtaster ergibt. Daher steuert das Programm den Roboter immer zu Beginn an die Heimposition. Die Vorgehensweise ist dabei folgende: Alle Motoren werden gestartet und laufen jeweils solange, bis der zugeordnete Endtaster anspricht. Die Programmierung erfolgt mit den einfacheren Kommandos, ohne Überwachung des Impulseingangs, z.B.:

```
* SYS M1, RECHTS
```

Anschließend laufen sie wieder, wie zuvor beschrieben, vom Endschalter weg. Diese Bahn, im internen Arbeitsraum aufgezeichnet, bildet eine Diagonallinie, da bis auf kleine Exemplarstreuungen beide Motoren der Armantriebe gleich schnell laufen.

Sobald die Bahn auf eine Koordinatenachse stößt, verläuft sie längs dieser bis in den Nullpunkt (Bild 5). Nehmen wir nun an, der Roboter sei nahe dem Punkt F abgestellt worden. Bei dem Anfahren der Heimposition würde die Bahn das erlaubte Gebiet verlassen. Daher muß der Eckabschnitt bei Punkt F so gewählt werden, daß er von einer Geraden mit 45° Neigung gebildet wird. Diese Begrenzungen des Arbeitsraums sind in den Bildern 2 und 3 als Verbindungslinie EG eingezeichnet.
Die Berücksichtigung dieser Begrenzungen erfordert nur wenige zusätzliche Programmzeilen. Wenn Sie nun das Programm ROBOT.RAUM laden, so haben Sie die gleichen Steuerungsmöglichkeiten des Roboters wie zuvor, jedoch wird er sich weigern, seinen Arbeitsraum zu überschreiten.

Bild 5



Teach-in Verfahren

Die Benutzung der bisherigen Programme war zwar sehr hilfreich zum Verständnis der Geometrie des Roboters, stellte jedoch keine Programmierung des Roboters im eigentlichen Sinne dar. Für jede Bewegung des Roboters war der Eingriff des Bedieners notwendig. Ein Kennzeichen der Roboter-Programmierung ist jedoch, daß der Roboter die aufgetragene Tätigkeit selbständig durchführt.

In dem vorangegangenen Abschnitt hatten wir auch die Gleichungen angegeben, die den Zusammenhang zwischen den Positionen der Bewegungsachsen im internen Arbeitsraum und der Position des TCP im realen Arbeitsraum vermitteln. Mit dieser Kenntnis wäre es möglich, eine bestimmte Bewegungsabfolge des Roboters festzulegen und in einer Tabelle zu codieren. Danach kann das Roboterprogramm der Reihe nach die Positionsdaten aus der Tabelle entnehmen und den Roboter entsprechend steuern. Diese Methode wird auch in der Praxis eingesetzt, insbesondere wenn die Positionsdaten nicht nur aus einer Tabelle ermittelt, sondern vielleicht aus vorangegangenen Messungen noch modifiziert werden. Denken Sie z.B. an ein Sichtsystem, das mit einem Roboter verbunden ist und ihm Kenntnisse über die genaue Lage und Orientierung eines Werkstückes vermittelt.
Praktischer für die überwiegende Zahl der Einsatzfälle ist jedoch das Teach-In Verfahren. Im Teach-In Verfahren wird wie zuvor die Handsteuerung des Roboters eingesetzt. Sie dient darüber hinaus zur Erzeugung der obengenannten Tabelle. Immer wenn ein wichtiger Punkt der Bahn des Roboters erreicht ist wird er auf ein besonderes Kommando abgespeichert. Nach und nach entsteht so die Tabelle entsprechend dem Geschick des Bedieners. Wenn die Tabelle komplett erstellt ist, kann sie dann als Vorlage dienen, nach der der Roboter selbständig den Bewegungsablauf wiederholt. Eine Kenntnis der Umrechnungsgleichungen ist nicht erforderlich und auch die Fehlerhäufigkeit bei der Festlegung der Bahn wird geringer sein. Der Roboterinstruktor hat ja den Roboter ständig vor Augen.
Ein solches Programm liegt unter dem Namen ROBOT.TEACH vor. Über das Beschriebene hinaus sind noch weitere Komfortstufen zur Pflege der Bewegungstabellen eingebaut. Die Tabelle kann z.B. auf Diskette oder Kassette abgespeichert und von dort auch wieder geladen werden. Sie kann über einen angeschlossenen Drucker ausgedruckt werden. Tabellenpunkte können gelöscht und mehrere Tabellen im Speicher des Computers vereinigt werden.
Wenn Sie das Programm laden und starten, wird Ihnen zunächst ein Hauptmenü auf dem Bildschirm angezeigt, mit dessen Hilfe Sie die gewünschte Funktion wählen können. Beim ersten Start können Sie auf eine beispielhafte Bewegung zurückgreifen, die schon auf Diskette bzw. Kassette abgespeichert ist. Wählen Sie daher zunächst den Menüpunkt „L" zum Laden eines Files an. Sie müssen anschließend den Filenamen angeben; er lautet BEISPIEL 1. Nach dem Laden des Files erscheint wieder das Menü. Sie können als nächstes die Bewegungstabelle ausdrucken, sofern Sie einen Drucker an Ihren Computer angeschlossen haben. Dies erfolgt durch Anwahl des Menüpunktes „P".

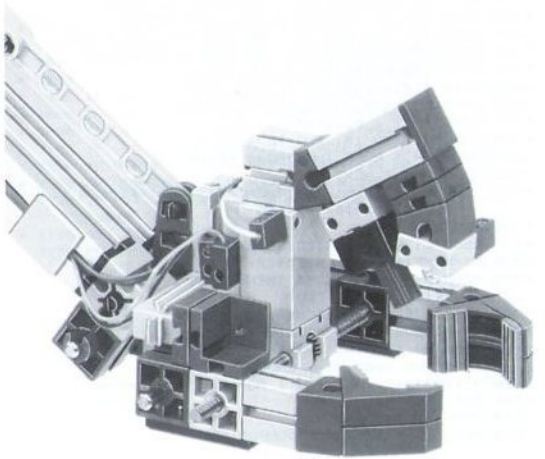
Jetzt wollen wir auch den Roboter in Betrieb nehmen. Mit dem Menüpunkt „R" wird die Ausführung der Bewegung angewählt. Nach Angabe der Anzahl der Durchläufe durch die Bewegungstabelle beginnt der Roboter die codierte Bewegung auszuführen. Wollen Sie nicht das Ende der Bewegung abwarten, so können Sie auch vorzeitig durch Drükken der Taste „M" wieder in das Hauptmenü zurückgelangen.
Doch nun zum Teach-In Verfahren. Wählen Sie Menüpunkt „T" an. Das Programm frägt Sie, ob die bisherige Bewegungsfolge gelöscht werden soll. Sie haben nun die Wahl, den eben geladenen Bewegungsablauf zu verlängern oder ganz von neuem anzufangen. Nach Beantwortung der Frage erscheint ein neues Menü, das Ihnen in groben Zügen schon von der Handsteuerung bekannt ist. Wie schon gewohnt werden die Motoren über die Zahlentasten, die Schrittweite über die Funktionstasten gesteuert. Auch das Anfahren der Heimposition ist vorgesehen. Neu kommt hinzu, daß mit jedem Betätigen der RETURN-Taste die gerade vorliegende Roboterposition an das bisherige Tabellenende angefügt wird. Mit der Löschtaste DEL wird jedoch der letzte Tabellenpunkt wieder ausgetragen, so daß Sie fehlerhafte Bahnen auch wieder korrigieren können.
Haben Sie einen Bewegungsablauf nach Ihren Vorstellungen eingegeben, so gelangen Sie durch Drücken der Taste „M" wieder in das Hauptmenü.
So wie das Menü angelegt ist, können Sie aber auch immer wieder zwischendurch einen Probedurchlauf anfordern und anschließend den Tabellenaufbau fortsetzen. Oder aber zwischendurch den Bewegungsablauf auf Diskette sichern oder ausdrucken.
Das Programm ROBOT.TEACH wird Ihr Standardwerkzeug in der Roboterprogrammierung werden. Gleichzeitig läßt es sich aufgrund seiner ausführlichen Dokumentation leicht an Ihre speziellen Anforderungen anpassen.

Weitere Experimente

Zum Schluß noch einige Hinweise, wie Sie den Trainingsroboter auch noch einsetzen können. Hierzu sollen Ihnen die folgenden Abbildungen einige Anregungen vermitteln. So kann die Greifzange auch nach unten zeigend montiert werden. Diese Greiferhaltung ist z.B. zum Greifen von Schachfiguren geeignet. Sicherlich eine besondere Herausforderung für einen Programmierer, einen Schachroboter zu bauen.

Im nächsten Bild ist ein Elektromagnet angebaut. Auf diese Weise lassen sich Eisenteile recht einfach greifen.

Zum Schluß der Anbau einer Reflexlichtschranke an den Greifarm. Deutlich ist die Lampe zu erkennen, die den Raum zwischen den Greiferbacken ausleuchtet. Sie wird direkt an das Netzgerät angeschlossen. Der benachbarte Fotowiderstand ist durch eine Kappe und einen Tubus so geschützt, daß er nicht das direkte Licht der Lampe registrieren kann. Er reagiert jedoch mit einer deutlichen Widerstandsänderung, wenn ein heller Gegenstand in den Greifbereich kommt. Diese Widerstandsänderung kann durch den Interfaceeingang EX oder EY registriert und dem Computer mitgeteilt werden. Durch geeignete Suchprogramme kann auf diese Weise ein Objekt erkannt werden und der Greifarm genau über diesem ausgerichtet werden. Danach muß der Arm nur noch abgesenkt werden, um das Objekt zu greifen. Auch diese Aufgabe ist eine rechte Herausforderung an Ihre Programmierkunst.



Prog. ROBOT.JUST

```
*1 PRINT CHR$(147)
*2 PRINT CHR$(31)
*3 POKE53280,0
*4 POKE53281,0
*5 PRINT"GRUNDPROGRAMM WIRD GELADEN"
*10 REM MODIFIZIERTES INTERFACE PROGRAMM ZUR
*15 REM LICHTSCHRANKENJUSTAGE (COMMODORE 64)
*20 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1984
*30 REM AUFRUF DES PROGRAMMS MIT
*40 REM SYS M1,EIN      SYS M1,AUS
*50 REM SYS M1,RECHTS   SYS M1,LINKS
*60 REM USR(E1)    USR(EX)    USR(EY)
*70 REM M1 BIS M4 SIND MOTORANSTEUERUNGEN
*80 REM E1 BIS E8 SIND DIGITALEINGAENGE
*90 REM EX UND EY SIND ANALOGEINGAENGE
*100 DATA 52736,169,0,240,38,169,3,208,10,53573
*110 DATA 169,12,208,6,169,48,208,2,54395
*120 DATA 169,192,120,133,255,32,253,174,55723
*130 DATA 165,254,5,255,133,254,32,158,56979
*140 DATA 183,138,37,255,133,255,165,254,58399
*150 DATA 69,255,133,254,168,169,63,141,59651
*160 DATA 3,221,162,8,169,48,6,254,60522
*170 DATA 144,2,9,4,141,1,221,9,61053
*180 DATA 8,141,1,221,202,208,237,169,62240
*190 DATA 57,141,1,221,132,254,88,96,63230
*200 DATA 120,32,161,183,134,255,169,0,64284
*210 DATA 141,156,206,169,255,141,155,206,65713
*220 DATA 169,50,141,1,221,9,8,141,66453
*230 DATA 1,221,162,8,10,44,1,221,67121
*240 DATA 16,2,9,1,160,48,140,1,67498
*250 DATA 221,160,56,140,1,221,202,208,68707
*260 DATA 235,37,255,240,2,169,1,24,69670
*270 DATA 109,156,206,141,156,206,206,155,71005
*280 DATA 206,208,205,172,156,206,32,162,72352
*290 DATA 179,88,96,0,0,0,0,0,72715
*350 DATA 1,2,4,8,16,32,64,128,72970
*360 DATA 255,170,85,85,80,206,73851
*370 READ INIT : M1=INIT
*380 FOR M3=0 TO 19: FOR M2=0 TO 7
*390 READ M4 : POKE INIT+M3*8+M2,M4
*400 M1=M1+M4 : NEXT
*410 READ M4 : IF M1<>M4 THEN PRINT"DATAFEHLER
     IN ZEILE";M3*10+100:END
*420 NEXT
*430 READ E1,E2,E3,E4,E5,E6,E7,E8
*440 M1=M1+E1+E2+E3+E4+E5+E6+E7+E8
*450 READ M4 : IF M1<>M4 THEN PRINT"DATAFEHLER
     IN ZEILE 350" : END
*460 READ AUS,LINKS,RECHTS,EIN,M2,M3
*470 M1=M1+AUS+LINKS+RECHTS+EIN+M2+M3
*480 READ M4 : IF M1<>M4 THEN PRINT"DATAFEHLER
     IN ZEILE 360",M1 : END
*490 M1=INIT+4 : M2=M1+4 : M3=M2+4 : M4=M3+4
*500 SYS INIT
 510 REM
 520 REM FISCHERTECHNIK COMPUTING
 530 REM
 540 REM LICHTSCHRANKENJUSTAGE
 550 REM
 560 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1985
 570 REM
 580 REM BELEGUNG DES INTERFACE
 590 REM MOTOR   IMPULS    KOMMANDO
 600 REM         EINGANG
*610 REM M1        E2       USR(E2)
*620 REM M2        E4       USR(E4)
*630 REM M3        E6       USR(E6)
*640 REM M4        E8       USR(E8)
 650 REM
 660 REM FUNKTION
 670 REM DAS PROGRAMM DIENT ZUR PUSBREITENJUSTAGE
 680 REM DER GABELLICHTSCHRANKE.
 690 REM UM OPTIMALE AUSWERTBEDINGUNGEN ZU ERHALTEN
 700 REM SOLLTE DAS TASTVERHAELTNIS 1:1 SEIN.
 710 REM DIES WIRD EINGESTELLT INDEM MAN MIT EINEM
 720 REM SCHRAUBENDREHER VORSICHTIG AM INNENLIEGENDEN
 730 REM POTENTIOMETER DREHT. DER PFEIL DER ANZEIGE
 740 REM SOLLTE SICH IM GRUENEN FELD BEWEGEN.
 750 REM
*1000 PRINT CHR$(147)
 1010 PRINT"WELCHE LICHTSCHRANKE SOLL"
 1020 INPUT"EINJUSTIERT WERDEN ";L
 1030 IF L=1THEN LET E=E2:M=M1
 1040 IF L=2THEN LET E=E4:M=M2
 1050 IF L=3THEN LET E=E6:M=M3
 1060 IF L=4THEN LET E=E8:M=M4
 1070 IF L>4 OR L<1 THEN GOTO 1020
*1080 PRINT CHR$(147)
*1090 PRINT CHR$(28)"      F I S C H E R"CHR$(31)
     " T E C H N I K"
 1100 PRINT
*1110 PRINT CHR$(158)"           C O M P U T I N G"
     CHR$(31)
 1120 PRINT
 1130 PRINT" PULSBREITENJUSTAGE GABELLICHTSCHRANKE"
 1140 PRINT:PRINT
 1150 PRINT"0       0.25       0.5       0.75      1"
*1160 DATA 180,160,160,160,160,103,160,160
*1170 DATA 160,160,104,160,160,160,160,103
*1180 DATA 160,160,160,160,103,160,160,160
*1190 DATA 160,160,103,160,160,160,160,180
*1200 DATA 160,160,160,103,160,160,160,170
 1210 FOR C=0 TO 39
 1220 READ D
 1230 PRINT CHR$(D);
 1240 NEXT C
*1250 DATA 28,111,183,183,183,183,183,183
*1260 DATA 183,183,183,183,183,183,183,30
*1270 DATA 183,183,183,183,183,183,183,183
*1280 DATA 183,183,183,183,183,183,28,183
*1290 DATA 183,183,183,183,183,183,183,183
*1300 DATA 183,183,112
 1310 FOR C=0 TO 42
 1320 READ D
 1330 PRINT CHR$(D);
 1340 NEXT C
 1350 PRINT
*1360 DATA 28,108,175,175,175,175,175,175
*1370 DATA 175,175,175,175,175,175,175,30
*1380 DATA 175,175,175,175,175,175,175,175
*1390 DATA 175,175,175,175,175,175,28,175
*1400 DATA 175,175,175,175,175,175,175,175
*1410 DATA 175,175,186
 1420 FOR C=0 TO 42
 1430 READ D
 1440 PRINT CHR$(D);
 1450 NEXT C
*2000 SYS M,EIN
 2010 PRINT:PRINT
*2020 PRINT CHR$(31)"BITTE ANZEIGEPFEIL IN GRUENEN"
 2030 PRINT"BEREICH BRINGEN"
*2040 PRINT"DABEI"CHR$(18);CHR$(28)" VORSICHTIG";
*2050 PRINT CHR$(31);CHR$(146)"MIT DEM SCHRAUBEN-"
 2060 PRINT"DREHER NACH LINKS ODER RECHTS DREHEN!"
 2070 PRINT:PRINT
 2080 PRINT"ENDE DER MESSUNG MIT "CHR$(18)"F1"
      CHR$(146)
 2090 FOR I=1 TO 11
*2100 PRINT CHR$(145);
 2110 NEXTI
*2120 PRINT SPC(38/255*USR(E));CHR$(5)"†"CHR$(31);
*2130 PRINT CHR$(145)
 2140 PRINT"
*2150 PRINT CHR$(145);
*2160 GET A$:IF A$=CHR$(133)THEN RESTORE:GOTO 10
 2170 GOTO 2120
```

Prog. ROBOT.HAND

```
* 500 SYS INIT
  510 REM
  520 REM FISCHERTECHNIK COMPUTING
  530 REM
  540 REM HANDSTEUERUNG DES TRAININGSROBOTERS
  550 REM MIT POSITIONSERKENNUNG
  560 REM
  570 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1985
  580 REM
  590 REM FUNKTION
  600 REM DER ROBOTER WIRD UEBER DIE TASTATUR VON HA
      ND GESTEUERT.
  610 REM GLEICHZEITIG WIRD UEBER DIE SEKTORSCHEIBE
  620 REM DIE POSITION DES ROBOTERS EINGELESEN.
  630 REM ZUR STEUERUNG DIENEN DABEI DIE TASTEN 1-8.
* 640 REM DIE HOMETASTE DIENT ZUM ANFAHREN DER HEIMP
      OSITION.
* 650 REM DIE TASTEN F1-F7 LEGEN DEN WEG PRO TASTEND
      RUCK FEST.
  660 REM TASTENBELEGUNG
  670 REM 1 UND 2 = M1 RECHTS UND LINKS
  680 REM 3 UND 4 = M2 RECHTS UND LINKS
  690 REM 5 UND 6 = M3 RECHTS UND LINKS
  700 REM 7 UND 8 = M4 RECHTS UND LINKS
* 710 REM F1=256 SEKTOREN WEG PRO TASTENDRUCK
* 720 REM F3=64 SEKTOREN WEG PRO TASTENDRUCK
* 730 REM F5=16 SEKTOREN WEG PRO TASTENDRUCK
* 740 REM F5=4 SEKTOREN WEG PRO TASTENDRUCK
* 750 REM HOME=HEIMPOSITION ANFAHREN
* 760 PRINT CHR$(147)
  770 PRINT:PRINT
* 780 PRINT CHR$(28)"      F I S C H E R"CHR$(31)" T
      E C H N I K"CHR$(144)
  790 PRINT
  800 PRINT"            C O M P U T I N G"
  810 PRINT:PRINT
  820 PRINT   "         D R E I A C H S I G E R"
  830 PRINT
  840 PRINT"    T R A I N I N G S R O B O T E R"
  850 PRINT:PRINT
  860 PRINT" STEUERUNG UEBER DIE TASTATUR MIT"
  870 PRINT" AUFZEICHNEN DER ROBOTERPOSITION "
  890 PRINT
  900 PRINT" ROBOTER FAEHRT IN HEIMPOSITION"
  910 GOSUB 3000
  920 LET ZZ=200
* 1000 PRINT CHR$(147)
  1010 PRINT"TASTENFUNKTIONEN:              POSITION:"
  1020 PRINT:
* 1030 PRINT CHR$(18)"1"CHR$(146)"  ROBOTER NACH LIN
      KS"
* 1040 PRINT CHR$(18)"2"CHR$(146)"  ROBOTER NACH REC
      HTS"
  1050 PRINT
* 1060 PRINT CHR$(18)"3"CHR$(146)"  OBERARM VOR"
* 1070 PRINT CHR$(18)"4"CHR$(146)"  OBERARM ZURUECK"
  1080 PRINT
* 1090 PRINT CHR$(18)"5"CHR$(146)"  UNTERARM AB"
* 1100 PRINT CHR$(18)"6"CHR$(146)"  UNTERARM AUF"
  1110 PRINT
* 1120 PRINT CHR$(18)"7"CHR$(146)"  ZANGE AUF"
* 1130 PRINT CHR$(18)"8"CHR$(146)"  ZANGE ZU"
  1140 PRINT
  1150 PRINT "SCHRITTWEITEN"
* 1155 PRINT"ZEITKONSTANTE GREIFZANGE  "CHR$(18)"+/-
      "CHR$(146)
  1160 PRINT
* 1170 PRINT CHR$(18)"F1"CHR$(146)"  256 SEGMENTE"
* 1180 PRINT CHR$(18)"F3"CHR$(146)"   64 SEGMENTE"
* 1190 PRINT CHR$(18)"F5"CHR$(146)"   16 SEGMENTE"
* 1200 PRINT CHR$(18)"F7"CHR$(146)"    4 SEGMENTE"
  1210 PRINT
* 1220 PRINT CHR$(18)"HOME"CHR$(146)"  HEIMPOSITION
      ANFAHREN"
  2000 LET O%=16
* 2010 LET CL$="      "+CHR$(157)+CHR$(157)+CHR$(157)
      +CHR$(157)+CHR$(157)
* 2020 LET O1%=USR(P1):IF O1%<0 THEN O1%=0
* 2030 LET O2%=USR(P2):IF O2%<0 THEN O2%=0
* 2040 LET O3%=USR(P3):IF O3%<0 THEN O3%=0
  2050 REM STEUERUNG UEBER TASTATUR
* 2060 LET A=PEEK(203):REM TASTATURREGISTER LESEN
* 2070 IF A=56 THEN O1%=O1%+O% : REM DREHUNG NACH LI
      NKS
* 2080 IF A=59 THEN O1%=O1%-O% : REM DREHUNG NACH RE
      CHTS
  2090 REM ARBEITSRAUM DREHUNG
  2100 IF O1%<0 THEN O1%=0
* 2120 PRINT CHR$(19)
* 2130 PRINT CHR$(17);CHR$(17);
* 2140 PRINT TAB(30);CL$;O1%
* 2150 SYS P1,O1%
* 2160 IF A=8  THEN O2%=O2%+O% : REM OBERARM NACH VO
      RNE
* 2170 IF A=11 THEN O2%=O2%-O% : REM OBERARM NACH HI
      NTEN
  2180 REM ARBEITSRAUM OBERARM
  2220 IF O2%<0 THEN O2%=0
* 2230 PRINT CHR$(17);CHR$(17);
* 2240 PRINT TAB(30);CL$;O2%
* 2250 SYS P2,O2%
* 2260 IF A=16 THEN O3%=O3%+O% : REM UNTERARM NACH U
      NTEN
* 2270 IF A=19 THEN O3%=O3%-O% : REM UNTERARM NACH O
      BEN
  2280 REM ARBEITSRAUM UNTERARM
  2320 IF O3%<0 THEN O3%=0
* 2330 PRINT CHR$(17);CHR$(17);
* 2340 PRINT TAB(30);CL$;O3%
* 2350 SYS P3,O3%
  2360 REM ROBOTERROUTINE STARTEN
* 2370 SYS ROBOT
  2380 REM ZANGENROUTINE
* 2390 IF USR(E7)=0 THEN PRINT CHR$(17);CHR$(17);TAB
      (28);"AUF"
* 2400 IF USR(E7)=1 THEN PRINT CHR$(17);CHR$(17);TAB
      (28);"ZU "
* 2410 IF A<>24 THEN GOTO 2470
  2420 IF ZA$="AUF" THEN GOTO 2470
  2430 LET ZA$="AUF"
* 2440 SYS M4,RECHTS
* 2450 IF USR(E7)=1 THEN GOTO 2440
* 2460 SYS M4,AUS
* 2470 IF A<>27 THEN GOTO 2540
  2480 IF ZA$="ZU " THEN GOTO 2540
  2490 LET ZA$="ZU "
  2500 FOR Z=1 TO ZZ
* 2510 SYS M4,LINKS
  2520 NEXT
* 2530 SYS M4,AUS
  2540 REM SCHRITTWEITE EINSTELLEN
* 2542 GET A$
  2543 IF A$="+" THEN IF ZZ<500 THEN LET ZZ=ZZ+50
  2547 IF A$="-" THEN IF ZZ>50 THEN LET ZZ=ZZ-50
* 2550 IF A=4 THEN O%=256
* 2560 IF A=5 THEN O%=64
* 2570 IF A=6 THEN O%=16
* 2580 IF A=3 THEN O%=4
* 2590 PRINT CHR$(17);CHR$(17);
* 2600 PRINT TAB(30);CL$;O%
* 2605 PRINT TAB(29);CL$;ZZ
* 2610 IF A=51 THEN GOSUB 3000
  2620 GOTO 2020
  3000 REM HEIMPOSITION ANFAHREN
  3010 LET H1=1:H2=1:H3=1:H4=1
* 3020 IF USR(E1)=1 AND H1=1 THEN SYS M1,RECHTS
* 3030 IF USR(E3)=1 AND H2=1 THEN SYS M2,RECHTS
* 3040 IF USR(E5)=1 AND H3=1 THEN SYS M3,RECHTS
* 3050 IF USR(E7)=1 AND H4=1 THEN SYS M4,RECHTS
* 3060 IF USR(E1)=0 THEN SYS M1,LINKS:H1=-1
* 3070 IF USR(E3)=0 THEN SYS M2,LINKS:H2=-1
* 3080 IF USR(E5)=0 THEN SYS M3,LINKS:H3=-1
* 3090 IF USR(E7)=0 THEN SYS M4,AUS:H4=0
* 3100 IF USR(E1)=1 AND H1=-1 THEN SYS M1,AUS:H1=0
* 3110 IF USR(E3)=1 AND H2=-1 THEN SYS M2,AUS:H2=0
* 3120 IF USR(E5)=1 AND H3=-1 THEN SYS M3,AUS:H3=0
  3130 IF H1<>0 OR H2<>0 OR H3<>0 OR H4<>0 THEN GOTO
       3020
* 3140 SYS INIT
  3150 RETURN
```

Prog. ROBOT.RAUM

```
560 REM UEBERWACHUNG DES ARBEITSRAUM


880 PRINT "UND UEBERWACHUNG DES ARBEITSRAUM"


2110 IF 01%>3600 THEN 01%=3600


2190 IF 02%>370+03% THEN 02%=370+03%
2200 IF 02%>1210 THEN 02%=1210
2210 IF 02%<-1709+1.82*03% THEN 02%=-1709+1.82*03%


2290 IF 03%>940+0.55*02% THEN 03%=940+0.55*02%
2300 IF 03%>1160 THEN 03%=1160
2310 IF 03%<-370+02% THEN 03%=-370+02%
```

Prog. ROBOT.TEACH

```
* 500 SYS INIT
  510 REM
  520 REM FISCHERTECHNIK COMPUTING
  530 REM
  540 REM TRAININGSROBOTER IM TEACH IN MODUS
  550 REM
  560 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1985
  570 REM
  580 REM FUNKTION
  590 REM DAS PROGRAMM ZEIGT ZUNAECHST DAS HAUPTMENU
      E.
  600 REM FOLGENDE FUNKTIONEN KOENNEN GEWAEHLT WERDE
      N:
  610 REM T = ROBOTER TEACH IN MODUS
  620 REM R = TEACH PROGRAMM AUSFUEHREN
  630 REM S = TEACH PROGRAMM ABSPEICHERN
  640 REM L = TEACH PROGRAMM LADEN
  650 REM D = DISKETTENINHALT
  660 REM P = TEACH PROGRAMM AUSDRUCKEN
  670 REM E = PROGRAMMENDE
  680 REM
  690 REM DER ROBOTER TEACH IN MODUS FUEHRT ZU EINEM
       UNTERMENUE ZUR STEUERUNG
  700 REM DES ROBOTERS PER HAND UND ZUR ABSPEICHERUN
      G DER BAHN.
  710 REM TASTENBELEGUNG:
  720 REM 1 UND 2 = M1 RECHTS UND LINKS (DREHUNG)
  730 REM 3 UND 4 = M2 RECHTS UND LINKS (OBERARM)
  740 REM 5 UND 6 = M3 RECHTS UND LINKS (UNTERARM)
  750 REM 7 UND 8 = M4 RECHTS UND LINKS (ZANGE)
* 760 REM F1 = 256 SEKTOREN WEG PRO TASTENDRUCK)
* 770 REM F3 = 64 SEKTOREN WEG PRO TASTENTRUCK
* 780 REM F5 = 16 SEKTOREN WEG PRO TASTENDRUCK
* 790 REM F5 = 4 SEKTOREN WEG PRO TASTENDRUCK
* 800 REM HOME = HEIMPOSITION ANFAHREN
  810 REM RETURN = ABSPEICHERN
* 820 REM DEL = LETZTE ABSPEICHERUNG LOESCHEN
  830 REM M = ZURUECK ZUM HAUPTMENUE
  1000 DIM DR(100),OA(100),UA(100),ZA$(100)
  1010 LET ID=-1:REM ZEIGER IN TEACH TABELLE
  1015 LET ZZ=200: REM ZEITKONSTANTE GREIFZANGE
  1020 GOSUB 10000
  1030 PRINT
* 1040 PRINT CHR$(18)"T"CHR$(146)"   ROBOTER TEACH I
      N MODUS"
  1050 PRINT
* 1060 PRINT CHR$(18)"R"CHR$(146)"   TEACH PROGRAMM
      AUSFUEHREN"
  1070 PRINT
* 1080 PRINT CHR$(18)"S"CHR$(146)"   TEACH PROGRAMM
      ABSPEICHERN"
  1090 PRINT
* 1100 PRINT CHR$(18)"L"CHR$(146)"   TEACH PROGRAMM
      LADEN"
  1110 PRINT
* 1120 PRINT CHR$(18)"D"CHR$(146)"   DISKETTENINHALT
      "
  1130 PRINT
* 1140 PRINT CHR$(18)"P"CHR$(146)"   TEACH PROGRAMM
      AUSDRUCKEN"
  1150 PRINT
* 1160 PRINT CHR$(18)"E"CHR$(146)"   PROGRAMMENDE";
  1170 REM TASTATURABFRAGE
* 1180 GET A$
  1190 IF A$="" THEN GOTO 1180
  1200 IF A$="T" THEN GOTO 2010:REM TEACH MODUS
  1210 IF A$="R" THEN GOTO 4010:REM AUSFUEHRMODUS
  1220 IF A$="S" THEN GOTO 5010:REM ABSPEICHERN
  1230 IF A$="L" THEN GOTO 6010:REM LADEN
  1240 IF A$="D" THEN GOTO 9010:REM DISKETTENINHALT
  1250 IF A$="P" THEN GOTO 7010:REM DRUCKERAUSGABE
  1260 IF A$="E" THEN GOTO 8010:REM PROGRAMMENDE
  1270 GOTO 1180
  2000 REM TEACH IN MODUS
  2010 IF ID=-1 THEN GOTO 2100
* 2020 PRINT CHR$(147)
  2030 FOR I=1 TO 5
  2040 PRINT
  2050 NEXT I
  2060 INPUT"ALTES TEACH PROGRAMM LOESCHEN (J/N)";K$
  2070 IF K$="N"THEN 2130
  2080 IF K$()"J"THEN 2060
  2090 LET ID=-1
* 2100 PRINT CHR$(147)
  2110 PRINT"ROBOTER FAEHRT IN HEIMPOSITION"
  2120 GOSUB 11000: REM HOMEROUTINE
* 2130 PRINT CHR$(147)
  2140 PRINT"TASTENFUNKTIONEN:              POSITION:"
  2150 PRINT
* 2160 PRINT CHR$(18)"1"CHR$(146)"  ROBOTER NACH LIN
      KS"
* 2170 PRINT CHR$(18)"2"CHR$(146)"  ROBOTER NACH REC
      HTS"
* 2180 PRINT CHR$(18)"3"CHR$(146)"  OBERARM VOR"
* 2190 PRINT CHR$(18)"4"CHR$(146)"  OBERARM ZURUECK"
* 2200 PRINT CHR$(18)"5"CHR$(146)"  UNTERARM AB"
* 2210 PRINT CHR$(18)"6"CHR$(146)"  UNTERARM AUF"
* 2220 PRINT CHR$(18)"7"CHR$(146)"  ZANGE AUF"
* 2230 PRINT CHR$(18)"8"CHR$(146)"  ZANGE ZU"
  2240 PRINT
  2250 PRINT "SCHRITTWEITE"
* 2255 PRINT "ZEITKONSTANTE GREIFZANGE  "CHR$(18)"+/
      -"CHR$(146)
  2260 PRINT
* 2270 PRINT CHR$(18)"F1"CHR$(146)"  256 SEGMENTE";
* 2280 PRINT TAB(20);CHR$(18)"M "CHR$(146)"     MENU
      E"
* 2290 PRINT CHR$(18)"F3"CHR$(146)"   64 SEGMENTE  "
      ;
* 2300 PRINT TAB(20);CHR$(18)"HOME"CHR$(146)"   HEIM
      POSITION"
* 2310 PRINT CHR$(18)"F5"CHR$(146)"   16 SEGMENTE";
* 2320 PRINT TAB(20);CHR$(18)"RETURN"CHR$(146)" LERN
      E"
```

```
* 2330 PRINT CHR$(18)"F7"CHR$(146)"    4 SEGMENTE";
* 2340 PRINT TAB(20);CHR$(18)"DEL"CHR$(146)"    LOES
      CHEN"
  2350 PRINT
  2360 PRINT
  2370 PRINT" NR DREH      OARM      UARM      ZANGE";
  2380 PRINT"
       ";
* 2390 PRINT CHR$(145);
  3000 LET Q%=16
* 3010 LET CL$="      "+CHR$(157)+CHR$(157)+CHR$(157)
      +CHR$(157)+CHR$(157)
  3020 REM ABFRAGESCHLEIFE
* 3030 LET Q1%=USR(P1)
* 3040 LET Q2%=USR(P2)
* 3050 LET Q3%=USR(P3)
  3060 REM TASTATURABFRAGE (BEWEGUNGEN)
* 3070 LET A=PEEK(203):REM TASTATURREGISTER LESEN
* 3080 IF A=56 THEN Q1%=Q1%+Q% : REM TASTE LINKS DRE
      HEN
* 3090 IF A=59 THEN Q1%=Q1%-Q% : REM TASTE RECHTS DR
      EHEN
  3100 REM ARBEITSRAUM DREHUNG
  3110 IF Q1%<0 THEN Q1%=0
  3120 IF Q1%>3600 THEN Q1%=3600
* 3130 PRINT CHR$(19)
* 3140 PRINT CHR$(17);CHR$(17);
* 3150 PRINT TAB(30);CL$;Q1%
* 3160 SYS P1,Q1%
* 3170 IF A=8   THEN Q2%=Q2%+Q% : REM OBERARM VOR
* 3180 IF A=11  THEN Q2%=Q2%-Q% : REM OBERARM ZURUECK
  3190 REM ARBEITSRAUM OBERARM
  3200 IF Q2%>370+Q3% THEN Q2%=370+Q3%
  3210 IF Q2%>1210 THEN Q2%=1210
  3220 IF Q2%<-1709+1.82*Q3% THEN Q2%=-1709+1.82*Q3%
  3230 IF Q2%<0 THEN Q2%=0
* 3240 PRINT CHR$(17);
* 3250 PRINT TAB(30);CL$;Q2%
* 3260 SYS P2,Q2%
* 3270 IF A=16 THEN Q3%=Q3%+Q% : REM UNTERARM AB
* 3280 IF A=19 THEN Q3%=Q3%-Q% : REM UNTERARM AUF
  3290 REM ARBEITSRAUM UNTERARM
  3300 IF Q3%>940+0.55*Q2% THEN Q3%=940+0.55*Q2%
  3310 IF Q3%>1160 THEN Q3%=1160
  3320 IF Q3%<-370+Q2% THEN Q3%=-370+Q2%
  3330 IF Q3%<0 THEN Q3%=0
* 3340 PRINT CHR$(17);
* 3350 PRINT TAB(30);CL$;Q3%
* 3360 SYS P3,Q3%
  3370 REM ROBOTERBEWEGUNG STARTEN
* 3380 SYS ROBOT
  3390 REM ROUTINE ZANGE
* 3400 IF USR(E7)=0 THEN PRINT CHR$(17);TAB(30);"AUF
      "
* 3410 IF USR(E7)=1 THEN PRINT CHR$(17);TAB(30);"ZU
      "
* 3420 IF A<>24 THEN GOTO3480
  3430 IF ZA$="AUF" THEN GOTO 3480
  3440 LET ZA$="AUF"
* 3450 SYS M4,RECHTS
* 3460 IF USR(E7)=1 THEN GOTO3450
* 3470 SYS M4,AUS
* 3480 IF A<>27 THEN GOTO 3560
  3490 IF ZA$="ZU " THEN GOTO 3560
  3500 LET ZA$="ZU "
  3510 FOR Z=1 TO 1.4*ZZ
* 3520 SYS M4,LINKS
  3530 NEXT
* 3540 SYS M4,AUS
  3550 REM SCHRITTWEITE EINSTELLEN
* 3560 GET A$:REM TASTATURABFRAGE(KOMMANDOS)
  3563 IF A$="+" THEN IF ZZ<500 THEN LET ZZ=ZZ+50
  3567 IF A$="-" THEN IF ZZ>50 THEN LET ZZ=ZZ-50
* 3570 IF A$=CHR$(133) THEN Q%=256
* 3580 IF A$=CHR$(134) THEN Q%=64
* 3590 IF A$=CHR$(135) THEN Q%=16
* 3600 IF A$=CHR$(136) THEN Q%=4
* 3610 PRINT CHR$(17);CHR$(17);
* 3620 PRINT TAB(30);CL$;Q%
* 3625 PRINT TAB(29);CL$;ZZ
* 3630 IF A$=CHR$(19) THEN GOSUB 11000:REM HOME TAST
      E
  3640 IF A$<>"M" THEN GOTO 3680: REM M =MENUETASTE
  3650 LET IMAX=ID
* 3660 PRINT CHR$(147)
  3670 GOTO 1020
* 3680 IF A$<>CHR$(13) THEN GOTO 3850:REM LERNETASTE
  3690 REM AKUST.SIGNAL 'ABSPEICHERN'
* 3700 S=54272
* 3710 POKE S+24,15
* 3720 POKE S+6,240
* 3730 POKE S+1,90
* 3740 POKE S+4,17
* 3750 POKE S+24,0:POKE S+1,0:POKE S+4,0:POKE S+6,0
  3760 REM ABLAGE IN TEACH TABELLE
  3770 LET ID=ID+1
* 3780 LET DR(ID)=USR(P1)
* 3790 LET OA(ID)=USR(P2)
* 3800 LET UA(ID)=USR(P3)
  3810 LET ZA$(ID)=ZA$
  3820 GOSUB 12010: REM AKTUELLE POSITION AUSDRUCKEN
  3830 GOTO 3030
  3840 REM LETZTE POSITION LOESCHEN
* 3850 IF A$<>CHR$(20) THEN GOTO 3030
  3860 IF ID>0 THEN LET ID=ID-1
  3870 REM AKUST. SIGNAL 'LOESCHEN'
* 3880 POKE S+24,15
* 3890 POKE S+6,240
* 3900 POKE S+1,70
* 3910 POKE S+4,17
* 3920 POKE S+24,0:POKE S+1,0:POKE S+4,0:POKE S+6,0
  3930 GOSUB 12010: REM AKTUELLE POSITION AUSDRUCKEN
  3940 GOTO 3030
  4000 REM AUSFUEHRMODUS
  4010 GOSUB 10000 : REM TITELMELDUNG
  4020 FOR T=1 TO5
* 4030 PRINT CHR$(17)
  4040 NEXT T
  4050 PRINT"> M <  MENUE"
* 4060 PRINT CHR$(19)
  4070 FOR U =1 TO 5
* 4080 PRINT CHR$(17)
  4090 NEXT U
  4100 PRINT
  4110 PRINT"            AUSFUEHRMODUS"
  4120 PRINT
  4130 INPUT"WIEVIELE DURCHLAEUFE ";D
  4140 FOR Y=1 TO D
  4150 GOSUB 11000: REM HOMEROUTINE
* 4160 PRINT CHR$(147)
  4170 PRINT"PROGRAMMTABELLE"
  4180 PRINT
  4190 PRINT" NR DREH      OARM     UARM     ZANGE"
  4200 PRINT
  4210 FOR I=0 TO IMAX
* 4220 GET A$
  4230 IF A$="M" THEN 1020
  4240 PRINT I;TAB(3);DR(I);TAB(12);OA(I);TAB(21);UA
      (I);TAB(31);ZA$(I)
* 4245 I1=USR(P1): IF I1<0 THEN LET I1=0
* 4250 SYS P1,I1: IF ABS(DR(I)-I1)>10 THEN SYS P1,DR
      (I)
* 4255 I2=USR(P2): IF I2<0 THEN LET I2=0
* 4260 SYS P2,I2: IF ABS(OA(I)-I2)>10 THEN SYS P2,OA
      (I)
* 4265 I3=USR(P3): IF I3<0 THEN LET I3=0
* 4270 SYS P3,I3: IF ABS(UA(I)-I3)>10 THEN SYS P3,UA
      (I)
* 4280 SYS ROBOT
  4290 REM ZANGENROUTINE
  4300 IF ZA$(I)<>"ZU "THEN GOTO4360
  4310 IF ZA$="ZU " THEN4360
  4320 FOR Z=1 TO ZZ
* 4330 SYS M4,LINKS
  4340 NEXT Z
  4350 LET ZA$="ZU "
  4360 IF ZA$(I)<>"AUF" THEN GOTO 4420
  4370 IF ZA$="AUF" THEN 4420
* 4380 SYS M4,RECHTS
* 4390 IF USR(E7)=1 THEN GOTO4370
  4400 LET ZA$="AUF"
* 4410 SYS M4,AUS
  4420 NEXT I
  4430 NEXT Y
  4440 GOTO 1020
  5000 REM TEACH PROGRAMM ABSPEICHERN
* 5010 PRINT CHR$(147)
  5020 PRINT
  5030 PRINT"TEACH PROGRAMM AUF DISKETTE ABSPEICHERN
       "
  5040 PRINT
```

```
5050 LET F$=""
5060 INPUT"FILENAME";F$
5070 IF F$="" THEN GOTO 1020
5080 OPEN15,8,15
5090 OPEN 8,8,8,F$+",W"
5100 INPUT#15,FE,FT$,SP,SE
5110 IF FE=0 THEN GOTO 5200
5120 IF FE<>63 THEN GOTO 6250
5130 PRINT"FILE EXISTIERT BEREITS."
5140 INPUT"ALTES FILE LOESCHEN(J/N)";C$
5150 IF C$="N" THEN GOTO 6200
5160 IF C$<>"J" THEN 5270
5170 PRINT#15,"S:"+F$
5180 CLOSE 8
5190 OPEN 8,8,8,F$+",W"
5200 PRINT#8,IMAX
5210 FOR I=0 TO IMAX
5220 PRINT#8,DR(I)
5230 PRINT#8,OA(I)
5240 PRINT#8,UA(I)
5250 PRINT#8,ZA$(I)
5260 NEXT I
5270 CLOSE 8
5280 CLOSE 15
5290 GOTO 1020
6000 REM TEACH PROGRAMM LADEN
6010 PRINT CHR$(147)
6020 PRINT
6030 PRINT"TEACH PROGRAMM VON DISKETTE LADEN"
6040 PRINT
6050 LET F$=""
6060 INPUT"FILENAME";F$
6070 IF F$="" THEN GOTO 1020
6080 OPEN 8,8,8,F$+",R"
6090 OPEN 15,8,15
6100 INPUT#15,FE,FT$,SP,SE
6110 IF FE<>0 THEN GOTO 6250
6120 INPUT#8,IMAX
6130 PRINT IMAX;"POSITIONSDATEN"
6140 FOR I=0 TO IMAX
6150 INPUT#8,DR(I)
6160 INPUT#8,OA(I)
6170 INPUT#8,UA(I)
6180 INPUT#8,ZA$(I)
6190 NEXT I
6200 CLOSE 8
6210 CLOSE 15
6220 LET ID=IMAX:ZA$=ZA$(IMAX)
6230 GOTO 1020
6240 REM DISKETTE FEHLERMELDUNG
6250 PRINT FT$
6260 PRINT"> M < MENUE"
6270 GET A$
6280 IF A$="M" THEN GOTO 6200
6290 GOTO 6270
7000 REM TEACH PROGRAMM AUSDRUCKEN
7010 PRINT CHR$(147)
7020 PRINT"SOLL DER AUSDRUCK AUF DRUCKER ODER "
7030 INPUT"BILDSCHIRM ERFOLGEN (D/B)";S$
7040 IF S$="D" THEN 7180
7050 GOSUB 10010
7060 PRINT
7070 PRINT"TEACH TABELLE"
7080 PRINT
7090 PRINT"NR. DREH     OARM     UARM     ZANGE"
7100 FOR I=0 TO IMAX
7110 PRINT I;TAB(3);DR(I);TAB(12);OA(I);TAB(21);UA
     (I);TAB(31);ZA$(I)
7120 NEXT I
7130 PRINT
7140 PRINT"> M < MENUE"
7150 GET A$
7160 IF A$="M" THEN GOTO 1020
7170 GOTO7150
7180 OPEN 4,4,0
7190 PRINT#4,"F I S C H E R T E C H N I K"
7200 PRINT#4
7210 PRINT#4,"    C O M P U T I N G"
7220 PRINT#4
7230 PRINT#4,"3 ACHSIGER TRAININGSROBOTER"
7240 PRINT#4
7250 PRINT#4,"TEACH TABELLE"
7260 PRINT#4,"              "
7270 PRINT#4," NR.          DREH          OARM
          UARM          ZANGE"
7280 FOR I=0 TO IMAX
7290 PRINT#4,I,DR(I),OA(I),UA(I),ZA$(I)
7300 NEXT I
7310 CLOSE 4
7320 GOTO 1020
8000 REM PROGRAMMENDE
8010 PRINT CHR$(147)
8020 FOR I=1 TO 12
8030 PRINT
8040 NEXT
8050 INPUT" SIND SIE SICHER (J/N)";L$
8060 IF L$="N" THEN 1020
8070 IF L$<>"J" THEN GOTO 8050
8080 PRINT CHR$(147)
8090 END
9000 REM DISKETTENINHALT
9010 PRINT CHR$(147)
9020 PRINT"FISCHERTECHNIK"
9030 PRINT"COMPUTING"
9040 PRINT
9050 OPEN1,8,0,"$0"
9060 GET#1,A$,B$
9070 GET#1,A$,B$
9080 GET#1,A$,B$
9090 C=0 : C$=""
9100 IF A$<>"" THEN C=ASC(A$)
9110 IF B$<>"" THEN C=C+ASC(B$)*256
9120 PRINT MID$(STR$(C),2);TAB(3);
9130 GET#1,B$:IF ST<>0 THEN 9190
9140 IF B$<>CHR$(34) THEN 9130
9150 GET#1,B$:IF B$<>CHR$(34) THEN C$=C$+B$:GOTO 9
     150
9160 GET#1,B$:IF B$<>"" THEN 9160
9170 PRINT C$
9180 IF ST=0 THEN 9070
9190 PRINT" BLOCKS FREE"
9200 CLOSE 1
9210 PRINT
9220 PRINT"  > M <  MENUE"
9230 GET Z$
9240 IF Z$<>"M" THEN 9230
9250 PRINT CHR$(147)
9260 GOTO 1020
10000 REM TITELMELDUNG
10010 PRINT CHR$(147)
10020 PRINT CHR$(28)"      F I S C H E R"CHR$(31)"
      T E C H N I K"CHR$(144)
10030 PRINT
10040 PRINT"           C O M P U T I N G"
10050 PRINT:PRINT
10060 PRINT   "        D R E I A C H S I G E R"
10070 PRINT
10080 PRINT"    T R A I N I N G S R O B O T E R"
10090 PRINT
10100 PRINT"           TEACH IN VERFAHREN "
10110 RETURN
11000 REM HEIMPOSITION ANFAHREN
11010 LET H1=1:H2=1:H3=1:H4=1
11020 IF USR(E1)=1 AND H1=1 THEN SYS M1,RECHTS
11030 IF USR(E3)=1 AND H2=1 THEN SYS M2,RECHTS
11040 IF USR(E5)=1 AND H3=1 THEN SYS M3,RECHTS
11050 IF USR(E7)=1 AND H4=1 THEN SYS M4,RECHTS
11060 IF USR(E1)=0 THEN SYS M1,LINKS:H1=-1
11070 IF USR(E3)=0 THEN SYS M2,LINKS:H2=-1
11080 IF USR(E5)=0 THEN SYS M3,LINKS:H3=-1
11090 IF USR(E7)=0 THEN SYS M4,AUS:H4=0
11100 IF USR(E1)=1 AND H1=-1 THEN SYS M1,AUS:H1=0
11110 IF USR(E3)=1 AND H2=-1 THEN SYS M2,AUS:H2=0
11120 IF USR(E5)=1 AND H3=-1 THEN SYS M3,AUS:H3=0
11130 IF H1<>0 OR H2<>0 OR H3<>0 OR H4<>0 THEN GOT
      O 11020
11140 SYS INIT
11150 LET ZA$="AUF"
11160 RETURN
12000 REM AKTUELLE POSITION AUSDRUCKEN
12010 PRINT CHR$(19);
12020 FOR I=1 TO 23
12030 PRINT CHR$(17);
12040 NEXT
12050 PRINT"
      "
12060 PRINT CHR$(145);
12070 PRINT ID;TAB(3);DR(ID);TAB(12);OA(ID);TAB(21
      );UA(ID);TAB(31);ZA$(ID)
12080 RETURN
```

Funktionsweise des Interface und des Roboter-Systemprogramms

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.

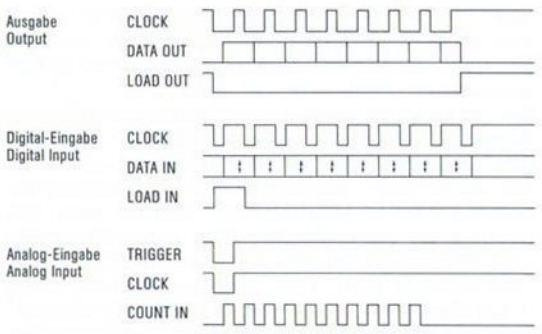
Ob die Daten allerdings auch die Leistungsverstärker durchsteuern, hängt wiederum von der Freigabesteuerung des Speicherbausteins ab. Die Freigabesteuerung erfolgt durch ein Monoflop. Diese Schaltung erzeugt ein Freigabesignal von einer halben Sekunde Dauer, wenn ein Impuls auf der CLOCK-Leitung vorliegt. 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.

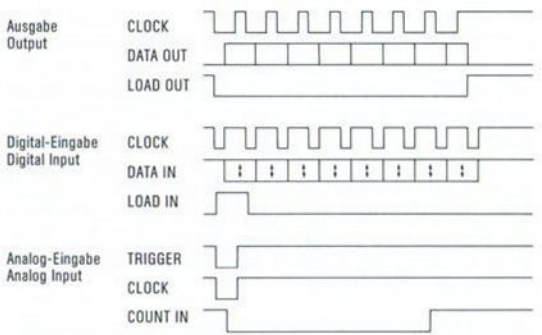
Nun zu der Übertragung der digitalen Signale an E1 bis E8. Im Prinzip findet bei der Eingabe eine Umkehrung des oben Beschriebenen statt. Durch das Ausgabe-Signal LOAD-IN werden die an den Eingängen anstehenden Signale in das Eingabeschieberegister übernommen. Dies erfolgt wiederum für alle acht Eingänge, 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,

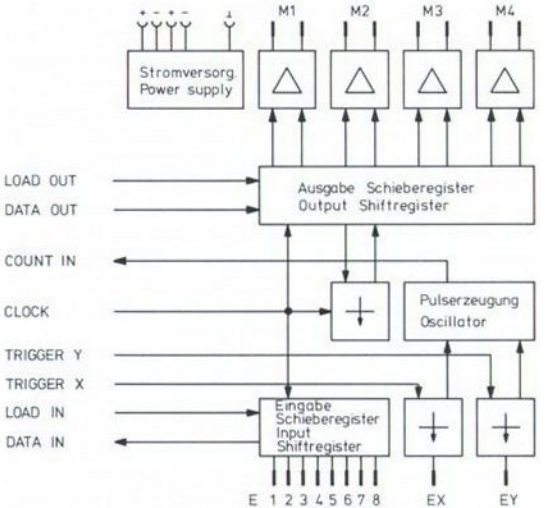
nicht aber letzteres, das ja nur auf das Signal LOAD-OUT reagiert.
Bleiben zum Schluß noch die Analogeingänge EX und EY. Potentiometer oder sonstige veränderlichen Widerstände dienen als zeitbestimmendes Bauelement in zwei weiteren Monoflop-Schaltungen. Ein niedriger Widerstandswert wird in einem Impuls 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. Dennoch besteht zwischen der letztlich ermittelten Zahl und dem Widerstandswert ein linearer Zusammenhang. Dieser muß gegebenenfalls im BASIC-Programm noch anhand einer Eichung in Winkelgrade oder Widerstandswerte umgerechnet werden.
Auf den folgenden Seiten ist der Quelltext des Robotersystemprogramms angegeben. Aus Platzgründen können wir nicht alle Versionen des Roboter-Systemprogramms für die verschiedenen Computer abdrucken. Außerdem sind die Unterschiede nur geringfügig. Stellvertretend für Computer mit Mikroprozessoren der 6502-Familie geben wir hier das Roboter-Systemprogramm des Commodore 64 an. Ein anderer weitverbreiteter Mikroprozessor ist der Z80. Das Roboter-Systemprogramm des Schneider CPC steht stellvertretend für all jene Computer.



Impulsdiagramme des fischertechnik Interface
Pulse Diagrams of the fischertechnik Interface
C 64, VIC 20, Acorn



Impulsdiagramme des fischertechnik Interface
Pulse Diagrams of the fischertechnik Interface
IBM-PC, CPC, CBM 4/8 xxx



Prog. ROBOT.SYS (6502)

```
                0010            ;PROGRAMM C64 INTERFACE
                0020            ;COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1984
                0030            ;VERSION 9 INKLUSIVE ROBOTERSTEUERUNG
                0040            ;FILE C64IF9A
                0050            ;MAI 1985
                0060            ;
                0070            ;AUFRUF DES FISCHERTECHNIK INTERFACE
                0080            ;VOM C64 DURCH KOMMANDOS
                0090            ;SYS M1,EIN       SYS M1,AUS
                0100            ;SYS M1,LINKS     SYS M1,RECHTS
                0110            ;USR(E1)      USR(EX)     USR(EY)
                0120            ;SPEZIELLE ROBOTERBEFEHLE:
                0130            ;SYS P1,NNNN  SOLLPOSITION ABLEGEN
                0140            ;USR(P1) ISTPOSITION ABFRAGEN
                0150            ;SYS ROBOT START DES ROBOTERS
                0160            ;**************************************
                0170            .OS                     ;OBJECTCODE ERZEUGEN
                0180            .BA $CDB0               ;PROGRAMM IM FREISPEICHER
                0190            ;**************************************
                0200 YFAC       .DE $B3A2               ;WANDLE Y IN FLIESSKOMMA
                0210 AYFAC      .DE $B391               ;WANDLE A/Y IN FLIESSKOMMA
                0220 CKCOM      .DE $AEFD               ;PRUEFE AUF KOMMA
                0230 F16INT     .DE $B7F7               ;WANDLE FLIESSKOMMA IN INT
                0240 GETBYTE    .DE $B79E               ;LIES EIN BYTE EIN
                0250 GETWORD    .DE $AD8A               ;LIES EIN WORT EIN
                0260 ROPOS      .DE $CE98               ;ROBOTER STEUERN (TEIL B)
                0270 ROPOSL     .DE $CFD0               ;ROBOTER TABELLE (TEIL B)
                0280            ;**************************************
                0290            ; EIN- AUSGABEREGISTER
                0300            ;**************************************
                0310 UP         .DE $DD01               ;USER PORT DATENREGISTER
                0320 DRR        .DE $DD03               ;USER PORT DATENRICHTUNG
                0330 TIL        .DE $DD04               ;TIMER LOW
                0340 TIH        .DE $DD05               ;TIMER HIGH
                0350 TIC        .DE $DD0E               ;TIMER CTRL REG
                0360            ;**************************************
                0370            ; VARIABLEN
                0380            ;**************************************
CDB0- 00        0390 AVAR       .BY $00                 ;AUSGABEVARIABLE
CDB1- 00        0400 MASK       .BY $00                 ;MASKENVARIABLE
                0410            ;**************************************
                0420            ; EINSPRUNGLEISTE
CDB2- A9 00     0430 INIT       LDA #$00                ;INITIALISIERUNG
CDB4- A2 17     0440            LDX #$17                ;TABELLENZEIGER
CDB6- 9D D0 CF  0450 CLOOP      STA ROPOSL,X            ;TABELLEN DER
CDB9- CA        0460            DEX                     ;ROBOTERPOSITIONEN
CDBA- 10 FA     0470            BPL CLOOP               ;LOESCHEN
CDBC- 30 2E     0480            BMI STVAR               ;BRANCH ALWAYS
CDBE- A9 03     0490 M1         LDA #%00000011          ;MOTOR 1
CDC0- D0 0A     0500            BNE BOUT
CDC2- A9 0C     0510 M2         LDA #%00001100          ;MOTOR 2
CDC4- D0 06     0520            BNE BOUT
CDC6- A9 30     0530 M3         LDA #%00110000          ;MOTOR 3
CDC8- D0 02     0540            BNE BOUT
CDCA- A9 C0     0550 M4         LDA #%11000000          ;MOTOR 4
                0560            ;**************************************
CDCC- 78        0570 BOUT       SEI                     ;INTERRUPT SPERREN
CDCD- 9D B1 CD  0580            STA MASK                ;SPEICHERE BITMASKE AB
CDD0- 20 FD AE  0590            JSR CKCOM               ;PRUEFE AUF KOMMA
CDD3- AD B0 CD  0600            LDA AVAR                ;AUSGABEVARIABLE
CDD6- 0D B1 CD  0610            ORA MASK                ;SETZE BIT
CDD9- 8D B0 CD  0620            STA AVAR                ;ZURUECK
CDDC- 20 9E B7  0630            JSR GETBYTE             ;LIES 2. ARGUMENT
CDDF- 8A        0640            TXA
CDE0- 2D B1 CD  0650            AND MASK                ;BLENDE MOTOR AUS
CDE3- 8D B1 CD  0660            STA MASK                ;ABSPEICHERN
CDE6- AD B0 CD  0670            LDA AVAR                ;AUSGABEVARIABLE
CDE9- 4D B1 CD  0680            EOR MASK                ;SETZE BIT
CDEC- 20 F1 CD  0690 STVAR      JSR SHOUT               ;AUSGABE AN INTERFACE
CDEF- 58        0700            CLI                     ;INTERRUPT FREIGEBEN
CDF0- 60        0710            RTS                     ;ZURUECK INS BASIC
                0720            ;**************************************
                0730            ;ROUTINE ZUR INTERFACESTEUERUNG
                0740            ;AUSGABE
                0750            ;AUSGABEMUSTER WIRD IM AKKU UEBERGEBEN
                0760            ;SEITENEFFEKT: SPEICHERT AKKU IN AVAR AB
                0770            ;BENUTZT AKKU UND X-REG
                0780            ;**************************************
CDF1- 8D B0 CD  0790 SHOUT      STA AVAR                ;AUSGABEWORT RETTEN
CDF4- 48        0800            PHA                     ;AKKU RETTEN
CDF5- A9 3F     0810            LDA #$3F                ;SETZE DATENRICHTUNG
CDF7- 8D 03 DD  0820            STA DRR
CDFA- A2 08     0830            LDX #$08                ;SCHLEIFENZAEHLER
CDFC- A9 30     0840 LOOP       LDA #$30                ;RUHEBITMUSTER USERPORT
CDFE- 0E B0 CD  0850            ASL AVAR                ;TESTE AUSGABEWORT
CE01- 90 02     0860            BCC NULL
CE03- 09 04     0870            ORA #$04                ;SETZE DATA OUT
CE05- 8D 01 DD  0880 NULL       STA UP                  ;AUSGABE
CE08- 09 08     0890            ORA #$08                ;SETZE CLOCK
CE0A- 8D 01 DD  0900            STA UP                  ;AUSGABE
CE0D- CA        0910            DEX
CE0E- D0 EC     0920            BNE LOOP                ;SCHLEIFENENDE
CE10- A9 39     0930            LDA #$39                ;SETZE LOAD OUT
CE12- 8D 01 DD  0940            STA UP                  ;AUSGABE
CE15- 68        0950            PLA                     ;AKKU RESTAURIEREN
CE16- 8D B0 CD  0960            STA AVAR                ;RESTAURIERE AVAR
CE19- 60        0970            RTS                     ;RUECKSPRUNG
                0980            ;**************************************
                0990            ;EINGABROUTINE
                1000            ;EINSPRUNG UEBER USR
                1010            ;**************************************
CE1A- 78        1020 BINP       SEI                     ;INTERRUPT SPERREN
CE1B- 20 F7 B7  1030            JSR F16INT              ;ARG NACH INTEGER WANDELN
CE1E- C9 00     1040            CMP #00                 ;HIGH BYTE GESETZT
CE20- D0 76     1050            BNE ROPOS               ;ABFRAGE ROBOTERPOSITION
CE22- C0 A2     1060            CPY #$A2                ;POTIABFRAGE?
CE24- F0 39     1070            BEQ POTI                ;X
CE26- C0 92     1080            CPY #$92                ;POTIABFRAGE?
CE28- F0 35     1090            BEQ POTI                ;Y
CE2A- 8C B1 CD  1100            STY MASK                ;INPUTMASKE ABSPEICHERN
CE2D- 20 3D CE  1110            JSR SHIN                ;EINGABE ROUTINE
CE30- 2D B1 CD  1120            AND MASK                ;BIT HERAUSMASKIEREN
CE33- A8        1130            TAY                     ;WERT IN Y-REGISTER
CE34- F0 02     1140            BEQ CVAR
```

```
CE36- A0 01     1150          LDY #$01            ;<>0 -> 1
CE38- 20 A2 B3  1160 CVAR     JSR YFAC            ;WANDLE Y IN FAC
CE3B- 58        1170          CLI                 ;INTERRUPT FREIGEBEN
CE3C- 60        1180          RTS                 ;ZURUECK INS BASIC
                1190          ;*************************************
                1200          ;INTERFACE STEUERUNG
                1210          ;EINGABE
                1220          ;BENUTZT WERDEN AKKU, X-REG. UND Y-REG.
                1230          ;ARGUMENT BEI RUECKKEHR IM AKKU
                1240          ;*************************************
CE3D- A9 32     1250 SHIN     LDA #$32            ;SETZE LOAD IN
CE3F- 8D 01 DD  1260          STA UP              ;AUSGABE
CE42- 09 08     1270          ORA #$08            ;SETZE CLOCK
CE44- 8D 01 DD  1280          STA UP              ;AUSGABE
CE47- A2 08     1290          LDX #8              ;SCHLEIFENZAEHLER
CE49- 0A        1300 LOOP2    ASL A               ;SCHIEBE AKKU HOCH
CE4A- 2C 01 DD  1310          BIT UP              ;TESTE UP BIT7
CE4D- 10 02     1320          BPL NULL2
CE4F- 09 01     1330          ORA #$01            ;SETZE BIT
CE51- A0 30     1340 NULL2    LDY #$30            ;SETZE CLOCK
CE53- 8C 01 DD  1350          STY UP              ;AUSGABE
CE56- A0 38     1360          LDY #$38            ;SETZE CLOCK
CE58- 8C 01 DD  1370          STY UP              ;AUSGABE
CE5B- CA        1380          DEX
CE5C- D0 EB     1390          BNE LOOP2           ;SCHLEIFENENDE
CE5E- 60        1400          RTS                 ;RUECKSPRUNG
                1410          ;*************************************
                1420          ;POTENTIOMETERABFRAGE
                1430          ;HIERHER WIRD NACH USR VERZWEIGT
                1440          ;WENN DAS ARGUMENT $92 ODER $A2 IST
                1450          ;*************************************
CE5F- A9 FF     1460 POTI     LDA #$FF            ;SETZE ZAEHLERREG. AUF $FF
CE61- 8D 04 DD  1470          STA TIL
CE64- 8D 05 DD  1480          STA TIH
CE67- A9 B9     1490          LDA #$B9            ;SETZE TIMER CTRL REG
CE69- 8D 0E DD  1500          STA TIC
CE6C- 8C 01 DD  1510          STY UP              ;MONOFLOP TRIGGERN
CE6F- A0 3A     1520          LDY #$3A            ;TRIGGER WEGNEHMEN
CE71- 8C 01 DD  1530          STY UP              ;AUSGABE
CE74- AD 04 DD  1540 TST      LDA TIL             ;LADE ZAEHLER LOW BYTE
CE77- A2 03     1550          LDX #$03            ;VERZOEGERUNGSSCHLEIFE
CE79- CA        1560 VERZOEG  DEX
CE7A- D0 FD     1570          BNE VERZOEG
CE7C- 38        1580          SEC                 ;SUBTRAKTION
CE7D- ED 04 DD  1590          SBC TIL             ;LAEUFT DER ZAEHLER NOCH
CE80- D0 F2     1600          BNE TST
CE82- A2 38     1610          LDX #$38            ;SETZE CLOCK, LOESCHE LOAD
CE84- 8E 01 DD  1620          STX UP              ;AUSGABE
CE87- 38        1630          SEC                 ;SUBTRAKTION VORBEREITEN
CE88- A9 FF     1640          LDA #$FF
CE8A- ED 04 DD  1650          SBC TIL             ;BESTIMME DIFFERENZZEIT
CE8D- A8        1660          TAY
CE8E- A9 FF     1670          LDA #$FF
CE90- ED 05 DD  1680          SBC TIH             ;HIGH BYTE
CE93- 20 91 B3  1690          JSR AYFAC           ;WANDLE A/Y IN FAC
CE96- 58        1700          CLI                 ;INTERRUPT FREIGEBEN
CE97- 60        1710          RTS                 ;ZURUECK INS BASIC
                1720          .EN

 --- LABEL FILE: ---

AVAR =CDB0      AYFAC =B391     BINP =CE1A
BOUT =CDCC      CKCOM =AEFD     CLOOP =CDB6
CVAR =CE38      DRR =DD03       F16INT =B7F7
GETBYTE =B79E   GETWORD =AD8A   INIT =CDB2
LOOP =CDFC      LOOP2 =CE49     M1 =CDBE
M2 =CDC2        M3 =CDC6        M4 =CDCA
MASK =CDB1      NULL =CE05      NULL2 =CE51
POTI =CE5F      ROPOS =CE98     ROPOSL =CFD0
SHIN =CE3D      SHOUT =CDF1     STVAR =CDEC
TIC =DD0E       TIH =DD05       TIL =DD04
TST =CE74       UP =DD01        VERZOEG =CE79
YFAC =B3A2
//0000,CE98,CE98

                0020          ;ROBOTER ROUTINEN
                0030          ;
                0040          ;FILE C64IF9B
                0050          ;COPYRIGHT ARTUR FISCHER FORSCHUNG
                0060          ;MAI 1985
                0070          ;*************************************
                0080          ;SPRUNGADRESSEN
                0090          ;*************************************
                0100 SHOUT    .DE $CDF1           ;ADRESSE AUS TEIL A
                0110 SHIN     .DE $CE3D           ;ADRESSE AUS TEIL A
                0120 AYFAC    .DE $B391           ;WANDLE A/Y IN REAL
                0130 CKCOM    .DE $AEFD           ;PRUEFE AUF KOMMA
                0140 GETWORD  .DE $AD8A           ;LIES 16 BIT INTEGER
                0150 F16INT   .DE $B7F7           ;WANDLE FAC IN 16 BIT INT.
                0160 AVAR     .DE $CDB0           ;ADRESSE AUS TEIL A
                0170 MASK     .DE $CDB1           ;ADRESSE AUS TEIL A
                0180          ;*****************************
                0190          ;ANFANGSADRESSE NACH TEIL A
                0200          ;*****************************
                0210          .OS                 ;OBJECTCODE ERZEUGEN
                0220          .BA $CE98           ;ANFANGSADRESSE
                0230          ;*****************************
                0240          ;ROBOTERROUTINE
                0250          ;
                0260          ;ABFRAGE ISTWERT
                0270          ;*****************************
CE98- A2 00     0280 ROPOS    LDX #$00            ;ZEIGER=0
CE9A- C0 C2     0290          CPY #L,P1           ;WELCHE POSITION
CE9C- F0 12     0300          BEQ LOPOS
```

```
CE9E- A2 02     0310           LDX #$02            ;ZEIGER=2
CEA0- C0 C6     0320           CPY ML,P2
CEA2- F0 0C     0330           BEQ LOPOS
CEA4- A2 04     0340           LDX #$04            ;ZEIGER=4
CEA6- C0 CA     0350           CPY ML,P3
CEA8- F0 06     0360           BEQ LOPOS
CEAA- A2 06     0370           LDX #$06            ;ZEIGER=6
CEAC- C0 CE     0380           CPY ML,P4
CEAE- D0 0B     0390           BNE SYNTAX
CEB0- BC D0 CF  0400 LOPOS     LDY ROPOSL,X        ;LADE LOW-BYTE
CEB3- BD D1 CF  0410           LDA ROPOSH,X        ;LADE HIGH BYTE
CEB6- 20 91 B3  0420           JSR AYFAC           ;WANDLE IN REAL
CEB9- 58        0430           CLI                 ;INTERRUPT FREIGEBEN
CEBA- 60        0440           RTS                 ;ZURUECK INS BASIC
CEBB- 8C B0 CD  0450 SYNTAX    STY AVAR
CEBE- 8D B1 CD  0460           STA MASK
CEC1- 60        0470           RTS
                0480           ;*****************************
                0490           ;ROBOTERROUTINE
                0500           ;
                0510           ;SETZE SOLLWERT FUER ROBOTERPOSITION
                0520           ;*****************************
CEC2- A9 00     0530 P1        LDA #$00
CEC4- F0 0A     0540           BEQ STOPOS
CEC6- A9 02     0550 P2        LDA #$02
CEC8- D0 06     0560           BNE STOPOS
CECA- A9 04     0570 P3        LDA #$04
CECC- D0 02     0580           BNE STOPOS
CECE- A9 06     0590 P4        LDA #$06
CED0- 8D B1 CD  0600 STOPOS    STA MASK            ;ZEIGER RETTEN
CED3- 20 FD AE  0610           JSR CKCOM           ;PRUEFE AUF KOMMA
CED6- 20 8A AD  0620           JSR GETWORD         ;LIES ARGUMENT
CED9- 20 F7 B7  0630           JSR F16INT          ;WANDLE ARGUMENT NACH INT.
CEDC- AE B1 CD  0640           LDX MASK            ;ZEIGER HOLEN
CEDF- 9D D9 CF  0650           STA ROSOLH,X        ;SPEICHERE HIGH BYTE
CEE2- 98        0660           TYA
CEE3- 9D D8 CF  0670           STA ROSOLL,X        ;SPEICHERE LOW BYTE
CEE6- 60        0680           RTS                 ;ZURUECK INS BASIC
                0690           ;*****************************
                0700           ;ROBOTERSTEUERUNG
                0710           ;
                0720           ;DIE ROUTINE VERGLEICHT DIE IN
                0730           ;ROSOL ABGESPEICHERTEN WERTE MIT
                0740           ;JENEN IN ROPOS. AUS DER DIFFERENZ
                0750           ;WIRD DIE MOTORDREHRICHTUNG BESTIMMT.
                0760           ;DIE MOTOREN MIT DIFFERENZ <>0 WERDEN
                0770           ;GESTARTET UND DIE IMPULSE DER LICHT-
                0780           ;SCHRANKE GEZAEHLT. ROPOS WIRD ENT-
                0790           ;SPRECHEND WEITERGEZAEHLT.
                0800           ;WENN ROPOS=ROSOL WIRD DER MOTOR AB-
                0810           ;GESTELLT,DIE IMPULSE JEDOCH WEITER-
                0820           ;GEZAEHLT.WENN ALLE ACHSEN ZUR RUHE
                0830           ;GEKOMMEN SIND,WIRD IN BASIC ZURUECK-
                0840           ;GEGEBEN.
                0850           ;*****************************
CEE7- A2 06     0860 ROBOT     LDX #6              ;SCHLEIFENZAEHLER
CEE9- 38        0870 MDIR      SEC                 ;SOLL-/ISTWERTVERGLEICH
CEEA- BD D0 CF  0880           LDA ROPOSL,X        ;ISTWERT
CEED- FD D8 CF  0890           SBC ROSOLL,X        ;SOLLWERT
CEF0- 8D F0 CF  0900           STA SCRATCL         ;ZWISCHENSPEICHERN
CEF3- BD D1 CF  0910           LDA ROPOSH,X        ;DTO.HIGHBYTE
CEF6- FD D9 CF  0920           SBC ROSOLH,X
CEF9- 8D F1 CF  0930           STA SCRATCH
CEFC- 10 05     0940           BPL PLUS
CEFE- BD C9 CF  0950           LDA ML,X            ;MOTORENDREHRICHTUNG
CF01- D0 0B     0960           BNE NPOS            ;BRANCH ALWAYS
CF03- AD F0 CF  0970 PLUS      LDA SCRATCL         ;UNTERSUCHE AUF NULL
CF06- 0D F1 CF  0980           ORA SCRATCH
CF09- F0 03     0990           BEQ NPOS
CF0B- BD C8 CF  1000           LDA MR,X            ;MOTORDREHRICHTUNG
CF0E- 9D E0 CF  1010 NPOS      STA AV,X            ;ABSPEICHERN
CF11- 9D E8 CF  1020           STA MD,X            ;MOTORRICHTUNG ABSPEICHERN
CF14- CA        1030           DEX                 ;SCHLEIFENZAEHLER
CF15- CA        1040           DEX
CF16- 10 D1     1050           BPL MDIR            ;SCHLEIFENENDE
CF18- 20 3D CE  1060           JSR SHIN            ;DIGITALEINGABE
CF1B- 8D F1 CF  1070           STA SCRATCH         ;PEGELANFANGSWERTE ABSP.
                1080           ;*****************************
                1090           ;EINLESEN DIGITALEINGAENGE
                1100           ;*****************************
CF1E- 20 3D CE  1110 DIGIN     JSR SHIN            ;DIGITAL E.EINLESEN
CF21- A8        1120           TAY
CF22- 4D F1 CF  1130           EOR SCRATCH         ;DETEKTIERE FLANKEN
CF25- 8D F0 CF  1140           STA SCRATCL         ;SEKTORFLANKEN ZWISCHENSP.
CF28- 8C F1 CF  1150           STY SCRATCH         ;PEGEL ZWISCHENSPEICHERN
CF2B- A9 00     1160           LDA #0
CF2D- 8D B0 CD  1170           STA AVAR            ;AVAR VORBEREITEN
CF30- A2 06     1180           LDX #6              ;SCHLEIFE UEBER MOTOREN
                1190           ;*************************
                1200           ;SCHLEIFE UEBER ALLE MOTOREN
                1210           ;*************************
CF32- AD F1 CF  1220 LOOPHEAD  LDA SCRATCH         ;DIGITALEINGABE
                1230           ;*************************
                1240           ;ENDTASTER TESTEN
                1250           ;*************************
CF35- 3D C9 CF  1260           AND ML,X            ;ET HERAUSMASKIEREN
CF38- D0 06     1270           BNE SEKTOR          ;ET NICHT AKTIV
CF3A- 9D E0 CF  1280           STA AV,X            ;MOTOR AUS
CF3D- 9D E1 CF  1290           STA NL,X            ;NACHLAUF AUF 0
                1300           ;*************************
                1310           ;SEKTORENFLANKEN TESTEN
                1320           ;*************************
CF40- AD F0 CF  1330 SEKTOR    LDA SCRATCL         ;DIGITALEINGABE
CF43- 3D C8 CF  1340           AND MR,X            ;SEKTOR HERAUSMASKIEREN
CF46- F0 4C     1350           BEQ NEXCOMP         ;SEKTOR KEINE FLANKE
CF48- BD E8 CF  1360           LDA MD,X
CF4B- DD C8 CF  1370           CMP MR,X            ;LINKSLAUF?
CF4E- D0 14     1380           BNE INCPOS
CF50- 38        1390           SEC
CF51- BD D0 CF  1400           LDA ROPOSL,X        ;ROBOTER POSITION -1
CF54- E9 01     1410           SBC #1
CF56- 9D D0 CF  1420           STA ROPOSL,X
CF59- BD D1 CF  1430           LDA ROPOSH,X
CF5C- E9 00     1440           SBC #0
```

```
CF5E- 9D D1 CF  1450            STA ROPOSH,X
CF61- 38        1460            SEC
CF62- B0 11     1470            BCS NEXSK           ;BRANCH ALWAYS
CF64- 18        1480 INCPOS     CLC
CF65- BD D0 CF  1490            LDA ROPOSL,X        ;ROBOTER POSITION +1
CF68- 69 01     1500            ADC #1
CF6A- 9D D0 CF  1510            STA ROPOSL,X
CF6D- BD D1 CF  1520            LDA ROPOSH,X
CF70- 69 00     1530            ADC #0
CF72- 9D D1 CF  1540            STA ROPOSH,X
CF75- BD D0 CF  1550 NEXSK      LDA ROPOSL,X        ;IST-/SOLLWERT VERGLEICHEN
CF78- DD D8 CF  1560            CMP ROSOLL,X
CF7B- D0 17     1570            BNE NEXCOMP
CF7D- BD D1 CF  1580            LDA ROPOSH,X
CF80- DD D9 CF  1590            CMP ROSOLH,X
CF83- D0 0F     1600            BNE NEXCOMP
CF85- A9 00     1610            LDA #0              ;MOTOR ABSCHALTEN
CF87- DD E0 CF  1620            CMP AV,X            ;SOLL = IST DAS ERSTE MAL?
CF8A- F0 08     1630            BEQ NEXCOMP
CF8C- 9D E0 CF  1640            STA AV,X            ;MOTOR AUS
CF8F- A9 FF     1650            LDA #$FF
CF91- 9D E1 CF  1660            STA NL,X            ;NACHLAUFZAEHLER SETZEN
CF94- A9 00     1670 NEXCOMP    LDA #0              ;NACHLAUFZAEHLER TESTEN
CF96- DD E1 CF  1680            CMP NL,X
CF99- F0 03     1690            BEQ OUT             ;NICHT DEKREMENTIEREN, WEN
CF9B- DE E1 CF  1700            DEC NL,X            ;NACHLAUFZ. DEKR.
CF9E- AD B0 CD  1710 OUT        LDA AVAR            ;AVAR ZUSAMMENSETZEN
CFA1- 1D E0 CF  1720            ORA AV,X            ;AUS AV DER EINZELNEN MOT.
CFA4- 8D B0 CD  1730            STA AVAR            ;ZURUECKSPEICHERN
CFA7- CA        1740            DEX                 ;SCHLEIFENZAEHLER DEKR.
CFA8- CA        1750            DEX
CFA9- 10 87     1760            BPL LOOPHEAD        ;SCHLEIFENENDE
CFAB- AD B0 CD  1770            LDA AVAR            ;AUSGABE ANS INTERFACE
CFAE- 20 F1 CD  1780            JSR SHOUT
CFB1- F0 03     1790            BEQ NLTST           ;ALLE MOTOREN AUS
CFB3- 4C 1E CF  1800            JMP DIGIN
CFB6- A2 06     1810 NLTST      LDX #$06            ;ALLE NACHLAUFZAEHLER
CFB8- 1D E1 CF  1820 TSTNL      ORA NL,X            ;NOCH TESTEN
CFBB- CA        1830            DEX                 ;SCHLEIFENZAEHLER
CFBC- CA        1840            DEX
CFBD- 10 F9     1850            BPL TSTNL
CFBF- C9 00     1860            CMP #$00            ;ALLE ABGELAUFEN?
CFC1- F0 03     1870            BEQ END             ;SCHON ALLE FERTIG
CFC3- 4C 1E CF  1880            JMP DIGIN
CFC6- 58        1890 END        CLI                 ;INTERRUPT FREIGEBEN
CFC7- 60        1900            RTS                 ;ZURUECK INS BASIC
CFC8- 02        1910 MR         .BY %00000010       ;MOTOR1 RECHTS
CFC9- 01        1920 ML         .BY %00000001       ;MOTOR1 LINKS
CFCA- 08        1930            .BY %00001000       ;MOTOR2 RECHTS
CFCB- 04        1940            .BY %00000100       ;MOTOR2 LINKS
CFCC- 20        1950            .BY %00100000       ;MOTOR3 RECHTS
CFCD- 10        1960            .BY %00010000       ;MOTOR3 LINKS
CFCE- 80        1970            .BY %10000000       ;MOTOR4 RECHTS
CFCF- 40        1980            .BY %01000000       ;MOTOR4 LINKS
CFD0- 00        1990 ROPOSL     .BY 0
CFD1- 00        2000 ROPOSH     .BY 0
CFD2-           2010            .DS 6
CFD8- 00        2020 ROSOLL     .BY 0
CFD9- 00        2030 ROSOLH     .BY 0
CFDA-           2040            .DS 6
CFE0- 00        2050 AV         .BY 0               ;AUSGABEVARIABLE
CFE1- 00        2060 NL         .BY 0               ;NACHLAUFZAEHLER
CFE2-           2070            .DS 6               ;VERZAHNTE TABELLE
CFE8- 00        2080 MD         .BY 0               ;MOTORRICHTUNG
CFE9- 00        2090            .BY 0
CFEA-           2100            .DS 6
CFF0- 00        2110 SCRATCL    .BY 0
CFF1- 00        2120 SCRATCH    .BY 0
                2130            .EN
```

--- LABEL FILE: ---

```
AV =CFE0            AVAR =CDB0          AYFAC =B391
CKCOM =AEFD         DIGIN =CF1E         END =CFC6
F16INT =B7F7        GETWORD =AD8A       INCPOS =CF64
LOOPHEAD =CF32      LOPOS =CEB0         MASK =CDB1
MD =CFE8            MDIR =CEE9          ML =CFC9
MR =CFC8            NEXCOMP =CF94       NEXSK =CF75
NL =CFE1            NLTST =CFB6         NPOS =CF0E
OUT =CF9E           P1 =CEC2            P2 =CEC6
P3 =CECA            P4 =CECE            PLUS =CF03
ROBOT =CEE7         ROPOS =CE98         ROPOSH =CFD1
ROPOSL =CFD0        ROSOLH =CFD9        ROSOLL =CFD8
SCRATCH =CFF1       SCRATCL =CFF0       SEKTOR =CF40
SHIN =CE3D          SHOUT =CDF1         STOPOS =CED0
SYNTAX =CEB8        TSTNL =CFB8
//0000,CFF2,CFF2
!
```

Prog. ROBOT.SYS (Z80)

```
Pass 1 errors: 00

                        ;Pogramm Schneider CPC464 Interface
                        ;Copyright (C) Artur Fischer Forschung
                        ;Version 2 inklusive Robotersteuerung
                        ;File ROSYS.GEN
                        ;August 1985
                        ;
                        ;Aufruf des fischertechnik Interface
                        ;vom CPC durch Kommandos:
                        ;CALL m1,ein        CALL m1,aus
                        ;CALL m1,links      CALL m1,rechts
                        ;CALL in,@e1        CALL in,@ex
                        ;Statt m1 kann auch m2, m3 und m4 benutzt werden.
                        ;Statt e1 kann auch e2, e3 bis e8 benutzt werden.
                        ;Statt ex kann auch ey benutzt werden.
                        ;
                        ;Spezielle Roboter-Befehle:
                        ;CALL p1,nnnn     Sollposition ablegen
                        ;CALL in,@i1      Istposition abfragen
                        ;Statt p1 kann auch p2, p3 und p4 benutzt werden.
                        ;Statt i1 kann auch i2, i3 und i4 benutzt werden.
                        ;CALL robot       Start des Roboters
                        ;*********************************
A400                    org  #a400              ;Programmstart
                        ;*********************************
A400  CDFEA4    INIT:   CALL SYNT0              ;Syntax-Pruefung
A403  212CA6            LD   HL,ROPOSL          ;Tabellenzeiger
A406  AF                XOR  A                  ;Akku loeschen
A407  0617              LD   B,#17              ;Schleifenzaehler
A409  77        LOOP1:  LD   (HL),A             ;Robotertabellen loeschen
A40A  23                INC  HL
A40B  10FC              DJNZ LOOP1              ;Schleifenende
A40D  3E00              LD   A,#00              ;Ausgabevariable loeschen
A40F  181D              JR   STVAR
A411  0603      M1:     LD   B,#03              ;Motor 1
A413  180A              JR   BOUT
A415  060C      M2:     LD   B,#0C              ;Motor 2
A417  1806              JR   BOUT
A419  0630      M3:     LD   B,#30              ;Motor 3
A41B  1802              JR   BOUT
A41D  06C0      M4:     LD   B,#C0              ;Motor 4
A41F  CD04A5    BOUT:   CALL SYNT1              ;Syntax-Pruefung
                        ;*********************************
                        ;Einzelbit Ausgabe
                        ;*********************************
A422  3A0EA5            LD   A,(AVAR)           ;Ausgabevariable
A425  B0                OR   B                  ;setze Bits
A426  4F                LD   C,A
A427  DD7E00            LD   A,(IX+0)
A42A  A0                AND  B
A42B  47                LD   B,A
A42C  79                LD   A,C
A42D  A8                XOR  B
A42E  320EA5    STVAR:  LD   (AVAR),A           ;setze Ausgabevariable
A431  CD36A4            CALL SHOUT              ;Ausgabe an Interface
A434  FB                EI                      ;Interrupt freigeben
A435  C9                RET                     ;Ruecksprung in BASIC

                        ;*********************************
                        ;Routine zur Interface-Steuerung
                        ;Ausgabe
                        ;Ausgabemuster wird im Akku uebergeben
                        ;benutzt A, BC, DE.
                        ;*********************************
A436  0100EF    SHOUT:  LD   BC,#EF00           ;Zeiger in Drucker Port
A439  4F                LD   C,A                ;c ist Arbeitsregister
A43A  1E08              LD   E,#08              ;Schleifenzaehler
A43C  1630      LOOP:   LD   D,#30              ;Ruhepegel Interface
A43E  79                LD   A,C
A43F  07                RLCA                    ;naechstes Bit
A440  4F                LD   C,A
A441  3002              JR   NC,NULL            ;=0?
A443  1634              LD   D,#34              ;setze DATA OUT
A445  ED51      NULL:   OUT  (C),D              ;Ausgabe
A447  7A                LD   A,D
A448  F608              OR   #08                ;setze CLOCK
A44A  ED79              OUT  (C),A              ;Ausgabe
A44C  1D                DEC  E                  ;Schleifenzaehler
A44D  20ED              JR   NZ,LOOP            ;Schleifenende
A44F  1639              LD   D,#39              ;setze LOAD OUT
A451  ED51              OUT  (C),D              ;Ausgabe
A453  C9                RET
                        ;*********************************
                        ;Eingaberoutine
                        ;Kommando in,@en
                        ;*********************************
A454  CD04A5    inp:    CALL SYNT1              ;Syntax-Pruefung
A457  2185AE            LD   HL,#AE85           ;Variablen-Speicher
A45A  4E                LD   C,(HL)             ;Zeiger in Variablen-Sp.
A45B  23                INC  HL
A45C  46                LD   B,(HL)
A45D  210500            LD   HL,#0005
A460  09                ADD  HL,BC              ;Adresse von E1
A461  DD5601            LD   D,(IX+1)
A464  DD5E00            LD   E,(IX+0)
A467  010100            LD   BC,#0001           ;Bit-Zaehler
A46A  7C        VARTST: LD   A,H                ;vergleiche Adressen
A46B  BA                CP   D                  ;high-Byte
A46C  2004              JR   NZ,NEXTVAR
A46E  7D                LD   A,L
A46F  BB                CP   E                  ;low-Byte
A470  2813              JR   Z,VARFOUND         ;Variable gefunden
A472  C5        NEXTVA: PUSH BC
A473  010700            LD   BC,#0007           ;inkrementiere Adresse
A476  09                ADD  HL,BC
A477  C1                POP  BC
A478  79                LD   A,C
A479  17                RLA                     ;Bit-Zaehler hochschieben
A47A  4F                LD   C,A
A47B  78                LD   A,B                ;high-Byte
A47C  17                RLA
A47D  47                LD   B,A
A47E  FE40              CP   #40                ;Variablentabelle zu Ende
A480  CA08A5            JP   Z,SYNTAX           ;Syntax-Fehler
A483  18E5              JR   VARTST             ;weiter suchen
A485  78        VARFOU: LD   A,B                ;Analogabfrage?
```

```
A486  FE04      1150          CP   #04
A488  D20FA5    1160          JP   NC,ROPOS
A48B  FE01      1170          CP   #01
A48D  2844      1180          JR   Z,XPOTI              ;Eingang EX
A48F  FE02      1190          CP   #02
A491  2844      1200          JR   Z,YPOTI              ;Eingang EY
A493  C5        1210          PUSH BC                   ;rette BC
A494  CD99A4    1220          CALL SHIN                 ;Digitaleingabe
A497  182F      1230          JR   CONT
                1240 ;*********************************
                1250 ;Routine zur Interface-Steuerung
                1260 ;Eingabe
                1270 ;benutzt A, BC und DE.
                1280 ;Eingabe wird im Akku uebergeben
                1290 ;*********************************
A499  1632      1300 SHIN:    LD   D,#32                ;setze LOAD IN
A49B  0100EF    1310          LD   BC,#EF00             ;Zeiger in Drucker-Port
A49E  ED51      1320          OUT  (C),D                ;Ausgabe
A4A0  163A      1330          LD   D,#3A                ;setze CLOCK
A4A2  ED51      1340          OUT  (C),D                ;Ausgabe
A4A4  1E08      1350          LD   E,#08                ;Schleifenzaehler
A4A6  17        1360 LOOP2:   RLA                       ;Datenwort hochschieben
A4A7  E6FE      1370          AND  #FE                  ;Bit 0 loeschen
A4A9  0100F5    1380          LD   BC,#F500             ;Zeiger auf BUSY-Input
A4AC  4F        1390          LD   C,A
A4AD  ED78      1400          IN   A,(C)                ;einlesen
A4AF  E640      1410          AND  #40                  ;Busy-Leitung maskieren
A4B1  17        1420          RLA                       ;und zum Testen in Carry
A4B2  17        1430          RLA
A4B3  79        1440          LD   A,C
A4B4  3002      1450          JR   NC,NULL2             ;teste DATA-IN
A4B6  F601      1460          OR   #01                  ;setze Bit 0
A4B8  0100EF    1470 NULL2:   LD   BC,#EF00             ;Zeiger in Drucker-Port
A4BB  1630      1480          LD   D,#30                ;loesche CLOCK
A4BD  ED51      1490          OUT  (C),D                ;Ausgabe
A4BF  1638      1500          LD   D,#38                ;setze CLOCK
A4C1  ED51      1510          OUT  (C),D                ;Ausgabe
A4C3  1D        1520          DEC  E                    ;Schleifenzaehler
A4C4  20E0      1530          JR   NZ,LOOP2             ;Schleifenende
A4C6  2F        1540          CPL                       ;negative Logik!
A4C7  C9        1550          RET
                1560 ;*********************************
A4C8  C1        1570 CONT:    POP  BC                   ;BC restaurieren
A4C9  A1        1580          AND  C
A4CA  2802      1590          JR   Z,BASRET
A4CC  3E01      1600          LD   A,#01                ;<>0 -> 1
A4CE  77        1610 BASRET:  LD   (HL),A               ;in Variablentabelle
A4CF  23        1620          INC  HL
A4D0  70        1630          LD   (HL),B
A4D1  FB        1640          EI                        ;Interrupt freigeben
A4D2  C9        1650          RET                       ;zurueck in BASIC
                1660 ;*********************************
                1670 ;Analogeingabe
                1680 ;hierher wird verzweigt, wenn ex oder ey
                1690 ;abgefragt werden soll.
                1700 ;*********************************
A4D3  16A0      1710 XPOTI:   LD   D,#A0                ;setze TRIGGER-X
A4D5  1802      1720          JR   POTI

A4D7  1690      1730 YPOTI:   LD   D,#90                ;setze TRIGGER-Y
A4D9  0100EF    1740 POTI:    LD   BC,#EF00             ;Zeiger in Drucker-Port
A4DC  ED51      1750          OUT  (C),D                ;Ausgabe
A4DE  1638      1760          LD   D,#38                ;setze CLOCK
A4E0  ED51      1770          OUT  (C),D                ;Ausgabe
A4E2  0100F5    1780          LD   BC,#F500             ;Zeiger auf BUSY-Input
A4E5  110000    1790          LD   DE,#0000             ;Zaehler auf Null
A4E8  ED78      1800 LOOP3:   IN   A,(C)                ;lies COUNT IN
A4EA  17        1810          RLA                       ;in Carry schieben
A4EB  17        1820          RLA                       ;zum Testen
A4EC  3804      1830          JR   C,STOP               ;Puls zu Ende?
A4EE  1C        1840          INC  E                    ;Zaehler erhoehen
A4EF  20F7      1850          JR   NZ,LOOP3             ;weiterzaehlen
A4F1  1D        1860          DEC  E                    ;Ueberlauf -> 255
A4F2  73        1870 STOP:    LD   (HL),E               ;in Variable ablegen
A4F3  23        1880          INC  HL
A4F4  72        1890          LD   (HL),D
A4F5  0100EF    1900          LD   BC,#EF00
A4F8  1638      1910          LD   D,#38                ;setze CLOCK
A4FA  ED51      1920          OUT  (C),D                ;Ausgabe
A4FC  FB        1930          EI                        ;Interupt freigeben
A4FD  C9        1940          RET                       ;zurueck in BASIC
                1950 ;*********************************
                1960 ;Syntax Pruefroutine
                1970 ;*********************************
A4FE  FE00      1980 SYNTO:   CP   #00                  ;CALL INIT, ROBOT 0 Arg.
A500  F3        1990          DI                        ;Interrupt sperren
A501  C8        2000          RET  Z
A502  1804      2010          JR   SYNTAX               ;Fehlermeldung
A504  FE01      2020 SYNT1:   CP   #01                  ;CALL Mn,Richt. ein Arg.
A506  F3        2030          DI                        ;Interrupt sperren
A507  C8        2040          RET  Z
A508  CD00B9    2050 SYNTAX:  CALL #B900                ;ROM einschalten
A50B  C3C6DD    2060          JP   #DDC6                ;Fehlermeldung drucken
A50E  00        2070 AVAR:    DEFB #00                  ;Ausgabewort
                2080 ;*********************************
                2090 ;Roboterroutinen
                2100 ;
                2110 ;Copyright (C) Artur Fischer Forschung
                2120 ;August 1985
                2130 ;
                2140 ;Abfrage Istwert der Roboterposition
                2150 ;*********************************
A50F  EB        2160 ROPOS:   EX   DE,HL
A510  212CA6    2170          LD   HL,ROPOSL            ;Zeiger Sollwerte
A513  1F        2180          RRA
A514  1F        2190          RRA
A515  1F        2200 LOOP4:   RRA
A516  3804      2210          JR   C,CONT1
A518  23        2220          INC  HL                   ;Zeiger weiterschalten
A519  23        2230          INC  HL
A51A  18F9      2240          JR   LOOP4
A51C  7E        2250 CONT1:   LD   A,(HL)               ;lade low-Byte
A51D  12        2260          LD   (DE),A               ;in Variable ablegen
A51E  23        2270          INC  HL                   ;high-Byte
A51F  13        2280          INC  DE
A520  7E        2290          LD   A,(HL)
A521  12        2300          LD   (DE),A
```

```
A522  FB        2310          EI                        ;Interrupt freigeben
A523  C9        2320          RET                       ;zurueck in BASIC
                2330 ;*********************************
                2340 ;Roboterroutine
                2350 ;
                2360 ;Setze Sollwerte der Roboterposition
                2370 ;*********************************
A524  0E00      2390 P1:      LD   C,#00                ;Zeiger=0
A526  180A      2390          JR   STOPOS
A528  0E02      2400 P2:      LD   C,#02                ;Zeiger=2
A52A  1806      2410          JR   STOPOS
A52C  0E04      2420 P3:      LD   C,#04                ;Zeiger=4
A52E  1802      2430          JR   STOPOS
A530  0E06      2440 P4:      LD   C,#06                ;Zeiger=6
A532  0600      2450 STOPOS:  LD   B,#00
A534  CD04A5    2460          CALL SYNT1                ;Syntax-Pruefung
A537  2134A6    2470          LD   HL,ROSOLL            ;Sollwert-Tabelle
A53A  09        2480          ADD  HL,BC                ;Zeiger addieren
A53B  DD7E00    2490          LD   A,(IX)               ;Argument holen
A53E  77        2500          LD   (HL),A               ;abspeichern
A53F  23        2510          INC  HL                   ;high-Byte
A540  DD7E01    2520          LD   A,(IX+#01)
A543  77        2530          LD   (HL),A
A544  FB        2540          EI                        ;Interrupt freigeben
A545  C9        2550          RET                       ;zurueck in BASIC
                2560 ;*********************************
                2570 ;Robotersteuerung
                2580 ;
                2590 ;Die Routine vergleicht die in ROSOL
                2600 ;abgespeicherten Werte mit jenen in
                2610 ;ROPOS. Aus der Differenz wird die
                2620 ;Motordrehrichtung bestimmt.
                2630 ;Die Motoren mit Differenz <>0 werden
                2640 ;gestartet und die Impulse der Licht-
                2650 ;schranke gezaehlt. ROPOS wird
                2660 ;entsprechend weitergezaehlt.
                2670 ;Wenn ROPOS=ROSOL wird der Motor ab-
                2680 ;gestellt,die Impulse jedoch weiter-
                2690 ;gezaehlt.Wenn alle Achsen zur Ruhe
                2700 ;gekommen sind,wird in Basic zurueck-
                2710 ;gegeben.
                2720 ;*********************************
A546  0603      2730 ROBOT:   LD   B,#03                ;Schleifenzaehler
A548  CDFEA4    2740          CALL SYNTO                ;Syntax-Pruefung
A54B  DD2124A6  2750          LD   IX,MR
A54F  DD6E08    2760 MDIR:    LD   L,(IX+#08)           ;ROPOS (Istwert)
A552  DD6609    2770          LD   H,(IX+#09)
A555  DD5E10    2780          LD   E,(IX+#10)           ;ROSOL (Sollwert)
A558  DD5611    2790          LD   D,(IX+#11)
A55B  A7        2800          AND  A                    ;Carry loeschen
A55C  ED52      2810          SBC  HL,DE                ;Differenz
A55E  DD7528    2820          LD   (IX+#28),L           ;abspeichern (SCRATCH)
A561  DD7429    2830          LD   (IX+#29),H
A564  3005      2840          JR   NC,PLUS              ;Differenz>0
A566  DD7E01    2850          LD   A,(IX+#01)           ;Linkslauf
A569  180B      2860          JR   NPOS
A56B  DD7E28    2870 PLUS:    LD   A,(IX+#28)           ;Differenz=0?
A56E  DDB629    2880          OR   (IX+#29)
A571  2803      2890          JR   Z,NPOS
A573  DD7E00    2900          LD   A,(IX+#00)           ;Rechtslauf
A576  DD7718    2910 NPOS:    LD   (IX+#18),A           ;Teil-Ausgabewort (AV)
A579  DD7720    2920          LD   (IX+#20),A           ;Drehrichtung abspeichern
A57C  DD23      2930          INC  IX                   ;Zeiger weiter
A57E  DD23      2940          INC  IX
A580  10CD      2950          DJNZ MDIR                 ;Schleifenende
A582  DD2124A6  2960          LD   IX,MR                ;Zeiger Drehrichtungen
A586  214DA6    2970          LD   HL,SCRATC            ;Zeiger auf SCRATCH
A589  CD99A4    2980          CALL SHIN                 ;Digital-Eingabe
A58C  77        2990          LD   (HL),A               ;Anfangswerte abspeichern
                3000 ;*********************************
                3010 ;Einlesen der Digitaleingaenge
                3020 ;*********************************
A58D  CD99A4    3030 DIGIN:   CALL SHIN                 ;Digital-Eingabe
A590  4E        3040          LD   C,(HL)               ;Anfangswerte
A591  77        3050          LD   (HL),A               ;neues Bitmuster
A592  A9        3060          XOR  C                    ;erkenne Flanken
A593  2B        3070          DEC  HL
A594  77        3080          LD   (HL),A
A595  23        3090          INC  HL
A596  AF        3100          XOR  A                    ;Akku loeschen
A597  320EA5    3110          LD   (AVAR),A             ;Ausgabewort loeschen
A59A  0603      3120          LD   B,#03                ;Schleifenzaehler
A59C  DD2124A6  3130          LD   IX,MR                ;Zeiger Drehrichtung
                3140 ;*********************************
                3150 ;Schleife ueber alle Motoren
                3160 ;*********************************
A5A0  7E        3170 LOOPH:   LD   A,(HL)               ;Schleifenkopf
                3180 ;*********************************
                3190 ;Endtaster testen
                3200 ;*********************************
A5A1  DDA601    3210          AND  (IX+#01)             ;Endtaster maskieren
A5A4  2006      3220          JR   NZ,SECTOR            ;betaetigt?
A5A6  DD7718    3230          LD   (IX+#18),A           ;Motor aus
A5A9  DD7719    3240          LD   (IX+#19),A           ;Nachlauf=0
                3250 ;*********************************
                3260 ;Sektorflanken testen
                3270 ;*********************************
A5AC  2B        3280 SECTOR:  DEC  HL
A5AD  7E        3290          LD   A,(HL)               ;Sektor maskieren
A5AE  23        3300          INC  HL
A5AF  DDA600    3310          AND  (IX+0)
A5B2  2831      3320          JR   Z,NEXCOM             ;keine Flanke
A5B4  DD5E08    3330          LD   E,(IX+#08)           ;Istwert (ROPOSL)
A5B7  DD5609    3340          LD   D,(IX+#09)           ;Istwert (ROPOSH)
A5BA  DD7E20    3350          LD   A,(IX+#20)           ;Linkslauf?
A5BD  DDBE01    3360          CP   (IX+#01)
A5C0  2803      3370          JR   Z,INCPOS
A5C2  1B        3380          DEC  DE                   ;Roboter Position -1
A5C3  1801      3390          JR   NEXSK
A5C5  13        3400 INCPOS:  INC  DE                   ;Roboter Position +1
A5C6  DD7308    3410 NEXSK:   LD   (IX+#08),E           ;Roboter Position absp.
A5C9  DD7209    3420          LD   (IX+#09),D
A5CC  7B        3430          LD   A,E                  ;Ist- und Sollwert vergl
A5CD  DDBE10    3440          CP   (IX+#10)
A5D0  2013      3450          JR   NZ,NEXCOM
A5D2  7A        3460          LD   A,D
```

```
A5D3  DDBE11    3470          CP   (IX+#11)
A5D6  200D      3480          JR   NZ,NEXCOM
A5D8  AF        3490          XOR  A                    ;Akku loeschen
A5D9  DDBE18    3500          CP   (IX+#18)             ;Soll=Ist das 1. Mal?
A5DC  2807      3510          JR   Z,NEXCOM
A5DE  DD7718    3520          LD   (IX+#18),A           ;Motor ausschalten
A5E1  3D        3530          DEC  A                    ;Akku=FF
A5E2  DD7719    3540          LD   (IX+#19),A           ;Nachlaufzaehler setzen
A5E5  AF        3550 NEXCOM:  XOR  A                    ;Akku loeschen
A5E6  DDBE19    3560          CP   (IX+#19)             ;Nachlaufzaehler testen
A5E9  2803      3570          JR   Z,OUT                ;Nicht dekr. wenn 0
A5EB  DD3519    3580          DEC  (IX+#19)             ;Nachlaufzaehler dekr.
A5EE  3A0EA5    3590 OUT:     LD   A,(AVAR)             ;Ausgabewort aus AV
A5F1  DDB618    3600          OR   (IX+#18)             ;der einzelnen Motoren
A5F4  320EA5    3610          LD   (AVAR),A             ;abspeichern
A5F7  DD23      3620          INC  IX                   ;Tabellenzeiger 2 Bytes
A5F9  DD23      3630          INC  IX                   ;weiter
A5FB  10A3      3640          DJNZ LOOPH                ;Schleifenende
A5FD  DD2124A6  3650          LD   IX,MR                ;Zeiger zuruecksetzen
A601  3A0EA5    3660          LD   A,(AVAR)             ;Ausgabewort
A604  F5        3670          PUSH AF                   ;Register retten
A605  CD36A4    3680          CALL SHOUT                ;Ausgabe ans Interface
A608  F1        3690          POP  AF                   ;Register restaurieren
A609  FE00      3700          CP   #00                  ;alle Motoren aus?
A60B  2803      3710          JR   Z,NLTST
A60D  C38DA5    3720          JP   DIGIN                ;nein - weiter
A610  0603      3730 NLTST:   LD   B,3                  ;Schleifenzaehler
A612  DDB619    3740 TSTNL:   OR   (IX+#19)             ;teste Nachlaufzaehler
A615  DD23      3750          INC  IX                   ;Zeiger 2 Bytes weiter
A617  DD23      3760          INC  IX
A619  10F7      3770          DJNZ TSTNL                ;Schleifenende
A61B  FE00      3780          CP   #00                  ;alle abgelaufen?
A61D  2803      3790          JR   Z,END                ;ja - Ende
A61F  C38DA5    3800          JP   DIGIN                ;nein - weiter
A622  FB        3810 END:     EI                        ;Interrupt freigeben
A623  C9        3820          RET                       ;zurueck in BASIC
                3830 ;*********************************
A624  02        3840 MR:      DEFB %00000010            ;Motor1 rechts
A625  01        3850 ML:      DEFB %00000001            ;Motor1 links
A626  08        3860          DEFB %00001000            ;Motor2 rechts
A627  04        3870          DEFB %00000100            ;Motor2 links
A628  20        3880          DEFB %00100000            ;Motor3 rechts
A629  10        3890          DEFB %00010000            ;Motor3 links
A62A  80        3900          DEFB %10000000            ;Motor4 rechts
A62B  40        3910          DEFB %01000000            ;Motor5 links
A62C  00        3920 ROPOSL:  DEFB 0                    ;Roboter Istposition
A62D  00        3930 ROPOSH:  DEFB 0
A62E            3940          DEFS 6
A634  00        3950 ROSOLL:  DEFB 0                    ;Roboter Sollposition
A635  00        3960 ROSOLH:  DEFB 0
A636            3970          DEFS 6
A63C  00        3980 AV:      DEFB 0                    ;Teil-Ausgabeworte
A63D  00        3990 NL:      DEFB 0                    ;Nachlaufzaehler
A63E            4000          DEFS 6                    ;verzahnte Tabelle
A644  00        4010 MD:      DEFB 0                    ;Motor-Drehrichtungen
A645  00        4020          DEFB 0
A646            4030          DEFS 6
A64C  00        4040          DEFB 0
A64D  00        4050 SCRATC:  DEFB 0                    ;Scratch
A64E            4060          DEFS 6
                4070 ;************** Ende ***************

Pass 2 errors: 00

Table used:   702  from  1408
```

Table of Contents

Introduction	33
What is a Robot?	34
The Driving and Positioning System	35
Interface and Software	36
The Robot System Program	38
First Experiments	39
Assembly of the Robot	40
Control of the Robot	41
Teach-in Procedure	44
Further Experiments	45
Print out of the Programs	46
Operation of the Interface and the Robot System Program	51
Illustrated Assembly Instructions	61
Ribbon Cable Configuration	62
Circuit Layout Photo-Interrupter	63
Mechanical Assembly	64
Circuit Layout Training Robot	91

fischertechnik Training Robot

Dear friend of fischertechnik,

there is hardly any technical instrument which is as versatile as a computer. One of the most fascinating fields of computer techniques, however, is the control of technical models. With the fischertechnik computing kit TRAINING ROBOT you have acquired a model which will introduce you to the most sophisticated field of control techniques, i.e. robotics.
The kit offers the fulfilment of various requirements in one efficient instrument. First of all, the robot is to be realistic. For this reason, there is no likeness at all between our training robot and those robots known from science fiction, i.e. those metallic figures bearing a resemblance to man. Our constructive designs and concepts were based on today's industrial robots. You may use the kit for assembling a three axes industrial robot of rotary axes of motion. The exact meaning of all this will follow in this instruction manual.
Secondly, you should not merely be able to assemble the robot and to control it by means of your home or personal computer but you should also be given the chance to understand the individual process flows. Therefore, the programs have predominantly been written in BASIC and abundantly documented. The familiarization with the programs is also the prerequisite for developing your own ideas. I would not hesitate to maintain that this is the most essential benefit offered by the robot.
Modification, experimentation, extension, programming . . . your own creativeness will play the decisive part.
This third requirement regarding the robot is fulfilled in an ideal way and manner by the mechanical structure of the fischertechnik system. Irrespective, of whether you want to extend the functions of the grab, or to provide the robot with sensors, or to construct an environment for a computer-integrated manufacturing – the versatility and high precision of the fischertechnik components will assist you in every respect.
New components are contributing to the high functional reliability of the training robot. First of all, the driving and positioning system should be mentioned in this respect. New compact D.C. motors of high power make the robot fast in movement. However, the control of the robot alone is not all that matters. The program in the computer must be able to interrogate the position achieved. The photo-interrupters have been developed especially for this purpose. By infrared light they are transilluminating the wheel connected with the driving axle. During motion, the segmentation located there will generate pulses which via a fischertechnik interface are forwarded to the computer. In order to be able to evaluate this fast sequence of pulses a software package is used which has been developed especially for the training robot. This part has been written in the machine language of your computer and can, without any counting losses, monitor even the simultaneous movement of all robot joints. These programs are called in by BASIC and thus can be used without problems.
I am sure that the fischertechnik computing training robot will incite you to perform quite a series of experiments of your own and will enlarge considerably your knowledge and experience in this field.

Yours

What is a Robot

The aspiration to design a robot is almost as old as mankind. As often evidenced in the history of our culture, something like first robots appear already in ancient Greece. The temples were equipped with mechanical servants which were able to perform predetermined movements. They were controlled by the weight of the sand running down in a large hour-glass incorporated in the robot. In the peak-time of craftmanship in 18th and 19th century mechanically controlled puppets again and again caused the admiration of the contemporaries and still nowadays we can admire the skilfulness of their inventors. An automatic chessplayer, however, shows us the limits to creative faculty: a man small in stature was hidden in that "robot" skilfully controlling the chessplaying puppet via levers and ties. The word "robot" was created in our century only. In the stage-play R.U.R. (Rossum's Universal Robot) of the Czechoslovakian author Karel Capek the mechanical slaves were called "robots", derived from the Slavonian word "robota" defining "hard work". Our today's ideas of robots frequently stick to this imagination, i.e. we think of a machine of human appearance, which meanwhile is no longer controlled by an hour-glass or spring-mechanism but which has built in a computer as a "brain". Such ideas are substantiated by a lot of science-fiction literature.

On the other hand, we in practice are confronted with another type of robot which is ". . . a universally applicable moving automatic machinery of several axes the movement of which regarding sequence and paths resp. angles of motion is freely programmable and eventually sensor-controlled" (rules and regulations of VDI). Depending on their equipment e.g. with grippers, robots can pick up workpieces and transfer them to the next process step or they may perform process steps like spot-welding or coating by means of a spray gun but they may effect also a multitude of other manipulations and services. In spite of the technical terms used in the definition of an industrial robot these instruments exhibit to some extent human appearance. In particular the most universal robots take nature as a guide. Their motional apparatus frequently equals a human arm. There are four distinct components of the robot: the base plate corresponds to the body followed by the upper arm, the elbow-joint and the forearm. It ends in the gripping hand or grab. Here after, we shall use these terms in order to facilitate an orientation.
We should not overlook that also other types of industrial robots can be found. In any optional combination what so ever, the rotary axes of the robot may be substituted by a sliding along a guidance. If, in an extreme case, all motions are taking place along a line we obtain an appliance which corresponds more to a portal crane than to an arm. That's why such a robot is also called "portal robot" and in most cases it is better suited for high loads. Also with this type of robot, the sliding displacement is referred too as motional axis, although a rotary axle in the strictest acceptation of the word is not available.
It is common to all robots that for a positioning of the gripper always three motional axes are required. In case of a robot with an articulated arm this will be realized after an extensive experimentation only whereas in case of a portal robot this fact is realized immediately. It can move along the three dimensions: height, width and depth.

Perhaps, you will have heard already that industrial robots have got five or more axes. In rare cases, this larger number of axes improves the motional mechanism, e.g. for reaching into a corner of a motorcar body. Mostly the fourth and the fifth axis serves for the pivoting of the wrist. In this way and manner the direction of the gripper can be altered. These features are not included in the fischertechnik training robot as it includes only the three main axes of motion. This, however, is not to be regarded as a disadvantage as all studies regarding the geometry of a robot can also be performed at the three main motional axes. Moreover, the gripper has been equipped with a mechanical position compensation. Perhaps, it might be interesting for you to tackle the problem of providing your robot with the additional axes for the orientation of the gripper? But first of all the basic model should be created.

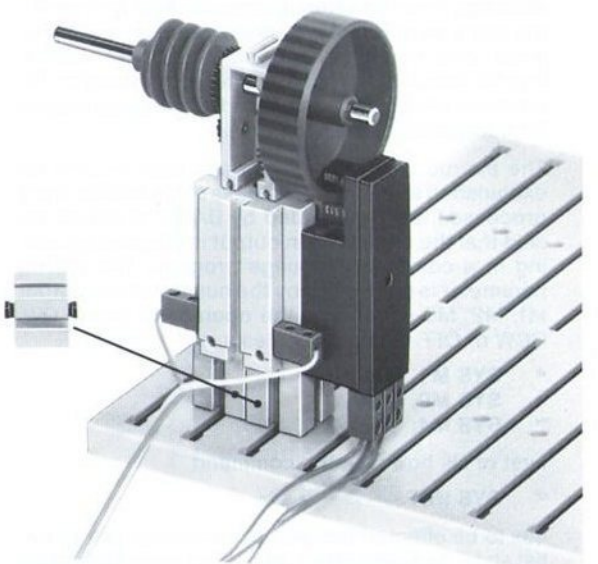
The Driving and Positioning System

D.C. motors serve for driving the robot. The three larger sized motors move the robot whereas the smaller sized motor effects the opening and closing of the gripper. D.C. motors feature the advantage that they are easily controlled and generate a high torque at low weight and volume. Thus they contribute to the quick motion of the robot. We can check the motors before installation by connecting them directly to the power unit. As in subsequent operation of the robot up to four motors can be operative simultaneously, a power unit of adequate capacity should be used. We recommend to use the fischertechnik computing power unit or any other power unit supplying 7 volts D.C., unregulated, at a load of 1.5 ampere.

With other types of motors, D.C. motors share a common disadvantage. It is true that they can move the robot via gear units but when the movement has been carried out, the positioning of the robot is known only approximately. Only the period during which the motor is operative, can be controlled via computer. Other factors like the exact mains voltage and consequently the voltage of the power unit, the easy-running of the robot and the change of loads at the gripper, are factors leading to an alteration of the motor speed. Consequently, the same periods of actuation must not always entail the same movements of the robot. Such a situation, however, is unbearable for a precision instrument like a robot. Therefore, the position of a robot must be measured again independent from the motor. The positioning system installed in the robot for this purpose consists of fork-type photo-interrupters. Fig. 1 shows the connection between such a fork-type photo-interrupter and motor/gear unit. The driving shaft of the gear holds a cup-shaped wheel on the circumference of which black lines have been printed at regular intervals. There are a total of 32 lines.
Now the transmission type photo-interrupter overlaps the rim of the cup. One side of the photo-interrupter is equipped with a light-emitting diode emitting infrared light. The other side of the photo-interrupter is provided with a photo transistor which is sensitive to infrared light. If no object is located in the photo-interrupter and the operation voltage of the photo-interrupter has been connected correctly (see below), a HIGH-signal will appear at the output defined by ⎍⎍. If, however, a non-transparent object is inserted between the prongs of the photo-interrupter, the light beam is interrupted. A LOW-signal will appear at the output. In exactly the same way and manner the photo-interrupter will react upon the different zones of the wheel. The black print will interrupt the light beam whereas the non-printed spaces will weaken the infrared light but will still transmit a sufficient quantity.
With a motor running, a sequence of pulses will appear at the output of the photo-interrupter which will correspond exactly to the dark and bright spots of the wheel. We are now able to precisely control the robot by means of such a device. We do not measure the period of running of the motor but we are counting the change of pulses at the output of the photo-interrupter. The resulting number is an exact measure for the rotation of the driving axle and consequently for the position of the robot. Naturally, the counting of the pulses requires a high operating speed of the computer. Such a task must be programmed directly in the machine language of the computer, as will be explained more fully in the documentation.

First of all, we should gather some experience with the positioning system. As mentioned above, the infrared beam must transilluminate the plastic material but not the black print. Sunlight and the light of neon tubes also contain a certain portion of infrared light which naturally will be interfering. We, therefore, have to look for the optimum adjustment of the photo-interrupter for operation. For this purpose, the sensitivity can be adjusted. By means of the attached screwdriver you may adjust a potentiometer through a hole in the upper side of the casing. You should do this with utmost care and sensitiveness in order not to damage the electronic components.
To assist you in adjustment, you will find the program ROBOT.ADJUSTMENT on the floppy disk resp. cassette. For the utilization of same you will need the fischertechnik interface so that, first of all, we should become familiar with this important instrument.

Fig. 1



Interface and Software

We should start with a brief remark regarding the documentation of the programs for fischertechnik computing. In the instruction manual the programs have been printed in the notation of Commodore 64. With the interface appropriate to your computer, a floppy disk or cassette is delivered which also includes the programs. The BASIC-notations of the various computers differ slightly. If you should not own a Commodore 64 but another type of computer, the program on the floppy disk or cassette will not exactly coincide with the program printed here. It has already been adapted to the respective type of computer. Those points in which deviations will occur at any rate, are identified by an asterisk at the beginning of the respective line in the print of the program. If you now want to compare the printed program with the program read in you must pay special attention when an asterisk appears. Also line numbers might be different in restricted ranges of the program code, essentially due to differences in controlling the layout of the screen. The instructions for the interface will give you further advice and assistance for adaptation of the programs.

The instructions for the interface also include an explanation as to how the signals of the interface are processed resp. generated by BASIC. It should be said that the control of an output is effected by calling in a computer language program. The calling parameter is composed by the number of the output M1, M2, M3 or M4 and the operational mode CW, CCW or OFF. Some examples are:

* **SYS M1, CW**
* **SYS M3, CCW**
* **SYS M4, OFF**

First of all, however, the command

* **SYS INIT**

has to be effected bringing the interface into an initial state. As a side effect, all motors are switched off simultaneously so that this command serves also for this purpose.
The inputs of the interface are operated by the USR-function. By the parameters E1, E2 through E8, the eight inputs are interrogated to which the mini-keys are connected. Other ON-OFF-signals may be put in here, too. The functions USR(EX) and USR(EY), however, serve for the input of gradually variable electrical values. These inputs are not required in case of the robot described here. They may become essential if you want to connect sensors, like e.g. photoresistors for recognition of objects.
It is also important to know that the interface incorporates a monitoring circuit for data transfer. If within half a second no new command – either output or input command – should be given, all motors will be switched off by this circuit. When stopping the computer program you thus do not need to switch off the power supply of the motors. If the transfer of data is initiated again, the interface will operate all motors again as before.

The machine language program effecting the data transfer between computer and interface, naturally must also be stored in the computer. The so-called driver routine serves for this purpose which is also included in the floppy disk resp. cassette. Simultaneously, it is a component of any other fischertechnik computing program occupying the line numbers 1 through 500. In the program lists of these instructions this part, however, will not appear as it will differ according to the type of computer concerned. The computer program must be adapted in all details to the structure of hard- and software of the computer. The driver routine is documented in the instructions for your interface.
What has been described so far, relates to the standard driver program as it comes on the floppy disc resp. cassette in the interface package.
Basing on the above described positioning system, the training robot makes use of an extended driver routine, the so-called robot system program. It is included on the floppy disk resp. cassette which comes along with the training robot as file ROBOT.SYSTEM. Further eight commands are added: the command

* **SYS P1, nnnn**

is quite similar to the output commands SYS M1, CW or SYS M4, OFF.
This command makes the motor M1 run until the robot has reached the position nnnn. In this, nnnn is a positive integer number representing the status of the above mentioned pulse counter. Thus, the robot system program always "knows" the prevailing status of the counter. If a new position is called in by the command, the robot system program will compute the difference between the positions. From the absolute numerical value it will take the number of pulses to be expected from the photo-interrupter, and from the sign the sense of rotation of the motor. But the motor will not yet move. As a programmer you now have the opportunity to call in the new positions for other motors, too. Only after having finalized the tasks for the robot system program, the command

* **SYS ROBOT**

is called in. Now all motors which are to obtain a new position, will start to run simultaneously. The pulse inputs of all operative motors are monitored and counted. Each motor will be switched off as soon as the new position has been reached. When all positions have been reached the command will return the control to the BASIC-program.

Some further remarks regarding the robot system program should be made in this context. A motor may already be switched off before reaching the position in case the limit key associated with the motor should have been triggered. On one hand, this limit key has been designed for stipulating a "home position" of the robot and on the other hand to avoid

inadmissible movements of the robot. If not actuated, these limit keys must be closed. With regard to the mini-keys attached to the model, consequently the contacts No. 1 and 2 will mostly have to be connected.
Moreover we had said that, when reaching the wanted position, the motor of each motional axis will be switched off. Those, however, who will have gone already into details regarding motor control, will surely know that after switch-off the motor will not come to an immediate stop. For this reason, the robot system program will monitor the pulse input still for a predetermined time after switch-off of the motor. Any pulse still arriving will be counted, too. Hence, the resulting position will not be exactly the position wanted but a position a little bit shifted. The position actually obtained can be interrogated by means of the function.

* **USR(P1) (idem for P2, P3 and P4).**

The position counters associated with the different motors may also be set to zero. This is effected by the command

* **SYS INIT**

which you know already from the driver routine. As within the scope of the robot system program it still has got this secondary effect it can no longer be used without hesitation for a simultaneous switch-off of the motors. On the other hand, however, this necessity will no longer occur as the motors are subjected to the reliable control of the robot system program.

Contrary to the simple commands of the driver routine, the commands of the robot system program include inputs and outputs interconnected with each other in order to achieve such a high performance. This, however, requires also a fixed allocation of the inputs and outputs of the interface as shown by the following table:

motor	limit key	pulse input	commands	
M1	E1	E2	SYS P1, nnnn	USR(P1)
M2	E3	E4	SYS P2, nnnn	USR(P2)
M3	E5	E6	SYS P3, nnnn	USR(P3)
M4	E7	E8	SYS P4, nnnn	USR(P4)

As to be seen from the table, the robot system commands are available for all four outputs of the interface even if the training robot will utilize the positioning system for three motors only. In this case, the motors 1 to 3 are controlled via the positioning commands. Motor 4 drives the gripper and is controlled by the simpler commands of the driver routine.
If your operations should not be based on fischertechnik computing interface but on other interface circuits, the information given here will not apply in all details. In any case the ideas outlined above can be adapted to any other hardware what so ever.

The Robot System Program

Below you will find the BASIC-program generating the robot system program for the Commodore 64 computer. This program as well as the programs printed from page 46 onwards can also be loaded from the fischertechnik floppy disk training robot/plotter/scanner. Disks resp. cassettes are available for other types of computers, too. However, as indicated above, those programs may differ from the listings due to the inherent differences of the computers' BASIC dialects.

```
1 PRINT CHR$(147):POKE 53280,3:POKE 53281,1
2 FOR I=1 TO 7
3 PRINT
4 NEXT
5 PRINT TAB(6);"PLEASE WAIT A MOMENT"
6 PRINT
7 PRINT TAB(6);"LOADING EXTENDED ROUTINES"
8 PRINT
9 PRINT TAB(6);"FOR THE THREE-AXES ROBOT"
10 REM ROBOT DRIVER FOR COMMODORE 64
13 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1985
15 REM COMMANDS OF THE DRIVER
20 REM SYS M1,OFF
25 REM SYS M1,CCW    SYS M1,CW
30 REM USR(E1)  USR(EX)  USR(EY)
35 REM SPECIAL ROBOT COMMANDS
40 REM SYS P1,NNNN  STORE POSITION
45 REM USR(P1) REQUEST POSITION
50 REM SYS ROBOT  START ROBOT
55 DATA 52656,0,0,169,0,162,23,157,208,53375
60 DATA 207,202,16,250,48,46,169,3,54316
65 DATA 208,10,169,12,208,6,169,48,55146
70 DATA 208,2,169,192,120,141,177,205,56360
75 DATA 32,253,174,173,176,205,13,177,57563
80 DATA 205,141,176,205,32,158,183,138,58801
85 DATA 45,177,205,141,177,205,173,176,60100
90 DATA 205,77,177,205,32,241,205,88,61330
95 DATA 96,141,176,205,72,169,63,141,62393
100 DATA 3,221,162,8,169,48,14,176,63194
105 DATA 205,144,2,9,4,141,1,221,63921
110 DATA 9,8,141,1,221,202,208,236,64947
115 DATA 169,57,141,1,221,104,141,176,65957
120 DATA 205,96,120,32,247,183,201,0,67041
125 DATA 208,118,192,162,240,57,192,146,68356
130 DATA 240,53,140,177,205,32,61,206,69470
135 DATA 45,177,205,168,240,2,160,1,70468
140 DATA 32,162,179,88,96,169,50,141,71385
145 DATA 1,221,9,8,141,1,221,162,72149
150 DATA 8,10,44,1,221,16,2,9,72460
155 DATA 1,160,48,140,1,221,160,56,73247
160 DATA 140,1,221,202,208,235,96,169 ,74519
165 DATA 255,141,4,221,141,5,221,169,75676
170 DATA 185,141,14,221,140,1,221,160,76759
175 DATA 58,140,1,221,173,4,221,162,77739
180 DATA 3,202,208,253,56,237,4,221,78923
185 DATA 208,242,162,56,142,1,221,56,80011
190 DATA 169,255,237,4,221,168,169,255,81489
195 DATA 237,5,221,32,145,179,88,96,82492
200 DATA 162,0,192,194,240,18,162,2,83462
205 DATA 192,198,240,12,162,4,192,202,84664
210 DATA 240,6,162,6,192,206,208,11,85695
215 DATA 188,208,207,189,209,207,32,145,87080
220 DATA 179,88,96,140,176,205,141,177,88282
225 DATA 205,96,169,0,240,10,169,2,89173
230 DATA 208,6,169,4,208,2,169,6,89945
235 DATA 141,177,205,32,253,174,32,138,91097
240 DATA 173,32,247,183,174,177,205,157,92445
245 DATA 217,207,152,157,216,207,96,162,93859
250 DATA 6,56,189,208,207,253,216,207,95201
255 DATA 141,240,207,189,209,207,253,217,96864
260 DATA 207,141,241,207,16,5,189,201,98071
265 DATA 207,208,11,173,240,207,13,241,99371
270 DATA 207,240,3,189,200,207,157,224,100798
275 DATA 207,157,232,207,202,202,16,209,102230
280 DATA 32,61,206,141,241,207,32,61,103211
285 DATA 206,168,77,241,207,141,240,207,104698
290 DATA 140,241,207,169,0,141,176,205,105977
295 DATA 162,6,173,241,207,61,201,207,107235
300 DATA 208,6,157,224,207,157,225,207,108626
305 DATA 173,240,207,61,200,207,240,76,110030
310 DATA 189,232,207,221,200,207,208,20,111514
315 DATA 56,189,208,207,233,1,157,208,112773
320 DATA 207,189,209,207,233,0,157,209,114184
325 DATA 207,56,176,17,24,189,208,207,115268
330 DATA 105,1,157,208,207,189,209,207,116551
335 DATA 105,0,157,209,207,189,208,207,117833
340 DATA 221,216,207,208,23,189,209,207,119313
345 DATA 221,217,207,208,15,169,0,221,120571
350 DATA 224,207,240,8,157,224,207,169,122007
355 DATA 255,157,225,207,169,0,221,225,123466
360 DATA 207,240,3,222,225,207,173,176,124919
365 DATA 205,29,224,207,141,176,205,202,126308
370 DATA 202,16,135,173,176,205,32,241,127488
375 DATA 205,240,3,76,30,207,162,6,128417
380 DATA 29,225,207,202,202,16,249,201,129748
385 DATA 0,240,3,76,30,207,88,96,130488
390 DATA 2,1,8,4,32,16,128,64,130743
395 DATA 1,2,4,8,16,32,64,128,130998
400 DATA 162,146,255,170,85,26,206,132048
405 DATA 52930,52934,52938,52942,52967,396759
410 READ INIT : M1=INIT
415 FOR M3=0 TO 67 : FOR M2=0 TO 7
420 READ M4 : POKE INIT+M3*8+M2,M4
425 M1=M1+M4 : NEXT
430 READ M4 : IF M1<>M4 THEN PRINT"DATA ERROR IN L
    INE";M3*5+55:PRINT M1:END
435 NEXT
440 READ E1,E2,E3,E4,E5,E6,E7,E8
445 M1=M1+E1+E2+E3+E4+E5+E6+E7+E8
450 READ M4 : IF M1<>M4 THEN PRINT"DATA ERROR IN L
    INE 395" : PRINT M1 :END
455 READ EX,EY,OFF,CCW,CW,M2,M3
460 M1=M1+EX+EY+OFF+CCW+CW+M2+M3
465 READ M4 : IF M1<>M4 THEN PRINT"DATA ERROR IN L
    INE 400" : PRINT M1 : END
470 POKE 785,M2 :POKE786,M3
475 READ P1,P2,P3,P4,ROBOT
480 M1=M1+P1+P2+P3+P4+ROBOT
485 READ M4 : IF M1<>M4 THEN PRINT"DATA ERROR IN L
    INE 405" : PRINT M1 :END
490 INIT=INIT+2
495 M1=INIT+12: M2=M1+4 : M3=M2+4 : M4=M3+4
500 SYS INIT
```

First Experiments

After such a lot of programming theory we now come to carry out the first experiments. We avail ourselves of the preliminary test configuration. By means of the enclosed 20-core cable, the motor and photo-interrupter are connected to the interface. The cable, however, should be prepared right away so that afterwards it can be pulled into the robot in form of a cable harness. The cable configuration is shown at page 62. Out of the cut-off part another cable harness is made. The remaining cable residues are separated into individual cores. These cables serve for a further wiring of the model and in particular for connecting the common ground and +5 V-wire at all keys and photo-interrupters.
There are four wires not used for the assembly of the training robot. They are allocated to the inputs EX, EY and E8. If you intend to extend the function of your robot you should not cut these wires but leave them full length, make a coil of them and attach with tape below the main cable harness. Now the cable ends are carefully installed over a length of approx. 3 to 5 mm without damaging the fine cores of the strands. The cores are then twisted. You should pull the cable into the robot before screwing the plugs in place.

For our first test it will be sufficient if output M1, inputs E1 and E2 as well as the +5 V-wire have been provided with plugs. It can be taken from the wiring scheme as well as from the inscription on the cover of the interface which cable cores are concerned. Double back the strand on the insulation. Loosen the small screw of the plug and insert the end of the cable into the sleeve. Now the screw is tightened again but not too much so as to avoid that the cable is squeezed off. Furthermore, you prepare two cables from the cable cuttings which at both ends are provided with fischertechnik plugs.
Now connect the motor with the two pertinent cables orange and yellow. The socket with the ⊕-symbol of the photo-interrupter is connected to the red +5 V-wire. Additionally, the +5 V-wire is continued to contact 2 of a mini-key. The socket with the ⊖-symbol of the photo-interrupter is connected at the separated ground socket of the interface. Finally, you connect wire E1 (brown) to contact 1 of the mini-key, and wire E2 (red) to the pulse output of the photo-interrupter ⎍.

We now revert to the aforementioned adjustment of sensitivity of the photo-interrupter. When all connections have been made and the interface connected at the switched-off(!) computer, you switch on the latter, load the program ROBOT.ADJUSTMENT and start it. To the question which photo-interrupter is to be adjusted, you respond by 1. Now the motor must run and a measuring instrument appear on the screen. The indicator of the measuring instrument must be situated on the green field, optimally it should point to 0.5. If not you may adjust the sensitivity of the photo-interrupter as described above.

If the adjustment is correct you proceed in the same way and manner with the other photo-interrupters. You may also adjust the photo-interrupter later when already installed in the robot but in that case you will have to release the rods of the robot. If these preliminary works have been finalized satisfactorily you may convince yourselves of the good functioning of the robot system commands. For this purpose load the program ROBOT.SYSTEM and start same. Similar to the driver routine, the robot system program will respond as if nothing had happened. But now you can test the previously discussed commands in direct mode. Remember, the asterisk is not part of the command but should remind you of the syntax differences of the computers.

Input:

```
* SYS P1, 64
```

and subsequently:

```
* SYS ROBOT
```

The motor must run for a short while and the wheel must turn round by a little bit more than one rotation. 64 changes in level corresponding to the 32 black and 32 bright sections are just one rotation. On account of the after-running of the motor a higher position number was obtained. Obtain this number by the command:

```
* PRINT USR(P1)
```

Perhaps, you will be astonished at the extent of the after-running. You may rest assured, however, that later the after-running in the robot will be inferior on account of the load and friction conditions.
The input in your next test will be:

```
* SYS P1, 10000
```

and

```
* SYS ROBOT
```

Now the motor will run for a longer period of time and you have got the opportunity to actuate the connected mini-key. Immediately, the motor will stop running and the command is terminated. By

```
* PRINT USR(P1)
```

you will find out the number reached by the position counter before you actuated the key.
Now we wish the motor to run in the reverse sense:

```
* SYS P1,0 : SYS ROBOT
```

Now the motor will rotate in the other direction and via zero position, the counter will get into the negative range of numbers. Make sure by:

```
* PRINT USR(P1)
```

Last but not least, here is some advice for a more precise positioning. For this purpose, you have to brake down the gear shaft. You may either install a brake mechanism or simply brake by hand. Now input:

```
* SYS P1, 64 : SYS ROBOT : SYS ROBOT :
  SYS ROBOT
```

Assembly of the Robot

Now the robot command is called in three times with the same rated value; thus the position is still corrected twice. On account of the brake, however, the motor will no longer run up to full speed and each time will stop more exactly. In this sense, you still may perform other tests and experiments when the robot has been assembled and the motors are subjected to the load of the robot arm.

For assembly of the robot you proceed as per subsequent illustrated instructions. When the robot has been assembled you should check whether all parts have been aligned and levelled exactly and whether the driving rods are of easy motion. For this purpose you may release the motors a little bit from catching with the gear toothing.

Now you start wiring the robot. Please proceed as per wiring scheme illustrated on page 91.

Begin with arresting the main cable harness in the strain relief. First of all, all motor outputs are connected at the row of lamps. Subsequently, the lines marked by an asterisk (*) and in particular the small cable harness, are inserted through the central opening of the turn-table and along the structure of the robot. Tweezers may help to pass through the cables. By means of the grey plates you can arrest the cables in the grooves of the metallic rods.

Now follows the electrical testing of the training robot. For this purpose load the diagnostic routine DIAGNOST. Check the limit switches at inputs E1, E3, E5. If not actuated, a ONE should appear on the screen. The contrary applies to the switch for monitoring the gripper, in which case the ONE appears on the screen when the switch has been actuated. It is, however, common to all limit switches that the ONE appears in the admissible operating range of the robot. Why exactly this polarity? If during operation of the robot a cable should break, the same situation will arise as if the limit switch opens. Therefore, the robot system program responds to cable breakage also by switching off the motors.

The +5 V-wire leading to all limit switches, is also equipped with a push-button switch. If this contact is opened, all inputs change for ZERO and the robot system program disables all motors. Hence, this key has got an emergency-OFF-function. It should be checked by you, too.

The operativeness of the fork-type photo-interrupters may be checked by you again by a slight turning of the printed wheel. The respective indication must move to and fro between ONE and ZERO.

Now for the motors. For first tests it will perhaps be better if the gear of the robot is still released. A motor is selected by means of the respective number keys. Now the sense of rotation either clockwise or counterclockwise can be chosen for this motor by means of the keys as indicated by the program menu. In this way and manner you can move the robot. Make sure that in clockwise rotation the spindle nut of the upper arm- and forearm drive goes down. If it moves the opposite way, the connections at the motor must be changed in poles. Likewise, the robot must rotate in clockwise direction (seen from above) if clockwise rotation is chosen. Finally, the gripper must open in case of clockwise rotation.

If everything has been checked through we now can pass on to the control of the robot.

Control of the Robot

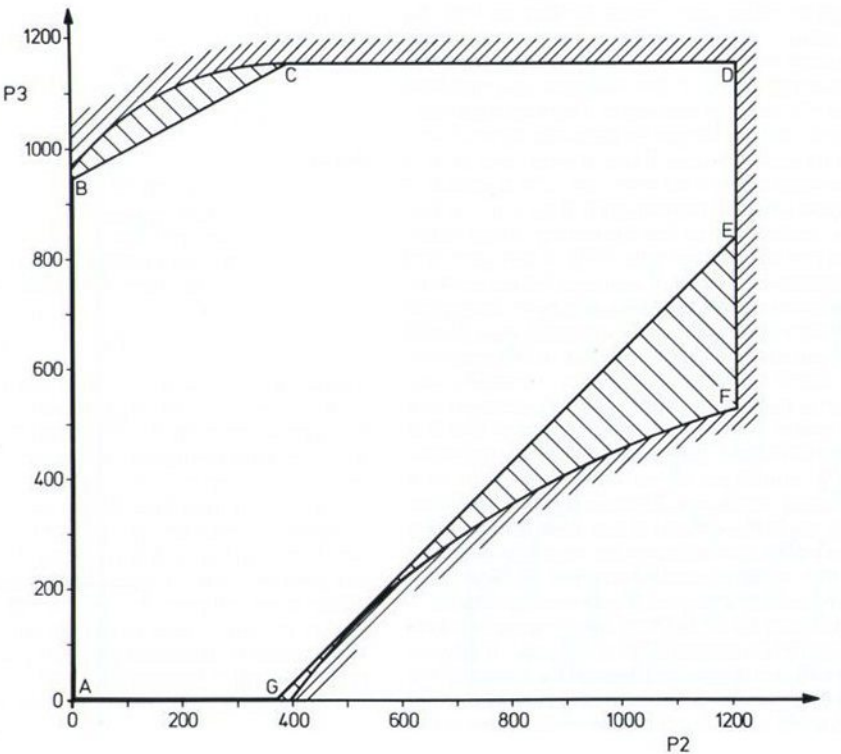
For programming the robot we first of all should familiarize with the kinds of movements of the robot. The program ROBOT.MAN will be more suitable in this respect than the diagnostic routine. In this program, the robot will be subjected to the control of the robot system program so that e.g. the limit keys and the emergency-OFF-circuit are monitored automatically. On the screen the program will indicate the operation of the robot. In the first step it will guide the robot to its home position. The home position of the robot is marked by the response of the limit key. In this position the arm of the robot is lifted, the gripper opened and the robot points to the side opposite the connecting cable. It is true that in the actual home position the limit keys have just been released as within the operating range of the robot system program the limit keys are not allowed to respond. After reaching the home position, the command

* **SYS INIT**

will be passed through so that the home position is given the position value Zero for all three axes of movement.

The movements of the robot are triggered by means of the number keys. The distance per actuation of key can be selected by means of the function keys, so that it is easy to effect either coarse or fine positioning. In contrast to the robot axes the gripper motor is activated during fixed time intervals when closing. You may change the duration of the time interval per command. Adjust it for a safe gripping of the object and avoid blocking of the gear due to excessive force on the other hand. When opening the gripper the program waits for a Low-Signal at E7 (limit key not activated). An actuation of the HOME-key will always return the robot to its home position. From its home position the robot may move only toward positive position data. This is quite natural for the two arm axes. For the rotation of the body, the home position had to be chosen rather arbitrarly. Since the robot system program cannot move the robot beyond its home position, it is recommended to arrange the main working area in front of the broad side of the base plate. This allows you to move the robot to the left and right liberally.
Experiment with your robot and try to grip and transfer the enclosed workpieces. You will become aware that you have to proceed carefully and with circumspection. You will also become familiar with the effect of the two motional axes of upper and forearm. The upper arm predominantly serves for extending and withdrawing the gripper, whilst the forearm is in first place designed for up-and-down movement. Furthermore you will find out that it is not possible to actuate both motional axes totally independent from each other. Under certain circumstances the rods will jostle here and there. Regarding this aspect we should go into details, substantiated by fig. 2.

Fig. 2

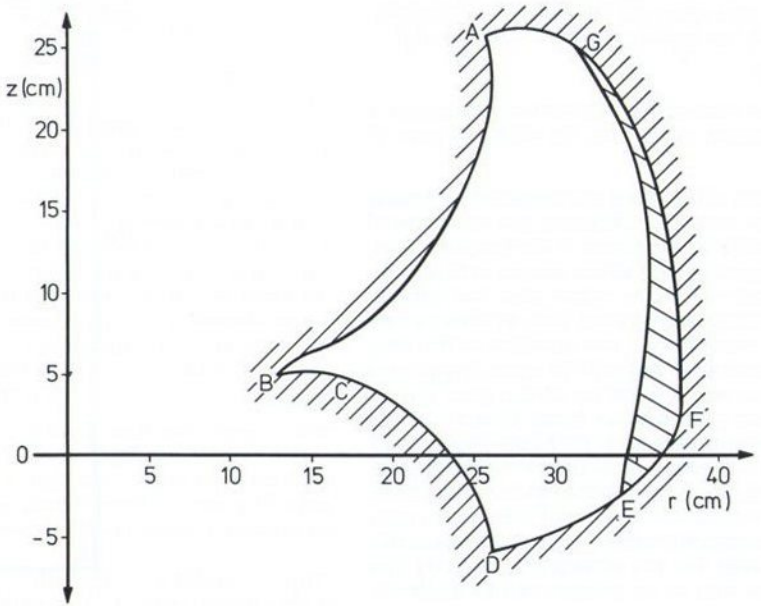


In view of the fact that the program always displays the position of the motional axes on the screen we can investigate which combinations of numbers are admissible and which are not. Each numerical position represents a spot in the system of coordinates shown in fig. 2. All permissible combinations will form an area which we shall define as internal operating range.

For determination of the internal operating range, we start at home position, coordinate (0,0). No one of the driving spindles can retract further so that the negative values of both motional axes are omitted. Now increase the position af axis 2 step by step, without altering axis 3. In the diagram you proceed along axis P2. At a certain point a further displacement of axis 2 will no longer be possible without provoking an impact of a part of the driving rods. Axis 3 must be moved away a bit from the zero position in order to allow a further movement of axis 2. In the diagram, the border line of the operating range separates from the coordinate axis. With much care and steadily observing the robot you may follow the border line until the spindle of axis 2 has been extended entirely. Now you displace the motional axis 3 until this has been entirely extended, too. In the diagram, the right border limitation extends vertically upwards. Along the upper border and on account of a decrease of the position values of motional axis 2, it turns to the left again. But also in this case, the motional axis 2 cannot be taken back entirely to zero without taking back simultaneously the motional axis 3. Arrived at the left rim of the internal operating range, the home position can be reached again by resetting the motional axis 3. Now that we have got a diagram indicating the area of admissible values, it will be of interest for us to know which movements of the robot are to be allocated to such area. The position of the robot is described easiest by means of the position of an idealized point, the "tool center point TCP". This point is the center between the jaws of the gripper. If the position of this point in space is given, the positions of all three motional axes can be derived therefrom. The interrelation between the height of TCP above base plate, z, and the distance from rotational center, r, on one hand and the axis positions P2 and P3 on the other hand results from the following equations of which further derivation should not be made on this occasion:

$r = 120*\cos\alpha + 180*\cos\beta + 110$
$z = 120*\sin\alpha + 180*\sin\beta + 117.5$
$\alpha = 126° - \delta$
$\beta = 36° - \varepsilon$
$\delta = \cos^{-1}\,[(14004-(P2*0.07363+60)^2)/12240]$
$\varepsilon = \cos^{-1}\,[(14004-(P3*0.07363+60)^2)/12240]$
(all dimensions in mm)

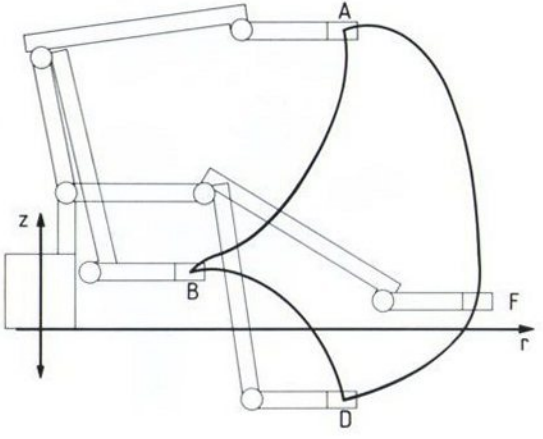
Fig. 3



When using these equations you have to take into consideration that they are representing the idealized configuration. In practice, a deviation will occur on account of the required bearing clearance. Comparing fig. 2 to fig. 3 you may find out also the interrelation between internal and real operating range. The characteristic corners are defined by the same letters in both figures. You may define the real operating range by yourselves. You follow up the border line of the operating range as per fig. 2 and measure the position of TCP. For giving you an idea of the position of the robot, fig. 4 shows the real operating range together with schematic drawings of the robot.

Now we intend to modify the program ROBOT.MAN so that we will not damage the robot accidentally. The border lines of the internal operating range have to be interrogated and any control outside of same should be avoided. For this purpose, the border line is approached by straight lines. This can be done without problems parallel to the coordinate axes and in the upper left corner of the operating range. The right corner section below deserves special attention, however. Trying to come to a definition we also have to take into consideration the operating method of the positioning system.

Fig. 4



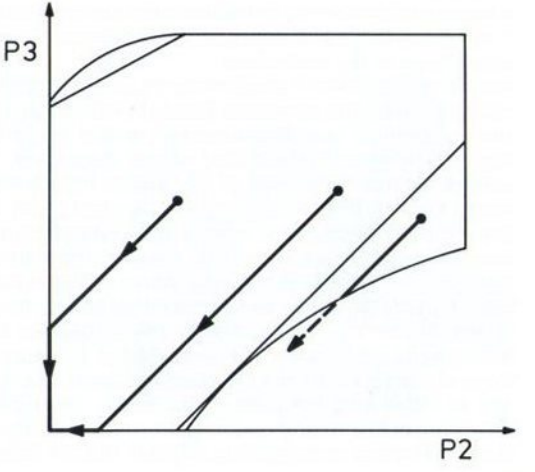
When the robot program is started, the program can have no knowledge at all of the prevailing position of the robot (in correct terms: it does not use an absolute positioning system). Any and all knowledge of position are gathered by the counting of pulses only, starting at a known position (incremental positioning system). Such an independently known position is the home-position as it results from the response of the limit switches. Therefore, the program will always bring the robot to its home position at the beginning. The procedure is as follows: all motors are started and continue running until the respective limit switch is triggered. Programming is effected by means of the simple commands without monitoring the pulse input, e.g.

*** SYS M1, CW**

Subsequently, they will run again, as described before, away from the limit switch. This path plotted inside the internal operating range, forms a diagonal as both motors of the arm-driving units are running at the same speed except for inferior exemplary scatterings. As soon as the path contacts a coordinate axis it will extend along the same through to zero (fig. 5). Let's assume e.g. that the robot has been switched off near point F. Approaching the home position, the path would leave the permissible area. Therefore, the corner section at point F has to be chosen in a way that it is formed by a straight line at an angle of 45°. These limitations of the operating area have been indicated in fig. 2 and fig. 3 as connecting line EG.

It requires only a few additional program lines to take such limitations into consideration. If you now load the program ROBOT.SPACE you have got the same possibilities of controlling the robot, as described before, but it will refuse to exceed the operating range.

Fig. 5



Teach-in Procedure

The utilization of the previous programs has been very helpful for understanding the geometry of the robot but it does not represent a programming of the robot in its literal meaning. For each movement of the robot the intervention of the operator was required. It is, however, a characteristic feature of the programming of robots that the robot performs the ordered job on its own.

In the preceding section we also had indicated the equations conveying the interrelation between the positions of the motional axes inside the internal operating range and the position of TCP in the real operating range. Basing on this knowledge it would be possible to stipulate a certain sequence of motions of the robot and to code them in a table. Correspondingly, the robot program can take the position data one after the other from the table and control the robot accordingly. This method is also used in practice, in particular if the position data have not only been taken from a table but perhaps are modified on basis of previous measurements. E. g. think of a video system connected to the robot and providing it with knowledge and data on the exact position and orientation of the workpiece.
More practicable for a predominant number of applications will be the so-called Teach-In Procedure. In this procedure, a manual control of the robot is applied as already described above. Moreover, it serves for the stipulation of the above mentioned table. Always, when reaching an essential point of the path of the robot, it is stored upon special command. By and by, a table will be created depending on the skilfulness of the operator. When the table has been completed it may serve as a pattern according to which the robot may repeat the sequence of movements on its own. A knowledge of the equations of conversion is not required and also the error rate in stipulating the path will be lower. The robot instructor is in a steady visual contact with the robot. Such a program is available as ROBOT.TEACH. Apart from the points described above, further convenient steps are incorporated for attending to and maintaining the tables of movement. E. g. the table may be stored on a floppy disk or cassette and may be loaded also again from such disk or cassette. It may be printed by means of a connected printer. Table entries may be erased and several tables may be collected and combined in the memory of the computer.
If you load and start the program, first of all a main menu will be displayed on the screen by means of which you may select the appropriate function. For the first start you may revert to an exemplary movement which has already been stored on the floppy disk resp. cassette. Therefore, you first of all have to select the menu point "L" for loading of a file. Hereafter you have to indicate the name of the file; it is SAMPLE. After loading of the file, the menu appears again. In a next step you may have printed the table of movements if you have connected a printer at your computer. This is achieved by selecting the menu point "P".
Now we want to operate the robot. By menu point "R" the execution of the movement is selected. According to the indication of the number of cycles through the table of movements the robot starts to perform the coded movement. If you do not want to wait for the termination of the movement you may return prematurely to the main menu by actuating the key "M".

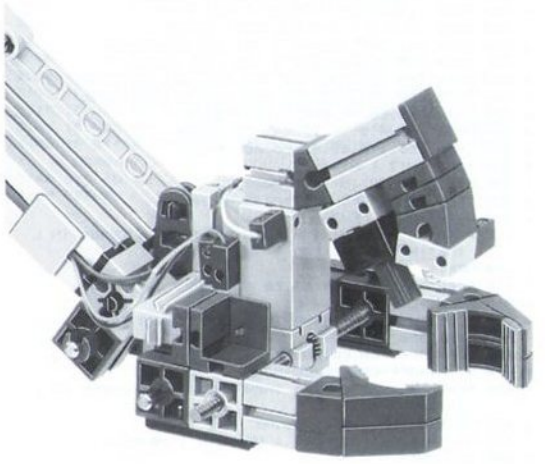
Now for the teach-in procedure. Select point of menu "T". The program will ask whether the previous sequence of movements is to be erased. Now you can choose either to extend the just stored sequence of movements or to start a new. When the question has been answered a new menu will be displayed which will be known to you already from manual control. As usual, the motors are controlled via number keys and the pulse width via function keys. It has also been provided for a triggering of the home position. A new addition is that with each actuation of the RETURN-key the presently prevailing robot position is added at the former table end. By means of the key DEL the last item of the table is, however, deleted again so that you may also correct faulty paths resp. curves.
After input of a sequence of movements according to your wishes you will return again to the main menu by pressing the key "M". In view of the arrangement of the menu, however, you may again and again call in a test run in between and subsequently continue to compose the table. Or you may, in between, secure the sequence of movements on a floppy disk or have printed them. The program ROBOT.TEACH will become your standard tool for programming the robot. At the same time, it is easily adaptable to your special requirements on account of the very detailed and informative documentation.

Further Experiments

Last but not least some advice how to use your training robot elsewhere. The illustrations below are meant to give you some idea. E.g. the gripper may also be mounted pointing downwards, this being suitable for instance for clamping chessmen. It surely will be a particular challenge for a programmer to build a chess-playing robot.

The next illustration shows an electromagnet. In this way and manner, iron parts can be easily gripped.

Finally the attachment of a reflection type photo-interrupter the gripping arm. You clearly can see the lamp which illuminates the space between the jaws. It is connected directly to the power unit. The neighboured photoresistor is protected by a cap and a tube so that it cannot be influenced by the direct light of the lamp. It reacts, however, by a distinct alteration of the resistance as soon as a bright object gets into the gripping area. Such alteration of resistance can be recorded by the interface input EX or EY and forwarded to the computer. By suitable search programs an object can be recognized in this way and manner and the gripper can be aligned and levelled exactly above it. Now the arm has only to be lowered to take hold of the object. This job, too, will be a good challenge to your art of programming.



Prog. ROBOT.ADJUST

```
* 1 PRINT CHR$(147)
* 2 PRINT"□"
* 3 POKE53280,0
* 4 POKE53281,0
* 5 PRINT"■■■■■■■■■"
* 7 PRINT"          LOADING DRIVER"
* 10 REM DRIVER FOR COMMODORE 64
* 20 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1984
* 30 REM COMMANDS OF THE DRIVER
* 40 REM SYS M1,OFF
* 50 REM SYS M1,CCW     SYS M1,CW
* 60 REM USR(E1)   USR(EX)   USR(EY)
* 70 REM M1 - M4 ARE MOTORS
* 80 REM E1 - E8 ARE DIGITAL INPUTS
* 90 REM EX - EY ARE ANALOG INPUTS
* 100 DATA 52736,169,0,240,38,169,3,208,10,53573
* 110 DATA 169,12,208,6,169,48,208,2,54395
* 120 DATA 169,192,120,133,255,32,253,174,55723
* 130 DATA 165,254,5,255,133,254,32,158,56979
* 140 DATA 183,138,37,255,133,255,165,254,58399
* 150 DATA 69,255,133,254,168,169,63,141,59651
* 160 DATA 3,221,162,8,169,48,6,254,60522
* 170 DATA 144,2,9,4,141,1,221,9,61053
* 180 DATA 8,141,1,221,202,208,237,169,62240
* 190 DATA 57,141,1,221,132,254,88,96,63230
* 200 DATA 120,32,161,183,134,255,169,0,64284
* 210 DATA 141,156,206,169,255,141,155,206,65713
* 220 DATA 169,50,141,1,221,9,8,141,66453
* 230 DATA 1,221,162,8,10,44,1,221,67121
* 240 DATA 16,2,9,1,160,48,140,1,67498
* 250 DATA 221,160,56,140,1,221,202,208,68707
* 260 DATA 235,37,255,240,2,169,1,24,69670
* 270 DATA 109,156,206,141,156,206,206,155,71005
* 280 DATA 206,208,205,172,156,206,32,162,72352
* 290 DATA 179,88,96,0,0,0,0,0,72715
* 350 DATA 1,2,4,8,16,32,64,128,72970
* 360 DATA 255,170,85,85,80,206,73851
* 370 READ INIT : M1=INIT
* 380 FOR M3=0 TO 19: FOR M2=0 TO 7
* 390 READ M4 : POKE INIT+M3*8+M2,M4
* 400 M1=M1+M4 : NEXT
* 410 READ M4 : IF M1<>M4 THEN PRINT"DATA ERROR IN L
      INE";M3*10+100:END
* 420 NEXT
* 430 READ E1,E2,E3,E4,E5,E6,E7,E8
* 440 M1=M1+E1+E2+E3+E4+E5+E6+E7+E8
* 450 READ M4 : IF M1<>M4 THEN PRINT"DATA ERROR IN L
      INE 350" : END
* 460 READ OFF,CCW,CW,CW,M2,M3
* 470 M1=M1+OFF+CCW+CW+CW+M2+M3
* 480 READ M4 : IF M1<>M4 THEN PRINT"DATA ERROR IN L
      INE 360",M1: END
* 485 POKE785,M2 : POKE786,M3
* 490 M1=INIT+4 : M2=M1+4 : M3=M2+4 : M4=M3+4
* 500 SYS INIT
  510 REM
  520 REM FISCHERTECHNIK COMPUTING
  530 REM
  540 REM PHOTO-INTERRUPTER ADJUSTMENT
  550 REM
  560 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1985
  570 REM
  580 REM ASSIGNMENTS FOR THE INTERFACE
  590 REM MOTOR   PULSE      COMMAND
  600 REM         INPUT
* 610 REM M1        E2       USR(E2)
* 620 REM M2        E4       USR(E4)
* 630 REM M3        E6       USR(E6)
* 640 REM M4        E8       USR(E8)
  650 REM
  660 REM FUNCTION DESCRIPTION:
  670 REM THE PROGRAM SERVES TO ADJUST
  680 REM THE PULSE WIDTH OF THE PHOTO-INTERRUPTER.
  690 REM THE BEST WORKING CONDITION IS GIVEN
  700 REM WHEN THE LOW PERIOD OF THE SIGNAL
  710 REM IS AS LONG AS THE HIGH PERIOD.
  720 REM YOU CAN ADJUST THIS BY TURNING
  730 REM CAUTIOUSLY THE INNER POT EITHER
  740 REM CW OR CCW. THE ARROW SHOULD BE
  750 REM IN THE GREEN FIELD.
  760 REM
* 1000 PRINT CHR$(147)
  1010 PRINT"WHICH PHOTO-INTERRUPTER SHOULD"
  1020 INPUT"BE ADJUSTED ";L
  1030 IF L=1THEN LET E=E2:M=M1
  1040 IF L=2THEN LET E=E4:M=M2
  1050 IF L=3THEN LET E=E6:M=M3
  1060 IF L=4THEN LET E=E8:M=M4
  1070 IF L>4 OR L<1 THEN GOTO 1020
* 1080 PRINT CHR$(147)
* 1090 PRINT CHR$(28)"      F I S C H E R"CHR$(31)"
       T E C H N I K"
  1100 PRINT
* 1110 PRINT CHR$(158)"          C O M P U T I N G"C
       HR$(31)
  1120 PRINT
  1130 PRINT" PULSE WIDTH ADJUSTMENT OF THE LDR "
  1140 PRINT:PRINT
  1150 PRINT"0       0.25       0.5       0.75     1
       "
* 1160 DATA 180,160,160,160,160,183,160,160
* 1170 DATA 160,160,164,160,160,160,160,183
* 1180 DATA 160,160,160,160,183,160,160,160
* 1190 DATA 160,160,183,160,160,160,160,180
* 1200 DATA 160,160,160,183,160,160,160,170
  1210 FOR C=0 TO 39
  1220 READ D
  1230 PRINT CHR$(D);
  1240 NEXT C
* 1250 DATA 28,111,183,183,183,183,183,183
* 1260 DATA 183,183,183,183,183,183,183,30
* 1270 DATA 183,183,183,183,183,183,183,183
* 1280 DATA 183,183,183,183,183,183,28,183
* 1290 DATA 183,183,183,183,183,183,183,183
* 1300 DATA 183,183,112
  1310 FOR C=0 TO 42
  1320 READ D
  1330 PRINT CHR$(D);
  1340 NEXT C
  1350 PRINT
* 1360 DATA 28,108,175,175,175,175,175,175
* 1370 DATA 175,175,175,175,175,175,175,30
* 1380 DATA 175,175,175,175,175,175,175,175
* 1390 DATA 175,175,175,175,175,175,28,175
* 1400 DATA 175,175,175,175,175,175,175,175
* 1410 DATA 175,175,186
  1420 FOR C=0 TO 42
  1430 READ D
  1440 PRINT CHR$(D);
  1450 NEXT C
* 2000 SYS M,CCW
  2010 PRINT:PRINT
* 2020 PRINT CHR$(31)"PLEASE POSITION ARROW IN THE "
  2030 PRINT"GREEN AREA BY TURNING THE"
* 2040 PRINT"SCREW-DRIVER "CHR$(18);CHR$(28)"CAUTIOU
       SLY";
* 2050 PRINT CHR$(31);CHR$(146)" CW OR CCW."
  2060 PRINT
  2070 PRINT:PRINT
* 2080 PRINT"STOP MEASUREMENT WITH "CHR$(18)"F1"CHR$
       (146)
  2090 FOR I=1 TO 11
* 2100 PRINT CHR$(145);
  2110 NEXTI
* 2120 PRINT SPC(38/255*USR(E));CHR$(5)"↑"CHR$(31);
* 2130 PRINT CHR$(145)
  2140 PRINT"
       ";
* 2150 PRINT CHR$(145);
* 2160 GET A$:IF A$=CHR$(133)THEN RESTORE:GOTO 10
  2170 GOTO 2120
```

Prog. ROBOT.MAN

```
* 500 SYS INIT
  510 REM
  520 REM FISCHERTECHNIK COMPUTING
  530 REM
  540 REM MANUAL CONTROL OF THE ROBOT
  550 REM WITH RECORD OF ACTUAL POSITION.
  560 REM
  570 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1985
  580 REM
  590 REM FUNCTION DESCRIPTION:
  600 REM THE ROBOT IS MANUALLY CONTROLLED VIA THE K
      EYBOARD.
  610 REM SIMULTANEOUSLY THE POSITION OF THE ROBOT
  620 REM IS SCANNED VIA THE SECTORWHEEL.
  630 REM THE ROBOT IS CONTROLLED BY THE KEYS 1...8.
* 640 REM THE KEYS F1-F7 DEFINE THE STEPWIDTH OF THE
      MOTOR
* 650 REM MOVEMENT PER KEYSTROKE.
  660 REM KEY DEFINITIONS:
  670 REM 1 AND 2 = M1 CW UND CCW
  680 REM 3 AND 4 = M2 CW UND CCW
  690 REM 5 AND 6 = M3 CW UND CCW
  700 REM 7 AND 8 = M4 CW UND CCW
* 710 REM F1=256 SECTORS PER KEYSTROKE
* 720 REM F3=64 SECTORS PER KEYSTROKE
* 730 REM F5=16 SECTORS PER KEYSTROKE
* 740 PEM F7=4 SECTORS PER KEYSTROKE
* 750 REM HOME=MOVE TO HOME POSITION
* 760 PRINT CHR$(147)
  770 PRINT:PRINT
* 780 PRINT CHR$(28)"      F I S C H E R"CHR$(31)" T
      E C H N I K"CHR$(144)
  790 PRINT
  800 PRINT"           C O M P U T I N G"
  810 PRINT:PRINT
  820 PRINT"      T R A I N I N G   R O B O T"
  830 PRINT
  840 PRINT"      W I T H   T H R E E   A X E S"
  850 PRINT:PRINT
  860 PRINT" CONTROL VIA KEYBOARD"
  870 PRINT" THE ROBOT POSITION IS RECORDED  "
  890 PRINT
  900 PRINT" ROBOT MOVES TO HOME POSITION  "
  910 GOSUB 3000
  920 LET ZZ=200
* 1000 PRINT CHR$(147)
  1010 PRINT"KEY FUNCTION:                    POSITION:"
  1020 PRINT:
* 1030 PRINT CHR$(18)"1"CHR$(146)"  ROBOT MOVES CW"
* 1040 PRINT CHR$(18)"2"CHR$(146)"  ROBOT MOVES CCW"
  1050 PRINT
* 1060 PRINT CHR$(18)"3"CHR$(146)"  UPPER ARM BACK"
* 1070 PRINT CHR$(18)"4"CHR$(146)"  UPPER ARM FORW."
  1080 PRINT
* 1090 PRINT CHR$(18)"5"CHR$(146)"  FOREARM UP"
* 1100 PRINT CHR$(18)"6"CHR$(146)"  FOREARM DOWN"
  1110 PRINT
* 1120 PRINT CHR$(18)"7"CHR$(146)"  OPEN GRAB"
* 1130 PRINT CHR$(18)"8"CHR$(146)"  CLOSE GRAB"
  1140 PRINT
  1150 PRINT "NUMBER OF STEPS"
* 1155 PRINT"OPERATION TIME GRIPPER    "CHR$(18)"+/-"
      CHR$(146)
  1160 PRINT
* 1170 PRINT CHR$(18)"F1"CHR$(146)"  256 SECTORS"
* 1180 PRINT CHR$(18)"F3"CHR$(146)"   64 SECTORS"
* 1190 PRINT CHR$(18)"F5"CHR$(146)"   16 SECTORS"
* 1200 PRINT CHR$(18)"F7"CHR$(146)"    4 SECTORS"
  1210 PRINT
* 1220 PRINT CHR$(18)"HOME"CHR$(146)"  MOVE TO HOME
      POSITION"
  2000 LET Q%=16
* 2010 LET CL$="      "+CHR$(157)+CHR$(157)+CHR$(157)
      +CHR$(157)+CHR$(157)
* 2020 LET Q1%=USR(P1):IF Q1%<0 THEN Q1%=0
* 2030 LET Q2%=USR(P2):IF Q2%<0 THEN Q2%=0
* 2040 LET Q3%=USR(P3):IF Q3%<0 THEN Q3%=0
  2050 REM CONTROL VIA KEYBOARD
* 2060 LET A=PEEK(203):REM READ KEYBOARD REGISTER
* 2070 IF A=56 THEN Q1%=Q1%+Q% : REM MOVE CCW
* 2080 IF A=59 THEN Q1%=Q1%-Q% : REM MOVE CW
  2090 REM MOVE BODY OF THE ROBOT
  2100 IF Q1%<0 THEN Q1%=0
* 2120 PRINT CHR$(19)
* 2130 PRINT CHR$(17);CHR$(17);
* 2140 PRINT TAB(30);CL$;Q1%
* 2150 SYS P1,Q1%
* 2160 IF A=8  THEN Q2%=Q2%+Q% : REM STRECH OUT UPPE
      R ARM
* 2170 IF A=11 THEN Q2%=Q2%-Q% : REM RETRACT UPPER A
      RM
  2180 REM MOVE UPPER ARM
  2220 IF Q2%<0 THEN Q2%=0
* 2230 PRINT CHR$(17);CHR$(17);
* 2240 PRINT TAB(30);CL$;Q2%
* 2250 SYS P2,Q2%
* 2260 IF A=16 THEN Q3%=Q3%+Q% : REM MOVE DOWN FOREA
      RM
* 2270 IF A=19 THEN Q3%=Q3%-Q% : REM LIFT FOREARM
  2280 REM MOVE FOREARM
  2320 IF Q3%<0 THEN Q3%=0
* 2330 PRINT CHR$(17);CHR$(17);
* 2340 PRINT TAB(30);CL$;Q3%
* 2350 SYS P3,Q3%
  2360 REM START ROBOT
* 2370 SYS ROBOT
  2380 REM GRAB ROUTINE
* 2390 IF USR(E7)=0 THEN PRINT CHR$(17);CHR$(17);TAB
      (30);"OPEN "
* 2400 IF USR(E7)=1 THEN PRINT CHR$(17);CHR$(17);TAB
      (30);"CLOSE"
* 2410 IF A<>24 THEN GOTO 2470
  2420 IF ZA$="OPEN " THEN GOTO 2470
  2430 LET ZA$="OPEN "
* 2440 SYS M4,CW
* 2450 IF USR(E7)=1 THEN GOTO 2440
* 2460 SYS M4,OFF
* 2470 IF A<>27 THEN GOTO 2540
  2480 IF ZA$="CLOSED" THEN GOTO 2540
  2490 LET ZA$="CLOSED"
  2500 FOR Z=1 TO ZZ
* 2510 SYS M4,CCW
  2520 NEXT
* 2530 SYS M4,OFF
  2540 REM DEFINE GRIPPERWIDTH
* 2542 GET A$
  2543 IF A$="+" THEN IF ZZ<500 THEN LET ZZ=ZZ+50
  2547 IF A$="-" THEN IF ZZ>50 THEN LET ZZ=ZZ-50
* 2550 IF A=4 THEN Q%=256
* 2560 IF A=5 THEN Q%=64
* 2570 IF A=6 THEN Q%=16
* 2580 IF A=3 THEN Q%=4
* 2590 PRINT CHR$(17);CHR$(17);
* 2600 PRINT TAB(30);CL$;Q%
* 2605 PRINT TAB(30);CL$;ZZ
* 2610 IF A=51 THEN GOSUB 3000
  2620 GOTO 2020
  3000 REM MOVE TO HOME POSITION
  3010 LET H1=1:H2=1:H3=1:H4=1
* 3020 IF USR(E1)=1 AND H1=1 THEN SYS M1,CW
* 3030 IF USR(E3)=1 AND H2=1 THEN SYS M2,CW
* 3040 IF USR(E5)=1 AND H3=1 THEN SYS M3,CW
* 3050 IF USR(E7)=1 AND H4=1 THEN SYS M4,CW
* 3060 IF USR(E1)=0 THEN SYS M1,CCW :H1=-1
* 3070 IF USR(E3)=0 THEN SYS M2,CCW :H2=-1
* 3080 IF USR(E5)=0 THEN SYS M3,CCW :H3=-1
* 3090 IF USR(E7)=0 THEN SYS M4,OFF:H4=0
* 3100 IF USR(E1)=1 AND H1=-1 THEN SYS M1,OFF:H1=0
* 3110 IF USR(E3)=1 AND H2=-1 THEN SYS M2,OFF:H2=0
* 3120 IF USR(E5)=1 AND H3=-1 THEN SYS M3,OFF:H3=0
  3130 IF H1<>0 OR H2<>0 OR H3<>0 OR H4<>0 THEN GOTO
       3020
* 3140 SYS INIT
  3150 RETURN
```

Prog. ROBOT.SPACE

```
560 REM AND CONTROL OF THE WORKING AREA


2110 IF 01%>3600 THEN 01%=3600


2190 IF 02%>370+03% THEN 02%=370+03%
2200 IF 02%>1210 THEN 02%=1210
2210 IF 02%<-1709+1.82*03% THEN 02%=-1709+1.82*03%


2290 IF 03%>940+0.55*02% THEN 03%=940+0.55*02%
2300 IF 03%>1160 THEN 03%=1160
2310 IF 03%<-370+02% THEN 03%=-370+02%
```

Prog. ROBOT.TEACH

```
* 500 SYS INIT
  510 REM
  520 REM FISCHERTECHNIK COMPUTING
  530 REM
  540 REM TRAINING ROBOT WITH TEACH IN MODE
  550 REM
  560 REM COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1986
  570 REM
  580 REM FUNCTION DESCRIPTION:
  590 REM FIRST THE PROGRAM DISPLAYS THE MAIN MENU.
  600 REM YOU CAN CHOOSE OUT OF THE FOLLOWING FUNCTI
      ONS:
  610 REM T = TEACH IN MODE
  620 REM R = EXECUTE PROGRAM
  630 REM S = SAVE PROGRAM
  640 REM L = LOAD PROGRAM
  650 REM D = DIRECTORY
  660 REM P = PRINT PROGRAM
  670 REM E = END OF PROGRAM
  680 REM
  690 REM THE TEACH IN MODE LEADS TO A SUB-MENU.
  700 REM IT INVOLVES THE COMMANDS TO CONTROL
  705 REM THE ROBOT AND TO RECORD THE PROGRAM
  710 REM KEY DEFINITIONS:
  720 REM 1 AND 2 = M1 ROBOT CW AND CCW (BODY)
  730 REM 3 AND 4 = M2 UPPER ARM CW AND CCW (UPPER A
      RM)
  740 REM 5 AND 6 = M3 LOWER ARM CW AND CCW (FOREARM
      )
  750 REM 7 AND 8 = M4 OPEN AND CLOSE GRIPPER
* 760 REM F1 = 256 SECTORS PER KEYSTROKE
* 770 REM F3 =  64 SECTORS PER KEYSTROKE
* 780 REM F5 =  16 SECTORS PER KEYSTROKE
* 790 REM F7 =   4 SECTORS PER KEYSTROKE
* 800 REM HOME = HOME POSITION
  810 REM RETURN = RECORD ACTUAL POSITION
* 820 REM DEL = DELETE LAST RECORD
  830 REM M = BACK TO THE MAIN MENU
  1000 DIM DR(100),OA(100),UA(100),ZA$(100)
  1010 LET ID=-1:REM POINTER INTO THE TEACH TABLE
  1015 LET ZZ=200: REM OPERATION TIME GRIPPER
  1020 GOSUB 10000
  1030 PRINT
* 1040 PRINT CHR$(18)"T"CHR$(146)"   TEACH IN MODE"
  1050 PRINT
* 1060 PRINT CHR$(18)"R"CHR$(146)"   EXECUTE PROGRAM
      "
  1070 PRINT
* 1080 PRINT CHR$(18)"S"CHR$(146)"   SAVE PROGRAM"
  1090 PRINT
* 1100 PRINT CHR$(18)"L"CHR$(146)"   LOAD PROGRAM"
  1110 PRINT
* 1120 PRINT CHR$(18)"D"CHR$(146)"   DIRECTORY"
  1130 PRINT
* 1140 PRINT CHR$(18)"P"CHR$(146)"   PRINT PROGRAM"
  1150 PRINT
* 1160 PRINT CHR$(18)"E"CHR$(146)"   END";
  1170 REM SCAN KEYBOARD
* 1180 GET A$
  1190 IF A$="" THEN GOTO 1180
  1200 IF A$="T" THEN GOTO 2010:REM TEACH IN MODE
  1210 IF A$="R" THEN GOTO 4010:REM EXECUTE MODE
  1220 IF A$="S" THEN GOTO 5010:REM SAVE
  1230 IF A$="L" THEN GOTO 6010:REM LOAD
  1240 IF A$="D" THEN GOTO 9010:REM DIRECTORY
  1250 IF A$="P" THEN GOTO 7010:REM PRINT
  1260 IF A$="E" THEN GOTO 8010:REM END
  1270 GOTO 1180
  2000 REM TEACH IN MODE
  2010 IF ID=-1 THEN GOTO 2100
* 2020 PRINT CHR$(147)
  2030 FOR I=1 TO 5
  2040 PRINT
  2050 NEXT I
  2060 INPUT"CLEAR OLD PROGRAM (Y/N)";K$
  2070 IF K$="N"THEN 2130
  2080 IF K$<>"Y"THEN 2060
  2090 LET ID=-1
* 2100 PRINT CHR$(147)
  2110 PRINT"ROBOT MOVES TO HOME POSITION"
  2120 GOSUB 11000: REM HOME
* 2130 PRINT CHR$(147)
  2140 PRINT"KEYS FUNCTION:                POSITION:"
  2150 PRINT
* 2160 PRINT CHR$(18)"1"CHR$(146)"  ROBOT CCW "
* 2170 PRINT CHR$(18)"2"CHR$(146)"  ROBOT CW"
* 2180 PRINT CHR$(18)"3"CHR$(146)"  UPPER ARM BACK"
* 2190 PRINT CHR$(18)"4"CHR$(146)"  UPPER ARM FORW."
* 2200 PRINT CHR$(18)"5"CHR$(146)"  FOREARM UP"
* 2210 PRINT CHR$(18)"6"CHR$(146)"  FOREARM DOWN"
* 2220 PRINT CHR$(18)"7"CHR$(146)"  OPEN GRIPPER"
* 2230 PRINT CHR$(18)"8"CHR$(146)"  CLOSE GRIPPER"
  2240 PRINT
  2250 PRINT "NUMBER OF STEPS"
* 2255 PRINT "OPERATION TIME GRIPPER "CHR$(18)"+/-"C
      HR$(146)
  2260 PRINT
* 2270 PRINT CHR$(18)"F1"CHR$(146)"  256 SECTORS ";
* 2280 PRINT TAB(20);CHR$(18)"M "CHR$(146)"     MENU
      "
* 2290 PRINT CHR$(18)"F3"CHR$(146)"   64 SECTORS   "
      ;
* 2300 PRINT TAB(20);CHR$(18)"HOME"CHR$(146)"   HOME
      "
* 2310 PRINT CHR$(18)"F5"CHR$(146)"   16 SECTORS ";
* 2320 PRINT TAB(20);CHR$(18)"RETURN"CHR$(146)" TEAC
      H"
* 2330 PRINT CHR$(18)"F7"CHR$(146)"    4 SECTORS ";
* 2340 PRINT TAB(20);CHR$(18)"DEL"CHR$(146)"    DELE
      TE"
  2350 PRINT
  2360 PRINT
  2370 PRINT" NR BODY  UPPER ARM  FOREARM   GRIPPER"
  2380 PRINT"
```

```
      ";
* 2390 PRINT CHR$(145);
  3000 LET Q%=16
* 3010 LET CL$="      "+CHR$(157)+CHR$(157)+CHR$(157)
      +CHR$(157)+CHR$(157)
  3020 REM LOOP
* 3030 LET Q1%=USR(P1)
* 3040 LET Q2%=USR(P2)
* 3050 LET Q3%=USR(P3)
  3060 REM CONTROL VIA KEYBOARD
* 3070 LET A=PEEK(203):REM READ KEYBOARD REGISTER
* 3080 IF A=56 THEN Q1%=Q1%+Q% : REM MOVE CCW
* 3090 IF A=59 THEN Q1%=Q1%-Q% : REM MOVE CW
  3100 REM MOVE BODY OF THE ROBOT
  3110 IF Q1%<0 THEN Q1%=0
  3120 IF Q1%>3600 THEN Q1%=3600
* 3130 PRINT CHR$(19)
* 3140 PRINT CHR$(17);CHR$(17);
* 3150 PRINT TAB(30);CL$;Q1%
* 3160 SYS P1,Q1%
* 3170 IF A=8  THEN Q2%=Q2%+Q% : REM STRETCH OUT UPP
      ER ARM
* 3180 IF A=11 THEN Q2%=Q2%-Q% : REM RETRACT UPPER A
      RM
  3190 REM MOVE UPPER ARM
  3200 IF Q2%>370+Q3% THEN Q2%=370+Q3%
  3210 IF Q2%>1210 THEN Q2%=1210
  3220 IF Q2%<-1709+1.82*Q3% THEN Q2%=-1709+1.82*Q3%
  3230 IF Q2%<0 THEN Q2%=0
* 3240 PRINT CHR$(17);
* 3250 PRINT TAB(30);CL$;Q2%
* 3260 SYS P2,Q2%
* 3270 IF A=16 THEN Q3%=Q3%+Q% : REM MOVE DOWN FOREA
      RM
* 3280 IF A=19 THEN Q3%=Q3%-Q% : REM LIFT FOREARM
  3290 REM MOVE FOREARM
  3300 IF Q3%>940+0.55*Q2% THEN Q3%=940+0.55*Q2%
  3310 IF Q3%>1160 THEN Q3%=1160
  3320 IF Q3%<-370+Q2% THEN Q3%=-370+Q2%
  3330 IF Q3%<0 THEN Q3%=0
* 3340 PRINT CHR$(17);
* 3350 PRINT TAB(30);CL$;Q3%
* 3360 SYS P3,Q3%
  3370 REM START ROBOT
* 3380 SYS ROBOT
  3390 REM GRIPPER ROUTINE
* 3400 IF USR(E7)=0 THEN PRINT CHR$(17);TAB(30);"OPE
      N "
* 3410 IF USR(E7)=1 THEN PRINT CHR$(17);TAB(30);"CLO
      SE"
* 3420 IF A<>24 THEN GOTO3480
  3430 IF ZA$="OPEN " THEN GOTO 3480
  3440 LET ZA$="OPEN "
* 3450 SYS M4,CW
* 3460 IF USR(E7)=1 THEN GOTO3450
* 3470 SYS M4,OUT
* 3480 IF A<>27 THEN GOTO 3560
  3490 IF ZA$="CLOSED" THEN GOTO 3560
  3500 LET ZA$="CLOSED"
  3510 FOR Z=1 TO 1.4*ZZ
* 3520 SYS M4,CCW
  3530 NEXT
* 3540 SYS M4,OUT
  3550 REM DEFINE STEPS
* 3560 GET A$:REM CONTROL VIA KEYBOARD
  3563 IF A$="+" THEN IF ZZ<500 THEN LET ZZ=ZZ+50
  3567 IF A$="-" THEN IF ZZ>50 THEN LET ZZ=ZZ-50
* 3570 IF A$=CHR$(133) THEN Q%=256
* 3580 IF A$=CHR$(134) THEN Q%=64
* 3590 IF A$=CHR$(135) THEN Q%=16
* 3600 IF A$=CHR$(136) THEN Q%=4
* 3610 PRINT CHR$(17);CHR$(17);
* 3620 PRINT TAB(30);CL$;Q%
* 3625 PRINT TAB(30);CL$;ZZ
* 3630 IF A$=CHR$(19) THEN GOSUB 11000:REM HOME
  3640 IF A$<>"M" THEN GOTO 3680: REM M =MENU
  3650 LET IMAX=ID
* 3660 PRINT CHR$(147)
  3670 GOTO 1020
* 3680 IF A$<>CHR$(13) THEN GOTO 3850:REM LEARN
  3690 REM RECORD SIGNAL
* 3700 S=54272
* 3710 POKE S+24,15
* 3720 POKE S+6,240
* 3730 POKE S+1,90
* 3740 POKE S+4,17
* 3750 POKE S+24,0:POKE S+1,0:POKE S+4,0:POKE S+6,0
  3760 REM STORE VALUES IN TEACH TABLE
  3770 LET ID=ID+1
* 3780 LET DR(ID)=USR(P1)
* 3790 LET OA(ID)=USR(P2)
* 3800 LET UA(ID)=USR(P3)
  3810 LET ZA$(ID)=ZA$
  3820 GOSUB 12010: REM PRINT ACTUAL POSITION
  3830 GOTO 3030
  3840 REM DELETE LAST RECORD
* 3850 IF A$<>CHR$(20) THEN GOTO 3030
  3860 IF ID>0 THEN LET ID=ID-1
  3870 REM DELETE SIGNAL
* 3880 POKE S+24,15
* 3890 POKE S+6,240
* 3900 POKE S+1,70
* 3910 POKE S+4,17
* 3920 POKE S+24,0:POKE S+1,0:POKE S+4,0:POKE S+6,0
  3930 GOSUB 12010: REM PRINT ACTUAL POSITION
  3940 GOTO 3030
  4000 REM EXECUTE MODE
  4010 GOSUB 10000 : REM PRINT TITLE
  4020 FOR T=1 TO5
* 4030 PRINT CHR$(17)
  4040 NEXT T
  4050 PRINT"> M <  MENU"
* 4060 PRINT CHR$(19)
  4070 FOR U =1 TO 5
* 4080 PRINT CHR$(17)
  4090 NEXT U
  4100 PRINT
  4110 PRINT"           EXECUTE MODE"
  4120 PRINT
  4130 INPUT"HOW MANY TIMES  ";D
  4140 FOR Y=1 TO D
  4150 GOSUB 11000: REM HOME
* 4160 PRINT CHR$(147)
  4170 PRINT"TEACH IN TABLE "
  4180 PRINT
  4190 PRINT" NR BODY UPPERARM  FOREARM    GRIPPER"
  4200 PRINT
  4210 FOR I=0 TO IMAX
* 4220 GET A$
  4230 IF A$="M" THEN 1020
  4240 PRINT I;TAB(3);DR(I);TAB(12);OA(I);TAB(21);UA
      (I);TAB(31);ZA$(I)
* 4245 I1=USR(P1): IF I1<0 THEN LET I1=0
* 4250 SYS P1,I1: IF ABS(DR(I)-I1)>10 THEN SYS P1,DR
      (I)
* 4255 I2=USR(P2): IF I2<0 THEN LET I2=0
* 4260 SYS P2,I2: IF ABS(OA(I)-I2)>10 THEN SYS P2,OA
      (I)
* 4265 I3=USR(P3): IF I3<0 THEN LET I3=0
* 4270 SYS P3,I3: IF ABS(UA(I)-I3)>10 THEN SYS P3,UA
      (I)
* 4280 SYS ROBOT
  4290 REM GRIPPER ROUTINE
  4300 IF ZA$(I)<>"CLOSED"THEN GOTO4360
  4310 IF ZA$="CLOSED" THEN4360
  4320 FOR Z=1 TO ZZ
* 4330 SYS M4,CCW
  4340 NEXT Z
  4350 LET ZA$="CLOSED"
  4360 IF ZA$(I)<>"OPEN " THEN GOTO 4420
  4370 IF ZA$="OPEN " THEN 4420
* 4380 SYS M4,CW
* 4390 IF USR(E7)=1 THEN GOTO4370
  4400 LET ZA$="OPEN "
* 4410 SYS M4,OUT
  4420 NEXT I
  4430 NEXT Y
  4440 GOTO 1020
  5000 REM SAVE PROGRAM ON DISK
* 5010 PRINT CHR$(147)
  5020 PRINT
  5030 PRINT"SAVE PROGRAM ON DISK"
  5040 PRINT
  5050 LET F$=""
  5060 INPUT"FILENAME";F$
  5070 IF F$="" THEN GOTO 1020
* 5080 OPEN15,8,15
* 5090 OPEN 8,8,8,F$+",W"
* 5100 INPUT#15,FE,FT$,SP,SE
* 5110 IF FE=0 THEN GOTO 5200
* 5120 IF FE<>63 THEN GOTO 6250
```

```
* 5130 PRINT"FILE EXISTS "
* 5140 INPUT"DELETE OLD FILE (Y/N)";C$
* 5150 IF C$="N" THEN GOTO 6200
* 5160 IF C$<>"Y" THEN 5270
* 5170 PRINT#15,"S:"+F$
* 5180 CLOSE 8
* 5190 OPEN 8,8,8,F$+",W"
* 5200 PRINT#8,IMAX
  5210 FOR I=0 TO IMAX
* 5220 PRINT#8,DR(I)
* 5230 PRINT#8,OA(I)
* 5240 PRINT#8,UA(I)
* 5250 PRINT#8,ZA$(I)
  5260 NEXT I
* 5270 CLOSE 8
* 5280 CLOSE 15
  5290 GOTO 1020
  6000 REM LOAD PROGRAM
* 6010 PRINT CHR$(147)
  6020 PRINT
  6030 PRINT"LOAD PROGRAM FROM DISK"
  6040 PRINT
  6050 LET F$=""
  6060 INPUT"FILENAME";F$
  6070 IF F$="" THEN GOTO 1020
* 6080 OPEN 8,8,8,F$+",R"
* 6090 OPEN 15,8,15
* 6100 INPUT#15,FE,FT$,SP,SE
* 6110 IF FE<>0 THEN GOTO 6250
* 6120 INPUT#8,IMAX
* 6130 PRINT IMAX;"POSITION DATA"
  6140 FOR I=0 TO IMAX
* 6150 INPUT#8,DR(I)
* 6160 INPUT#8,OA(I)
* 6170 INPUT#8,UA(I)
* 6180 INPUT#8,ZA$(I)
  6190 NEXT I
* 6200 CLOSE 8
  6210 CLOSE 15
  6220 LET ID=IMAX:ZA$=ZA$(IMAX)
  6230 GOTO 1020
  6240 REM DISK ERROR
* 6250 PRINT FT$
  6260 PRINT"> M < MENU"
* 6270 GET A$
  6280 IF A$="M" THEN GOTO 6200
  6290 GOTO 6270
  7000 REM PRINT PROGRAM
* 7010 PRINT CHR$(147)
  7020 PRINT"DO YOU WANT THE LISTING ON THE SCREEN O
       R";
  7030 INPUT"ON THE PRINTER (S/P)";S$
  7040 IF S$="P" THEN 7180
  7050 GOSUB 10010
  7060 PRINT
  7070 PRINT"TEACH TABLE"
  7080 PRINT
  7090 PRINT"NR. BODY  UPPERARM  FOREARM    GRIPPER"
  7100 FOR I=0 TO IMAX
  7110 PRINT I;TAB(3);DR(I);TAB(12);OA(I);TAB(21);UA
       (I);TAB(31);ZA$(I)
  7120 NEXT I
  7130 PRINT
  7140 PRINT"> M < MENU"
* 7150 GET A$
  7160 IF A$="M" THEN GOTO 1020
  7170 GOTO7150
* 7180 OPEN 4,4,0
* 7190 PRINT#4,"F I S C H E R T E C H N I K"
* 7200 PRINT#4
* 7210 PRINT#4,"    C O M P U T I N G"
* 7220 PRINT#4
* 7230 PRINT#4,"TRAINING ROBOT WITH 3 AXES "
* 7240 PRINT#4
* 7250 PRINT#4,"TEACH TABLE"
* 7260 PRINT#4,"             "
* 7270 PRINT#4," NR.          BODY      UPPERARM
       LOWERARM       GRIPPER"
  7280 FOR I=0 TO IMAX
* 7290 PRINT#4,I,DR(I),OA(I),UA(I),ZA$(I)
  7300 NEXT I
* 7310 CLOSE 4
  7320 GOTO 1020
  8000 REM END OF PROGRAM
* 8010 PRINT CHR$(147)
  8020 FOR I=1 TO 12
  8030 PRINT
  8040 NEXT
  8050 INPUT" ARE YOU SURE (Y/N)";L$
  8060 IF L$="N" THEN 1020
  8070 IF L$<>"Y" THEN GOTO 8050
* 8080 PRINT CHR$(147)
  8090 END
  9000 REM DIRECTORY
* 9010 PRINT CHR$(147)
  9020 PRINT"FISCHERTECHNIK"
  9030 PRINT"COMPUTING"
  9040 PRINT
* 9050 OPEN1,8,0,"$0"
* 9060 GET#1,A$,B$
* 9070 GET#1,A$,B$
* 9080 GET#1,A$,B$
* 9090 C=0 : C$=""
* 9100 IF A$<>"" THEN C=ASC(A$)
* 9110 IF B$<>"" THEN C=C+ASC(B$)*256
* 9120 PRINT MID$(STR$(C),2);TAB(3);
* 9130 GET#1,B$:IF ST<>0 THEN 9190
* 9140 IF B$<>CHR$(34) THEN 9130
* 9150 GET#1,B$:IF B$<>CHR$(34) THEN C$=C$+B$:GOTO 9
       150
* 9160 GET#1,B$:IF B$<>"" THEN 9160
* 9170 PRINT C$
* 9180 IF ST=0 THEN 9070
  9190 PRINT" BLOCKS FREE"
* 9200 CLOSE 1
  9210 PRINT
  9220 PRINT"  > M <  MENU"
* 9230 GET Z$
  9240 IF Z$<>"M" THEN 9230
* 9250 PRINT CHR$(147)
  9260 GOTO 1020
  10000 REM TITLE SCREEN
* 10010 PRINT CHR$(147)
* 10020 PRINT CHR$(28)"      F I S C H E R"CHR$(31)"
       T E C H N I K"CHR$(144)
  10030 PRINT
  10040 PRINT"           C O M P U T I N G"
  10050 PRINT:PRINT
  10060 PRINT    "      T R A I N I N G  R O B O T"
  10070 PRINT
  10080 PRINT"     W I T H  T H R E E  A X E S  "
  10090 PRINT
  10100 PRINT"           TEACH IN METHOD    "
  10110 RETURN
  11000 REM MOVE TO HOME POSITION
  11010 LET H1=1:H2=1:H3=1:H4=1
* 11020 IF USR(E1)=1 AND H1=1 THEN SYS M1,CW
* 11030 IF USR(E3)=1 AND H2=1 THEN SYS M2,CW
* 11040 IF USR(E5)=1 AND H3=1 THEN SYS M3,CW
* 11050 IF USR(E7)=1 AND H4=1 THEN SYS M4,CW
* 11060 IF USR(E1)=0 THEN SYS M1,CCW:H1=-1
* 11070 IF USR(E3)=0 THEN SYS M2,CCW:H2=-1
* 11080 IF USR(E5)=0 THEN SYS M3,CCW:H3=-1
* 11090 IF USR(E7)=0 THEN SYS M4,OFF:H4=0
* 11100 IF USR(E1)=1 AND H1=-1 THEN SYS M1,OFF:H1=0
* 11110 IF USR(E3)=1 AND H2=-1 THEN SYS M2,OFF:H2=0
* 11120 IF USR(E5)=1 AND H3=-1 THEN SYS M3,OFF:H3=0
  11130 IF H1<>0 OR H2<>0 OR H3<>0 OR H4<>0 THEN GOT
       O 11020
* 11140 SYS INIT
  11150 LET ZA$="OPEN"
  11160 RETURN
  12000 REM PRINT ACTUAL POSITION
* 12010 PRINT CHR$(19);
  12020 FOR I=1 TO 23
* 12030 PRINT CHR$(17);
  12040 NEXT
  12050 PRINT"
        "
* 12060 PRINT CHR$(145);
  12070 PRINT ID;TAB(3);DR(ID);TAB(12);OA(ID);TAB(21
       );UA(ID);TAB(31);ZA$(ID)
  12080 RETURN
```

Operation of the Interface and the Robot System 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.

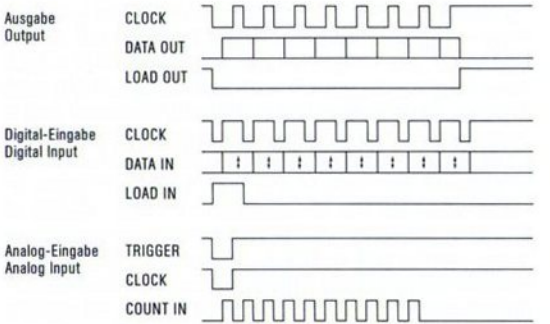
Whether the data are fed to the power amplifiers, however, depends on the enabling control of the memory module. The enabling circuit is controlled by a monostable multivibrator. This circuit generates an enabling signal with a duration of half a second if there is a pulse on the CLOCK line. We may assume 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. Then 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.

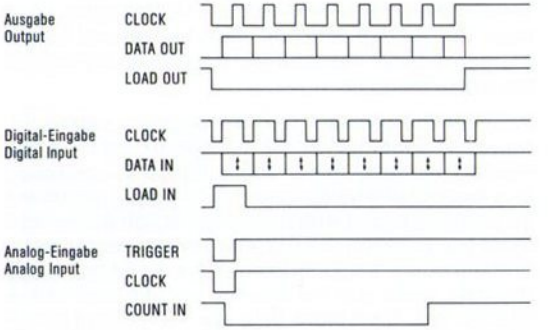
That leaves the analog inputs EX and EY. Potentiometers or other variable resistors are used as the timing element in two additional monostable multivibrator circuits. A low resistance value is converted to a short pulse, a high ressistance value to a pulse with a long duration. The pulse itself is triggered by the starting signals TRIGGER-X and TRIGGER-Y (with negative logic), respectively, and then appears on the COUNT IN line. A machine language programm 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 or the resistance of the potentiometer. The clock rate of the processor, however is involved. Nevertheless, 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.

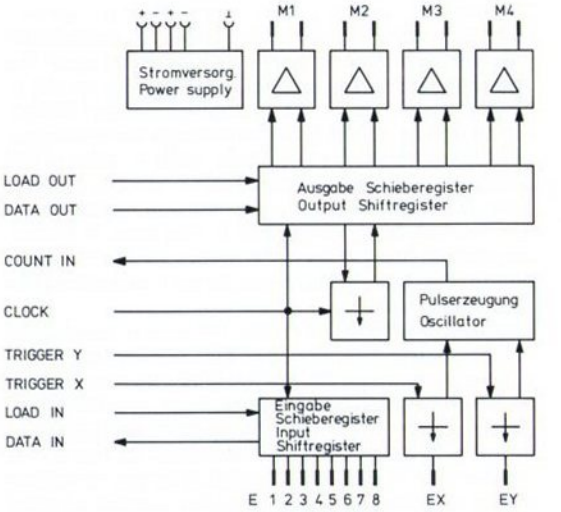
On the following pages is listed the source-code of the robot system program. As detailed on the previous pages of this manual, there is no need to understand the inner construction of the robot system program in order to program the training robot. However you might be interested to continue in developing hard- and software of the trainingrobot, adopt it to different computers etc. and therefore like to browse in the source-code. Due to limitation of space we cannot list all versions of the robot system program fitting to all different computers. In addition, deviations are mostly insignificant. As an example for Computers with microprocessors of the 6502 family we list here the robot system program for the Commodore 64. Another widely used microprocessor is the Z80. The robot system program of the Amstrad CPC stands as an example for all those computers.



Impulsdiagramme des fischertechnik Interface
Pulse Diagrams of the fischertechnik Interface
C 64, VIC 20, Acorn



Impulsdiagramme des fischertechnik Interface
Pulse Diagrams of the fischertechnik Interface
IBM-PC, CPC, CBM 4/8 xxx



Prog. ROBOT.SYS (6502)

```
                0010            ;C64 INTERFACE DRIVER
                0020            ;COPYRIGHT (C) ARTUR FISCHER FORSCHUNG 1984
                0030            ;VERSION 9 INCLUDING ROBOT CONTROL
                0040            ;FILE C64IF9A
                0050            ;MAY 1985
                0060            ;
                0070            ;CONTROL OF THE FISCHERTECHNIK
                0080            ;INTERFACE USING COMMANDS:
                0090            ;SYS M1,CW      SYS M1,CCW
                0100            ;SYS M1,OFF
                0110            ;USR(E1)    USR(EX)    USR(EY)
                0120            ;SPECIAL ROBOT COMMANDS:
                0130            ;SYS P1,NNNN - SET NOMINAL POSITION
                0140            ;USR(P1) - INTERROGATE ACTUAL POSITION
                0150            ;SYS ROBOT - START ROBOT CONTROL
                0160            ;***************************************
                0170            .OS                 ;GENERATE OBJECT CODE
                0180            .BA $CD80           ;PROGRAM IN FREE MEMORY
                0190            ;***************************************
                0200 YFAC       .DE $B3A2           ;CONVERT Y TO FLOATING
                0210 AYFAC      .DE $B391           ;CONVERT A/Y TO FLOATING
                0220 CKCOM      .DE $AEFD           ;CHECK FOR COMMA
                0230 F16INT     .DE $B7F7           ;CONVERT FLOATING TO INT.
                0240 GETBYTE    .DE $B79E           ;EVALUATE BYTE EXPR.
                0250 GETWORD    .DE $AD8A           ;EVALUATE WORD EXPR.
                0260 ROPOS      .DE $CE98           ;CONTROL ROBOT (PART B)
                0270 ROPOSL     .DE $CFD0           ;ROBOT TABLE (PART B)
                0280            ;***************************************
                0290            ; INPUT / OUTPUT REGISTER
                0300            ;***************************************
                0310 UP         .DE $DD01           ;USER PORT DATA REGISTER
                0320 DRR        .DE $DD03           ;USER PORT DATA DIRECTION
                0330 TIL        .DE $DD04           ;TIMER LOW
                0340 TIH        .DE $DD05           ;TIMER HIGH
                0350 TIC        .DE $DD0E           ;TIMER CTRL REG
                0360            ;***************************************
                0370            ; VARIABLES
                0380            ;***************************************
CDB0- 00        0390 AVAR       .BY $00             ;OUTPUT VARIABLE
CDB1- 00        0400 MASK       .BY $00             ;MASK VARIABLE
                0410            ;***************************************
                0420            ; JUMP DISPATCHER
CDB2- A9 00     0430 INIT       LDA #$00            ;INITIALIZE
CDB4- A2 17     0440            LDX #$17            ;TABLE POINTER
CDB6- 9D D0 CF  0450 CLOOP      STA ROPOSL,X        ;ERASE TABLE
CDB9- CA        0460            DEX                 ;OF ROBOT POSITIONS
CDBA- 10 FA     0470            BPL CLOOP
CDBC- 30 2E     0480            BMI STVAR           ;BRANCH ALWAYS
CDBE- A9 03     0490 M1         LDA #%00000011      ;MOTOR 1
CDC0- D0 0A     0500            BNE BOUT            ;BRANCH ALWAYS
CDC2- A9 0C     0510 M2         LDA #%00001100      ;MOTOR 2
CDC4- D0 06     0520            BNE BOUT            ;BRANCH ALWAYS
CDC6- A9 30     0530 M3         LDA #%00110000      ;MOTOR 3
CDC8- D0 02     0540            BNE BOUT            ;BRANCH ALWAYS
CDCA- A9 C0     0550 M4         LDA #%11000000      ;MOTOR 4
                0560            ;***************************************
CDCC- 78        0570 BOUT       SEI                 ;DISABLE INTERRUPT
CDCD- 8D B1 CD  0580            STA MASK            ;STOR BIT MASK
CDD0- 20 FD AE  0590            JSR CKCOM           ;CHECK FOR COMMA
CDD3- AD B0 CD  0600            LDA AVAR            ;OUTPUT VARIABLE
CDD6- 0D B1 CD  0610            ORA MASK            ;SET BIT
CDD9- 8D B0 CD  0620            STA AVAR            ;STORE
CDDC- 20 9E B7  0630            JSR GETBYTE         ;GET 2ND ARGUMENT
CDDF- 8A        0640            TXA
CDE0- 2D B1 CD  0650            AND MASK            ;MASK MOTOR
CDE3- 8D B1 CD  0660            STA MASK            ;STORE
CDE6- AD B0 CD  0670            LDA AVAR            ;OUTPUT VARIABLE
CDE9- 4D B1 CD  0680            EOR MASK            ;SET BIT
CDEC- 20 F1 CD  0690 STVAR      JSR SHOUT           ;OUTPUT TO INTERFACE
CDEF- 58        0700            CLI                 ;ENABLE INTERRUPT
CDF0- 60        0710            RTS                 ;RETURN TO BASIC
                0720            ;***************************************
                0730            ;ROUTINE FOR INTERFACE CONTROL
                0740            ;OUTPUT
                0750            ;OUTPUT BIT PATTERN IN ACCUMULATOR
                0760            ;SIDE EFFECT: STORES ACC IN AVAR
                0770            ;USES ACC AND X-REG
                0780            ;***************************************
CDF1- 8D B0 CD  0790 SHOUT      STA AVAR            ;STORE BIT PATTERN
CDF4- 48        0800            PHA                 ;SAVE ACC
CDF5- A9 3F     0810            LDA #%3F            ;SET DATA DIRECTION
CDF7- 8D 03 DD  0820            STA DRR
CDFA- A2 08     0830            LDX #$08            ;LOOP COUNTER
CDFC- A9 30     0840 LOOP       LDA #%30            ;STATIC BIT PATTERN
CDFE- 0E B0 CD  0850            ASL AVAR            ;TEST OUTPUT BYTE
CE01- 90 02     0860            BCC NULL
CE03- 09 04     0870            ORA #%04            ;SET DATA OUT
CE05- 8D 01 DD  0880 NULL       STA UP              ;OUTPUT
CE08- 09 08     0890            ORA #%08            ;SET CLOCK
CE0A- 8D 01 DD  0900            STA UP              ;OUTPUT
CE0D- CA        0910            DEX
CE0E- D0 EC     0920            BNE LOOP            ;END OF LOOP
CE10- A9 39     0930            LDA #%39            ;SET LOAD OUT
CE12- 8D 01 DD  0940            STA UP              ;OUTPUT
CE15- 68        0950            PLA                 ;RESTORE ACC
CE16- 8D B0 CD  0960            STA AVAR            ;RESTORE AVAR
CE19- 60        0970            RTS                 ;RETURN
                0980            ;***************************************
                0990            ;INPUT ROUTINE
                1000            ;CALLED BY USR
                1010            ;***************************************
CE1A- 78        1020 BINP       SEI                 ;DISABLE INTERRUPT
CE1B- 20 F7 B7  1030            JSR F16INT          ;CONVERT TO INTEGER
CE1E- C9 00     1040            CMP #00             ;HIGH BYTE SET?
CE20- D0 76     1050            BNE ROPOS           ;GET ROBOT POSITION
CE22- C0 A2     1060            CPY #$A2            ;ANALOG INPUT?
CE24- F0 39     1070            BEQ POTS            ;X
CE26- C0 92     1080            CPY #$92            ;ANALOG INPUT?
CE28- F0 35     1090            BEQ POTS            ;Y
CE2A- 8C B1 CD  1100            STY MASK            ;STORE INPUT MASK
CE2D- 20 3D CE  1110            JSR SHIN            ;DIGITAL INPUT ROUTINE
CE30- 2D B1 CD  1120            AND MASK            ;GET INDIVIDUAL BIT
CE33- A8        1130            TAY                 ;TRANSFER TO Y-REG.
CE34- F0 02     1140            BEQ CVAR
```

```
CE36- A0 01     1150          LDY #$01            ;<>0 -> 1
CE38- 20 A2 B3  1160 CVAR     JSR YFAC            ;CONVERT Y TO FAC
CE3B- 58        1170          CLI                 ;ENABLE INTERRUPT
CE3C- 60        1180          RTS                 ;RETURN TO BASIC
                1190          ;*****************************************
                1200          ;INTERFACE CONTROL
                1210          ;INPUT
                1220          ;USES ACCUMULATOR, X-REG. AND Y-REG.
                1230          ;VALUE IN ACC ON RETURN
                1240          ;*****************************************
CE3D- A9 32     1250 SHIN     LDA #$32            ;SET LOAD IN
CE3F- 8D 01 DD  1260          STA UP              ;OUTPUT
CE42- 09 08     1270          ORA #$08            ;SET CLOCK
CE44- 8D 01 DD  1280          STA UP              ;OUTPUT
CE47- A2 08     1290          LDX #8              ;LOOP COUNTER
CE49- 0A        1300 LOOP2    ASL A               ;SHIFT ACC TO LEFT
CE4A- 2C 01 DD  1310          BIT UP              ;TEST USERPORT BIT 7
CE4D- 10 02     1320          BPL NULL2
CE4F- 09 01     1330          ORA #$01            ;SET BIT
CE51- A0 30     1340 NULL2    LDY #$30            ;SET CLOCK
CE53- 8C 01 DD  1350          STY UP              ;OUTPUT
CE56- A0 38     1360          LDY #$38            ;SET CLOCK
CE58- 8C 01 DD  1370          STY UP              ;OUTPUT
CE5B- CA        1380          DEX
CE5C- D0 EB     1390          BNE LOOP2           ;END OF LOOP
CE5E- 60        1400          RTS                 ;RETURN
                1410          ;*****************************************
                1420          ;ANALOG INPUT
                1430          ;BRANCHES HERE IF THE ARGUMENT
                1440          ;OF USR IS EITHER $92 OR $A2
                1450          ;*****************************************
CE5F- A9 FF     1460 POTS     LDA #$FF            ;SET COUNTER REG. TO $FFFF
CE61- 8D 04 DD  1470          STA TIL
CE64- 8D 05 DD  1480          STA TIH
CE67- A9 B9     1490          LDA #$B9            ;SET TIMER CTRL REG
CE69- 8D 0E DD  1500          STA TIC
CE6C- 8C 01 DD  1510          STY UP              ;TRIGGER ONE-SHOT
CE6F- A0 3A     1520          LDY #$3A            ;REMOVE TRIGGER
CE71- 8C 01 DD  1530          STY UP              ;OUTPUT
CE74- AD 04 DD  1540 TST      LDA TIL             ;GET COUNTER LOW BYTE
CE77- A2 03     1550          LDX #$03            ;DELAY LOOP
CE79- CA        1560 DELAY    DEX
CE7A- D0 FD     1570          BNE DELAY
CE7C- 38        1580          SEC                 ;SUBTRACT
CE7D- ED 04 DD  1590          SBC TIL             ;COUNTER STILL RUNNING?
CE80- D0 F2     1600          BNE TST
CE82- A2 38     1610          LDX #$38            ;SET CLOCK, RESET LOAD IN
CE84- 8E 01 DD  1620          STX UP              ;AUSGABE
CE87- 38        1630          SEC                 ;PREPARE SUBTRACTION
CE88- A9 FF     1640          LDA #$FF
CE8A- ED 04 DD  1650          SBC TIL             ;GET COUNT DIFFERENCE
CE8D- A8        1660          TAY
CE8E- A9 FF     1670          LDA #$FF
CE90- ED 05 DD  1680          SBC TIH             ;HIGH BYTE
CE93- 20 91 B3  1690          JSR AYFAC           ;CONVERT A/Y TO FAC
CE96- 58        1700          CLI                 ;ENABLE INTERRUPT
CE97- 60        1710          RTS                 ;RETURN TO BASIC
                1720          .EN
```

--- LABEL FILE: ---

```
AVAR =CDB0          AYFAC =B391         BINP =CE1A
BOUT =CDCC          CKCOM =AEFD         CLOOP =CDB6
CVAR =CE38          DELAY =CE79         DRR =DD03
F16INT =B7F7        GETBYTE =B79E       GETWORD =AD8A
INIT =CDB2          LOOP =CDFC          LOOP2 =CE49
M1 =CDBE            M2 =CDC2            M3 =CDC6
M4 =CDCA            MASK =CDB1          NULL =CE05
NULL2 =CE51         POTS =CE5F          ROPOS =CE98
ROPOSL =CFD0        SHIN =CE3D          SHOUT =CDF1
STVAR =CDEC         TIC =DD0E           TIH =DD05
TIL =DD04           TST =CE74           UP =DD01
YFAC =B3A2
//0000,CE98,CE98
```

```
                0020          ;ROBOT ROUTINES
                0030          ;
                0040          ;FILE C64IF9B
                0050          ;COPYRIGHT ARTUR FISCHER FORSCHUNG
                0060          ;MAY 1985
                0070          ;*****************************************
                0080          ;ADDRESSES
                0090          ;*****************************************
                0100 SHOUT    .DE $CDF1           ;INTERFACE OUT (PART A)
                0110 SHIN     .DE $CE3D           ;INTERFACE IN (PART A)
                0120 AYFAC    .DE $B391           ;CONVERT A/Y TO FLOATING
                0130 CKCOM    .DE $AEFD           ;CHECK FOR COMMA
                0140 GETWORD  .DE $AD8A           ;EVALUATE WORD EXPR.
                0150 F16INT   .DE $B7F7           ;CONVERT FLOATING TO INT.
                0160 AVAR     .DE $CDB0           ;OUTPUT VAR. (PART A)
                0170 MASK     .DE $CDB1           ;MASK VAR. (PART A)
                0180          ;*****************************
                0190          ;PART B IS CONTIGEOUS TO PART A
                0200          ;*****************************
                0210          .OS                 ;GENERATE OBJECT CODE
                0220          .BA $CE98           ;START ADDRESS
                0230          ;*****************************
                0240          ;ROBOT ROUTINE
                0250          ;
                0260          ;INTERROGATE ACTUAL POSITION
                0270          ;*****************************
CE98- A2 00     0280 ROPOS    LDX #$00            ;RESET POINTER
CE9A- C0 C2     0290          CPY #L,P1           ;WHICH POSITION
CE9C- F0 12     0300          BEQ LOPOS
CE9E- A2 02     0310          LDX #$02            ;POINTER=2
```

```
CEA0- C0 C6     0320            CPY #L,P2
CEA2- F0 0C     0330            BEQ LOPOS
CEA4- A2 04     0340            LDX #$04            ;POINTER=4
CEA6- C0 CA     0350            CPY #L,P3
CEA8- F0 06     0360            BEQ LOPOS
CEAA- A2 06     0370            LDX #$06            ;POINTER=6
CEAC- C0 CE     0380            CPY #L,P4
CEAE- D0 0B     0390            BNE SYNTAX
CEB0- BC D0 CF  0400 LOPOS      LDY ROPOSL,X        ;GET LOW BYTE
CEB3- BD D1 CF  0410            LDA ROPOSH,X        ;GET HIGH BYTE
CEB6- 20 91 B3  0420            JSR AYFAC           ;CONVERT TO FLOATING
CEB9- 58        0430            CLI                 ;ENABLE INTERRUPT
CEBA- 60        0440            RTS                 ;RETURN TO BASIC
CEBB- 8C B0 CD  0450 SYNTAX     STY AVAR
CEBE- 8D B1 CD  0460            STA MASK
CEC1- 60        0470            RTS
                0480            ;*****************************
                0490            ;ROBOT ROUTINE
                0500            ;
                0510            ;SET NOMINAL POSTION OF ROBOT
                0520            ;*****************************
CEC2- A9 00     0530 P1         LDA #$00            ;RESET POINTER
CEC4- F0 0A     0540            BEQ STOPOS
CEC6- A9 02     0550 P2         LDA #$02            ;POINTER=2
CEC8- D0 06     0560            BNE STOPOS
CECA- A9 04     0570 P3         LDA #$04            ;POINTER=4
CECC- D0 02     0580            BNE STOPOS
CECE- A9 06     0590 P4         LDA #$06            ;POINTER=6
CED0- 8D B1 CD  0600 STOPOS     STA MASK            ;SAVE POINTER
CED3- 20 FD AE  0610            JSR CKCOM           ;CHECK FOR COMMA
CED6- 20 8A AD  0620            JSR GETWORD         ;GET 2ND ARGUMENT
CED9- 20 F7 B7  0630            JSR F16INT          ;CONVERT TO INTEGER
CEDC- AE B1 CD  0640            LDX MASK            ;GET POINTER
CEDF- 9D D9 CF  0650            STA ROSOLH,X        ;STORE HIGH BYTE
CEE2- 98        0660            TYA
CEE3- 9D D8 CF  0670            STA ROSOLL,X        ;STORE LOW BYTE
CEE6- 60        0680            RTS                 ;RETURN TO BASIC
                0690            ;*****************************
                0700            ;ROBOT CONTROL
                0710            ;
                0720            ;THIS ROUTINE COMPARES THE VALUES
                0730            ;STORED IN ROSOL WITH THOSE STORED
                0740            ;IN ROPOS. THE DIRECTION OF THE MOTOR
                0750            ;IS EVALUATED FROM THE SIGN OF THE
                0760            ;DIFFERENCE. MOTORS WITH DIFFERENCE
                0770            ;<>0 ARE STARTED AND THE PULSES COUNTING
                0780            ;THE PULSES OF THE PHOTO-INTERRUPTER.
                0790            ;ROPOS IS UPDATED CORRESPONDINGLY.
                0800            ;WHEN ROPOS=ROSOL THE MOTOR IS STOPPED,
                0810            ;STILL CONTINUEING COUNTING THE PULSES
                0820            ;UNTIL ALL AXES HAVE COME TO THEIR
                0830            ;COMPLETE STOP. THEN THE PROGRAM
                0840            ;RETURNS TO BASIC
                0850            ;*********************************
CEE7- A2 06     0860 ROBOT      LDX #6              ;LOOP COUNTER
CEE9- 38        0870 MDIR       SEC                 ;GET DIFFERENCE
CEEA- BD D0 CF  0880            LDA ROPOSL,X        ;ACTUAL VALUE
CEED- FD D8 CF  0890            SBC ROSOLL,X        ;NOMINAL VALUE
CEF0- 8D F0 CF  0900            STA SCRATCL         ;INTERMED. STORE
CEF3- BD D1 CF  0910            LDA ROPOSH,X        ;SAME FOR THE
CEF6- FD D9 CF  0920            SBC ROSOLH,X        ;HIGH BYTE
CEF9- 8D F1 CF  0930            STA SCRATCH
CEFC- 10 05     0940            BPL PLUS
CEFE- BD C9 CF  0950            LDA ML,X            ;DIRECTION CCW
CF01- D0 0B     0960            BNE NPOS            ;BRANCH ALWAYS
CF03- AD F0 CF  0970 PLUS       LDA SCRATCL         ;TEST FOR ZERO
CF06- 0D F1 CF  0980            ORA SCRATCH
CF09- F0 03     0990            BEQ NPOS
CF0B- BD C8 CF  1000            LDA MR,X            ;DIRECTION CW
CF0E- 9D E0 CF  1010 NPOS       STA AV,X            ;STORE DIRECTION
CF11- 9D E8 CF  1020            STA MD,X            ;OF MOTOR
CF14- CA        1030            DEX                 ;LOOP COUNTER
CF15- CA        1040            DEX
CF16- 10 D1     1050            BPL MDIR            ;END OF LOOP
CF18- 20 3D CE  1060            JSR SHIN            ;DIGITAL INPUT
CF1B- 8D F1 CF  1070            STA SCRATCH         ;STORE BIT PATTERN
                1080            ;*****************************
                1090            ;READ DIGITAL INPUTS
                1100            ;*****************************
CF1E- 20 3D CE  1110 DIGIN      JSR SHIN            ;DIGITAL INPUT
CF21- A8        1120            TAY
CF22- 4D F1 CF  1130            EOR SCRATCH         ;DETECT EDGE
CF25- 8D F0 CF  1140            STA SCRATCL         ;AND SAVE IT
CF28- 8C F1 CF  1150            STY SCRATCH         ;STORE LEVELS
CF2B- A9 00     1160            LDA #0
CF2D- 8D B0 CD  1170            STA AVAR            ;RESET OUTPUT VAR.
CF30- A2 06     1180            LDX #6              ;LOOP COUNTER MOTOR
                1190            ;*************************
                1200            ;LOOP FOR ALL MOTORS
                1210            ;*************************
CF32- AD F1 CF  1220 LOOPHEAD   LDA SCRATCH         ;GET LEVELS
                1230            ;*************************
                1240            ;TEST FOR LIMIT SWITCH
                1250            ;*************************
CF35- 3D C9 CF  1260            AND ML,X            ;MASK LIMIT SWITCH
CF38- D0 06     1270            BNE SECTOR          ;LIMIT SW. ACTIV?
CF3A- 9D E0 CF  1280            STA AV,X            ;TURN OFF MOTOR
CF3D- 9D E1 CF  1290            STA NL,X            ;RESET DELAY COUNTER
                1300            ;*************************
                1310            ;TEST FOR EDGES
                1320            ;*************************
CF40- AD F0 CF  1330 SECTOR     LDA SCRATCL         ;GET EDGES
CF43- 3D C8 CF  1340            AND MR,X            ;MASK SECTOR
CF46- F0 4C     1350            BEQ NEXCOMP         ;EDGE DETECTED?
CF48- BD E8 CF  1360            LDA MD,X
CF4B- DD C8 CF  1370            CMP MR,X            ;DIRECTION?
CF4E- D0 14     1380            BNE INCPOS
CF50- 38        1390            SEC
CF51- BD D0 CF  1400            LDA ROPOSL,X        ;ROBOT POSITION -1
CF54- E9 01     1410            SBC #1
CF56- 9D D0 CF  1420            STA ROPOSL,X
CF59- BD D1 CF  1430            LDA ROPOSH,X
CF5C- E9 00     1440            SBC #0
CF5E- 9D D1 CF  1450            STA ROPOSH,X
```

```
CF61- 38        1460            SEC
CF62- B0 11     1470            BCS NEXSK           ;BRANCH ALWAYS
CF64- 18        1480 INCPOS     CLC
CF65- BD D0 CF  1490            LDA ROPOSL,X        ;ROBOT POSITION +1
CF68- 69 01     1500            ADC #1
CF6A- 9D D0 CF  1510            STA ROPOSL,X
CF6D- BD D1 CF  1520            LDA ROPOSH,X
CF70- 69 00     1530            ADC #0
CF72- 9D D1 CF  1540            STA ROPOSH,X
CF75- BD D0 CF  1550 NEXSK      LDA ROPOSL,X        ;COMPARE TO NOMINAL POS.
CF78- DD D8 CF  1560            CMP ROSOLL,X
CF7B- D0 17     1570            BNE NEXCOMP
CF7D- BD D1 CF  1580            LDA ROPOSH,X
CF80- DD D9 CF  1590            CMP ROSOLH,X
CF83- D0 0F     1600            BNE NEXCOMP
CF85- A9 00     1610            LDA #0              ;TURN OFF MOTOR
CF87- DD E0 CF  1620            CMP AV,X            ;IS IT THE FIRST TIME?
CF8A- F0 08     1630            BEQ NEXCOMP
CF8C- 9D E0 CF  1640            STA AV,X            ;TURN OFF MOTOR
CF8F- A9 FF     1650            LDA #$FF
CF91- 9D E1 CF  1660            STA NL,X            ;SET DELAY COUNTER
CF94- A9 00     1670 NEXCOMP    LDA #0              ;TEST DELAY COUNTER
CF96- DD E1 CF  1680            CMP NL,X
CF99- F0 03     1690            BEQ OUT             ;DO NOT DECREMENT WHEN REA
CF9B- DE E1 CF  1700            DEC NL,X            ;DECR. DELAY COUNTER
CF9E- AD B0 CD  1710 OUT        LDA AVAR            ;SET UP OUTPUT VAR.
CFA1- 1D E0 CF  1720            ORA AV,X            ;FROM AV OF INDIVIDUAL MOT
CFA4- 8D B0 CD  1730            STA AVAR            ;STORE OUTPUT VAR.
CFA7- CA        1740            DEX                 ;DECR. LOOP COUNTER
CFA8- CA        1750            DEX
CFA9- 10 87     1760            BPL LOOPHEAD        ;END OF LOOP
CFAB- AD B0 CD  1770            LDA AVAR            ;OUPUT TO THE INTERFACE
CFAE- 20 F1 CD  1780            JSR SHOUT
CFB1- F0 03     1790            BEQ NLTST           ;TURN ALL MOTORS OFF
CFB3- 4C 1E CF  1800            JMP DIGIN
CFB6- A2 06     1810 NLTST      LDX #$06            ;LOOP COUNTER
CFB8- 1D E1 CF  1820 TSTNL      ORA NL,X            ;TEST ALL DELAY COUNTER
CFBB- CA        1830            DEX                 ;DECR. LOOP COUNTER
CFBC- CA        1840            DEX
CFBD- 10 F9     1850            BPL TSTNL
CFBF- C9 00     1860            CMP #$00            ;ANY COUNTER ACTIVE?
CFC1- F0 03     1870            BEQ END             ;TASK COMPLETED
CFC3- 4C 1E CF  1880            JMP DIGIN           ;NEXT TEST
CFC6- 58        1890 END        CLI                 ;ENABLE INTERRUPT
CFC7- 60        1900            RTS                 ;RETURN TO BASIC
CFC8- 02        1910 MR         .BY %00000010       ;MOTOR1 CW
CFC9- 01        1920 ML         .BY %00000001       ;MOTOR1 CCW
CFCA- 08        1930            .BY %00001000       ;MOTOR2 CW
CFCB- 04        1940            .BY %00000100       ;MOTOR2 CCW
CFCC- 20        1950            .BY %00100000       ;MOTOR3 CW
CFCD- 10        1960            .BY %00010000       ;MOTOR3 CCW
CFCE- 80        1970            .BY %10000000       ;MOTOR4 CW
CFCF- 40        1980            .BY %01000000       ;MOTOR4 CCW
CFD0- 00        1990 ROPOSL     .BY 0
CFD1- 00        2000 ROPOSH     .BY 0
CFD2-           2010            .DS 6
CFD8- 00        2020 ROSOLL     .BY 0
CFD9- 00        2030 ROSOLH     .BY 0
CFDA-           2040            .DS 6
CFE0- 00        2050 AV         .BY 0               ;OUTPUT VARIABLES
CFE1- 00        2060 NL         .BY 0               ;DELAY COUNTER
CFE2-           2070            .DS 6               ;INTERLOCKED TABLE
CFE8- 00        2080 MD         .BY 0               ;MOTOR DIRECTION
CFE9- 00        2090            .BY 0
CFEA-           2100            .DS 6
CFF0- 00        2110 SCRATCL    .BY 0
CFF1- 00        2120 SCRATCH    .BY 0
                2130            .EN
```

--- LABEL FILE: ---

```
AV =CFE0            AVAR =CDB0          AYFAC =B391
CKCOM =AEFD         DIGIN =CF1E         END =CFC6
F16INT =B7F7        GETWORD =AD8A       INCPOS =CF64
LOOPHEAD =CF32      LOPOS =CEB0         MASK =CDB1
MD =CFE8            MDIR =CEE9          ML =CFC9
MR =CFC8            NEXCOMP =CF94       NEXSK =CF75
NL =CFE1            NLTST =CFB6         NPOS =CF0E
OUT =CF9E           P1 =CEC2            P2 =CEC6
P3 =CECA            P4 =CECE            PLUS =CF03
ROBOT =CEE7         ROPOS =CE98         ROPOSH =CFD1
ROPOSL =CFD0        ROSOLH =CFD9        ROSOLL =CFD8
SCRATCH =CFF1       SCRATCL =CFF0       SECTOR =CF40
SHIN =CE3D          SHOUT =CDF1         STOPOS =CED0
SYNTAX =CEB8        TSTNL =CFB8
//0000,CFF2,CFF2
```

Prog. ROBOT.SYS (Z80)

```
Pass 1 errors: 00

                        10 ;Program Amstrad CPC464 Interface Driver
                        20 ;Copyright (C) Artur Fischer Forschung
                        30 ;Version 2 including robot control
                        40 ;File ROSYS.GEN
                        50 ;August 1985
                        60 ;
                        70 ;Control of the fischertechnik Interface
                        80 ;by the CPC using commands:
                        90 ;CALL m1,cw          CALL m1,ccw
                       100 ;CALL m1,off
                       110 ;CALL in,@e1         CALL in,@ex
                       120 ;Instead m1 you may also use m2, m3 or m4
                       130 ;Instead e1 you may also use e2 thru e8
                       140 ;Instead ex you may also use ey
                       150 ;
                       160 ;Special robot commands:
                       170 ;CALL p1,nnnn  deposit nominal position
                       180 ;CALL in,@i1   interrogate actual position
                       190 ;Instead p1 you may also use p2, p3 or p4
                       200 ;Instead i1 you may also use i2, i3 or i4
                       210 ;CALL robot    start robot control
                       220 ;*********************************
A400                   230         org  #a400              ;start of program
                       240 ;*********************************
A400  CDFEA4           250 INIT:   CALL SYNTO              ;syntax check
A403  212CA6           260         LD   HL,ROPOSL          ;table pointer
A406  AF               270         XOR  A                  ;clear accumulator
A407  0617             280         LD   B,#17              ;loop counter
A409  77               290 LOOP1:  LD   (HL),A             ;clear robot tables
A40A  23               300         INC  HL
A40B  10FC             310         DJNZ LOOP1              ;end of loop
A40D  3E00             320         LD   A,#00              ;clear output variable
A40F  181D             330         JR   STVAR
A411  0603             340 M1:     LD   B,#03              ;motor 1
A413  180A             350         JR   BOUT
A415  060C             360 M2:     LD   B,#0C              ;motor 2
A417  1806             370         JR   BOUT
A419  0630             380 M3:     LD   B,#30              ;motor 3
A41B  1802             390         JR   BOUT
A41D  06C0             400 M4:     LD   B,#C0              ;motor 4
A41F  CD04A5           410 BOUT:   CALL SYNT1              ;syntax check
                       420 ;*********************************
                       430 ;Bit output
                       440 ;*********************************
A422  3A0EA5           450         LD   A,(AVAR)           ;output variable
A425  B0               460         OR   B                  ;set both bits
A426  4F               470         LD   C,A
A427  DD7E00           480         LD   A,(IX+0)
A42A  A0               490         AND  B
A42B  47               500         LD   B,A
A42C  79               510         LD   A,C
A42D  A8               520         XOR  B                  ;set sense of rotation
A42E  320EA5           530 STVAR:  LD   (AVAR),A           ;set output variable
A431  CD36A4           540         CALL SHOUT              ;output to the interface
A434  FB               550         EI                      ;enable interrupt
A435  C9               560         RET                     ;back to BASIC
                       570 ;*********************************
                       580 ;Routine for interface control
                       590 ;Output
                       600 ;Output bit pattern in accumulator.
                       610 ;Modifies A, BC, and DE.
                       620 ;*********************************
A436  0100EF           630 SHOUT:  LD   BC,#EF00           ;pointer to printer port
A439  4F               640         LD   C,A                ;use C as work register
A43A  1E08             650         LD   E,#08              ;loop counter
A43C  1630             660 LOOP:   LD   D,#30              ;static bit pattern
A43E  79               670         LD   A,C
A43F  07               680         RLCA                    ;next bit
A440  4F               690         LD   C,A
A441  3002             700         JR   NC,NULL            ;=0?
A443  1634             710         LD   D,#34              ;set DATA OUT
A445  ED51             720 NULL:   OUT  (C),D              ;output
A447  7A               730         LD   A,D
A448  F608             740         OR   #08                ;set CLOCK
A44A  ED79             750         OUT  (C),A              ;Ausgabe
A44C  1D               760         DEC  E                  ;loop counter
A44D  20ED             770         JR   NZ,LOOP            ;end of loop
A44F  1639             780         LD   D,#39              ;set LOAD OUT
A451  ED51             790         OUT  (C),D              ;output
A453  C9               800         RET
                       810 ;*********************************
                       820 ;Input routine
                       830 ;Called by command in,@en
                       840 ;*********************************
A454  CD04A5           850 inp:    CALL SYNT1              ;syntax check
A457  2185AE           860         LD   HL,#AE85           ;variable storage
A45A  4E               870         LD   C,(HL)             ;pointer to variables
A45B  23               880         INC  HL
A45C  46               890         LD   B,(HL)
A45D  210500           900         LD   HL,#0005
A460  09               910         ADD  HL,BC              ;address of E1
A461  DD5601           920         LD   D,(IX+1)
A464  DD5E00           930         LD   E,(IX+0)
A467  010100           940         LD   BC,#0001           ;bit counter
A46A  7C               950 VARTST: LD   A,H                ;compare addresses
A46B  BA               960         CP   D                  ;high byte
A46C  2004             970         JR   NZ,NEXTVAR
A46E  7D               980         LD   A,L
A46F  BB               990         CP   E                  ;low byte
A470  2813            1000         JR   Z,VARFOUND         ;variable found
A472  C5              1010 NEXTVA: PUSH BC
A473  010700          1020         LD   BC,#0007           ;increment address
A476  09              1030         ADD  HL,BC
A477  C1              1040         POP  BC
A478  79              1050         LD   A,C
A479  17              1060         RLA                     ;rotate bit counter
A47A  4F              1070         LD   C,A
A47B  78              1080         LD   A,B                ;high byte
A47C  17              1090         RLA
A47D  47              1100         LD   B,A
A47E  FE40            1110         CP   #40                ;end of variable table
A480  CA0BA5          1120         JP   Z,SYNTAX           ;syntax error
A483  18E5            1130         JR   VARTST             ;continue searching
A485  78              1140 VARFOU: LD   A,B                ;analog input?
```

```
A486  FE04     1150          CP   #04
A488  D20FA5   1160          JP   NC,ROPOS
A48B  FE01     1170          CP   #01
A48D  2844     1180          JR   Z,XPOTI              ;input EX
A48F  FE02     1190          CP   #02
A491  2844     1200          JR   Z,YPOTI              ;input EY
A493  C5       1210          PUSH BC                   ;save BC
A494  CD99A4   1220          CALL SHIN                 ;digital input
A497  182F     1230          JR   CONT
               1240 ;*********************************
               1250 ;Routine for interface control
               1260 ;Input
               1270 ;Modifies A, BC, and DE.
               1280 ;Input value is returned in A.
               1290 ;*********************************
A499  1632     1300 SHIN:    LD   D,#32                ;set LOAD IN
A49B  0100EF   1310          LD   BC,#EF00             ;pointer to printer port
A49E  ED51     1320          OUT  (C),D                ;output
A4A0  163A     1330          LD   D,#3A                ;set CLOCK
A4A2  ED51     1340          OUT  (C),D                ;output
A4A4  1E08     1350          LD   E,#08                ;loop counter
A4A6  17       1360 LOOP2:   RLA                       ;shift data byte
A4A7  E6FE     1370          AND  #FE                  ;clear bit 0
A4A9  0100F5   1380          LD   BC,#F500             ;pointer to BUSY input
A4AC  4F       1390          LD   C,A
A4AD  ED78     1400          IN   A,(C)                ;read BUSY
A4AF  E640     1410          AND  #40                  ;mask BUSY input
A4B1  17       1420          RLA                       ;and shift into carry
A4B2  17       1430          RLA
A4B3  79       1440          LD   A,C
A4B4  3002     1450          JR   NC,NULL2             ;test DATA-IN
A4B6  F601     1460          OR   #01                  ;set Bit 0
A4B8  0100EF   1470 NULL2:   LD   BC,#EF00             ;pointer to printer port
A4BB  1630     1480          LD   D,#30                ;reset CLOCK
A4BD  ED51     1490          OUT  (C),D                ;output
A4BF  1638     1500          LD   D,#38                ;set CLOCK
A4C1  ED51     1510          OUT  (C),D                ;output
A4C3  1D       1520          DEC  E                    ;loop counter
A4C4  20E0     1530          JR   NZ,LOOP2             ;end of loop
A4C6  2F       1540          CPL                       ;negative logic!
A4C7  C9       1550          RET
               1560 ;*********************************
A4C8  C1       1570 CONT:    POP  BC                   ;restore BC
A4C9  A1       1580          AND  C
A4CA  2802     1590          JR   Z,BASRET
A4CC  3E01     1600          LD   A,#01                ;<>0 -> 1
A4CE  77       1610 BASRET:  LD   (HL),A               ;deposit in variable
A4CF  23       1620          INC  HL
A4D0  70       1630          LD   (HL),B
A4D1  FB       1640          EI                        ;enable interrupt
A4D2  C9       1650          RET                       ;zurueck in BASIC
               1660 ;*********************************
               1670 ;Analog input
               1680 ;branches here when EX or EY is
               1690 ;to be tested.
               1700 ;*********************************
A4D3  16A0     1710 XPOTI:   LD   D,#A0                ;set TRIGGER-X
A4D5  1802     1720          JR   POTI

A4D7  1690     1730 YPOTI:   LD   D,#90                ;set TRIGGER-Y
A4D9  0100EF   1740 POTI:    LD   BC,#EF00             ;pointer to printer port
A4DC  ED51     1750          OUT  (C),D                ;output
A4DE  1638     1760          LD   D,#38                ;set CLOCK
A4E0  ED51     1770          OUT  (C),D                ;output
A4E2  0100F5   1780          LD   BC,#F500             ;pointer to BUSY input
A4E5  110000   1790          LD   DE,#0000             ;reset counter
A4E8  ED78     1800 LOOP3:   IN   A,(C)                ;test COUNT IN
A4EA  17       1810          RLA                       ;shift into carry
A4EB  17       1820          RLA                       ;for testing
A4EC  3804     1830          JR   C,STOP               ;end of pulse?
A4EE  1C       1840          INC  E                    ;increment counter
A4EF  20F7     1850          JR   NZ,LOOP3             ;continue testing
A4F1  1D       1860          DEC  E                    ;overflow -> 255
A4F2  73       1870 STOP:    LD   (HL),E               ;deposit in variable
A4F3  23       1880          INC  HL
A4F4  72       1890          LD   (HL),D
A4F5  0100EF   1900          LD   BC,#EF00
A4F8  1638     1910          LD   D,#38                ;set CLOCK
A4FA  ED51     1920          OUT  (C),D                ;output
A4FC  FB       1930          EI                        ;enable interrupt
A4FD  C9       1940          RET                       ;back to BASIC
               1950 ;*********************************
               1960 ;routine for syntax check
               1970 ;*********************************
A4FE  FE00     1980 SYNTO:   CP   #00                  ;CALL INIT, ROBOT 0 arg.
A500  F3       1990          DI                        ;disable interrupt
A501  C8       2000          RET  Z
A502  1804     2010          JR   SYNTAX               ;error message
A504  FE01     2020 SYNT1:   CP   #01                  ;CALL Mn,mode 1 arg.
A506  F3       2030          DI                        ;disable interrupt
A507  C8       2040          RET  Z
A508  CD00B9   2050 SYNTAX:  CALL #B900                ;enable ROM
A50B  C3C6DD   2060          JP   #DDC6                ;print error message
A50E  00       2070 AVAR:    DEFB #00                  ;output variable
               2080 ;*********************************
               2090 ;Robot control routines
               2100 ;
               2110 ;Copyright (C) Artur Fischer Forschung
               2120 ;August 1985
               2130 ;
               2140 ;Interrogate actual robot position.
               2150 ;*********************************
A50F  EB       2160 ROPOS:   EX   DE,HL
A510  212CA6   2170          LD   HL,ROPOSL            ;pointer actual position
A513  1F       2180          RRA
A514  1F       2190          RRA
A515  1F       2200 LOOP4:   RRA
A516  3804     2210          JR   C,CONT1
A518  23       2220          INC  HL                   ;increment pointer
A519  23       2230          INC  HL
A51A  18F9     2240          JR   LOOP4
A51C  7E       2250 CONT1:   LD   A,(HL)               ;load low byte
A51D  12       2260          LD   (DE),A               ;deposit in variable
A51E  23       2270          INC  HL                   ;high byte
A51F  13       2280          INC  DE
A520  7E       2290          LD   A,(HL)
A521  12       2300          LD   (DE),A
```

```
A522  FB        2310          EI                        ;enable interrupt
A523  C9        2320          RET                       ;back to BASIC
                2330 ;*********************************
                2340 ;Robot control routine
                2350 ;
                2360 ;Set nominal robot position.
                2370 ;*********************************
A524  0E00      2380 P1:      LD    C,#00               ;pointer=0
A526  180A      2390          JR    STOPOS
A528  0E02      2400 P2:      LD    C,#02               ;pointer=2
A52A  1806      2410          JR    STOPOS
A52C  0E04      2420 P3:      LD    C,#04               ;pointer=4
A52E  1802      2430          JR    STOPOS
A530  0E06      2440 P4:      LD    C,#06               ;pointer=6
A532  0600      2450 STOPOS:  LD    B,#00
A534  CD04A5    2460          CALL  SYNT1               ;syntax check
A537  2134A6    2470          LD    HL,ROSOLL           ;nominal positions
A53A  09        2480          ADD   HL,BC               ;add pointer
A53B  DD7E00    2490          LD    A,(IX)              ;get 2nd arg. of CALL
A53E  77        2500          LD    (HL),A              ;save it
A53F  23        2510          INC   HL                  ;high byte
A540  DD7E01    2520          LD    A,(IX+#01)
A543  77        2530          LD    (HL),A
A544  FB        2540          EI                        ;enable interrupt
A545  C9        2550          RET                       ;back to BASIC
                2560 ;*********************************
                2570 ;Robot control routine
                2580 ;
                2590 ;This routine compares the values
                2600 ;stored in ROSOL with those stored
                2610 ;in ROPOS. The direction of the motor
                2620 ;is evaluated from the sign of the
                2630 ;difference. Motors with difference
                2640 ;<>0 are started counting the pulses
                2650 ;of the photo-interrupter.
                2660 ;ROPOS is updated correspondingly.
                2670 ;When ROPOS=ROSOL the motor is stopped,
                2680 ;still continueing counting the pulses
                2690 ;until all axes have come to their
                2700 ;complete stop. Then the program
                2710 ;returns to BASIC.
                2720 ;*********************************
A546  0603      2730 ROBOT:   LD    B,#03               ;loop counter
A548  CDFEA4    2740          CALL  SYNTO               ;syntax check
A54B  DD2124A6  2750          LD    IX,MR
A54F  DD6E08    2760 MDIR:    LD    L,(IX+#08)          ;ROPOS (actual pos.)
A552  DD6609    2770          LD    H,(IX+#09)
A555  DD5E10    2780          LD    E,(IX+#10)          ;ROSOL (nominal pos.)
A558  DD5611    2790          LD    D,(IX+#11)
A55B  A7        2800          AND   A                   ;reset carry
A55C  ED52      2810          SBC   HL,DE               ;difference
A55E  DD7528    2820          LD    (IX+#28),L          ;save it (SCRATCH)
A561  DD7429    2830          LD    (IX+#29),H
A564  3005      2840          JR    NC,PLUS             ;difference>0
A566  DD7E01    2850          LD    A,(IX+#01)          ;counter-clockwise
A569  180B      2860          JR    NPOS
A56B  DD7E28    2870 PLUS:    LD    A,(IX+#28)          ;difference=0?
A56E  DDB629    2880          OR    (IX+#29)
A571  2803      2890          JR    Z,NPOS
A573  DD7E00    2900          LD    A,(IX+#00)          ;clockwise
A576  DD7718    2910 NPOS:    LD    (IX+#18),A          ;partial output word AV
A579  DD7720    2920          LD    (IX+#20),A          ;save sense of rotation
A57C  DD23      2930          INC   IX                  ;increment pointer
A57E  DD23      2940          INC   IX
A580  10CD      2950          DJNZ  MDIR                ;end of loop
A582  DD2124A6  2960          LD    IX,MR               ;pointer sense of rot.
A586  214DA6    2970          LD    HL,SCRATC           ;pointer SCRATCH
A589  CD99A4    2980          CALL  SHIN                ;digital input routine
A58C  77        2990          LD    (HL),A              ;save starting bit patt.
                3000 ;*********************************
                3010 ;Read digital inputs
                3020 ;*********************************
A58D  CD99A4    3030 DIGIN:   CALL  SHIN                ;digital input
A590  4E        3040          LD    C,(HL)              ;starting bit pattern
A591  77        3050          LD    (HL),A              ;actual bit pattern
A592  A9        3060          XOR   C                   ;detect any changes
A593  2B        3070          DEC   HL
A594  77        3080          LD    (HL),A
A595  23        3090          INC   HL
A596  AF        3100          XOR   A                   ;clear accumulator
A597  320EA5    3110          LD    (AVAR),A            ;clear output word
A59A  0603      3120          LD    B,#03               ;loop counter
A59C  DD2124A6  3130          LD    IX,MR               ;pointer sense of rot.
                3140 ;*********************************
                3150 ;Loop for all motors
                3160 ;*********************************
A5A0  7E        3170 LOOPH:   LD    A,(HL)              ;begin of loop
                3180 ;*********************************
                3190 ;Test limit switches
                3200 ;*********************************
A5A1  DDA601    3210          AND   (IX+#01)            ;mask limit switches
A5A4  2006      3220          JR    NZ,SECTOR           ;active?
A5A6  DD7718    3230          LD    (IX+#18),A          ;turn off motor
A5A9  DD7719    3240          LD    (IX+#19),A          ;reset delay counter
                3250 ;*********************************
                3260 ;Test for any pulse edges
                3270 ;*********************************
A5AC  2B        3280 SECTOR:  DEC   HL
A5AD  7E        3290          LD    A,(HL)              ;mask pulse input
A5AE  23        3300          INC   HL
A5AF  DDA600    3310          AND   (IX+0)
A5B2  2831      3320          JR    Z,NEXCOM            ;no edge
A5B4  DD5E08    3330          LD    E,(IX+#08)          ;actual pos. (ROPOSL)
A5B7  DD5609    3340          LD    D,(IX+#09)          ;actual pos. (ROPOSH)
A5BA  DD7E20    3350          LD    A,(IX+#20)          ;counter-clockwise?
A5BD  DDBE01    3360          CP    (IX+#01)
A5C0  2803      3370          JR    Z,INCPOS
A5C2  1B        3380          DEC   DE                  ;robot position -1
A5C3  1801      3390          JR    NEXSK
A5C5  13        3400 INCPOS:  INC   DE                  ;robot position +1
A5C6  DD7308    3410 NEXSK:   LD    (IX+#08),E          ;save robot position
A5C9  DD7209    3420          LD    (IX+#09),D
A5CC  7B        3430          LD    A,E                 ;compare actual and nom.
A5CD  DDBE10    3440          CP    (IX+#10)
A5D0  2013      3450          JR    NZ,NEXCOM
A5D2  7A        3460          LD    A,D
```

```
A5D3  DDBE11    3470          CP   (IX+#11)
A5D6  200D      3480          JR   NZ,NEXCOM
A5D8  AF        3490          XOR  A                    ;clear accumulator
A5D9  DDBE18    3500          CP   (IX+#18)             ;diff=0 the first time?
A5DC  2807      3510          JR   Z,NEXCOM
A5DE  DD7718    3520          LD   (IX+#18),A           ;turn off motor
A5E1  3D        3530          DEC  A                    ;accumulator=FF
A5E2  DD7719    3540          LD   (IX+#19),A           ;set delay counter
A5E5  AF        3550 NEXCOM:  XOR  A                    ;clear accumulator
A5E6  DDBE19    3560          CP   (IX+#19)             ;test delay counter
A5E9  2803      3570          JR   Z,OUT                ;do not decr. when =0
A5EB  DD3519    3580          DEC  (IX+#19)             ;decrement delay counter
A5EE  3A0EA5    3590 OUT:     LD   A,(AVAR)             ;set up output word
A5F1  DDB618    3600          OR   (IX+#18)             ;from individual AV
A5F4  320EA5    3610          LD   (AVAR),A             ;save it
A5F7  DD23      3620          INC  IX                   ;increment pointer by 2
A5F9  DD23      3630          INC  IX
A5FB  10A3      3640          DJNZ LOOPH                ;end of loop
A5FD  DD2124A6  3650          LD   IX,MR                ;reset pointer
A601  3A0EA5    3660          LD   A,(AVAR)             ;output variable
A604  F5        3670          PUSH AF                   ;save registers
A605  CD36A4    3680          CALL SHOUT                ;output to the interface
A608  F1        3690          POP  AF                   ;restore registers
A609  FE00      3700          CP   #00                  ;all motors turned off?
A60B  2803      3710          JR   Z,NLTST
A60D  C38DA5    3720          JP   DIGIN                ;no - continue
A610  0603      3730 NLTST:   LD   B,3                  ;loop counter
A612  DDB619    3740 TSTNL:   OR   (IX+#19)             ;test delay counter
A615  DD23      3750          INC  IX                   ;increment pointer by 2
A617  DD23      3760          INC  IX
A619  10F7      3770          DJNZ TSTNL                ;end of loop
A61B  FE00      3780          CP   #00                  ;test all delay counter
A61D  2803      3790          JR   Z,END                ;any counter active?
A61F  C38DA5    3800          JP   DIGIN                ;no - continue testing
A622  FB        3810 END:     EI                        ;enable interrupt
A623  C9        3820          RET                       ;back to BASIC
                3830 ;*********************************
A624  02        3840 MR:      DEFB %00000010            ;motor1 cw
A625  01        3850 ML:      DEFB %00000001            ;motor1 ccw
A626  08        3860          DEFB %00001000            ;motor2 cw
A627  04        3870          DEFB %00000100            ;motor2 ccw
A628  20        3880          DEFB %00100000            ;motor3 cw
A629  10        3890          DEFB %00010000            ;motor3 ccw
A62A  80        3900          DEFB %10000000            ;motor4 cw
A62B  40        3910          DEFB %01000000            ;motor4 ccw
A62C  00        3920 ROPOSL:  DEFB 0                    ;robot actual position
A62D  00        3930 ROPOSH:  DEFB 0
A62E            3940          DEFS 6
A634  00        3950 ROSOLL:  DEFB 0                    ;nominal robot position
A635  00        3960 ROSOLH:  DEFB 0
A636            3970          DEFS 6
A63C  00        3980 AV:      DEFB 0                    ;partial output word
A63D  00        3990 NL:      DEFB 0                    ;delay counter
A63E            4000          DEFS 6                    ;interlaced table
A644  00        4010 MD:      DEFB 0                    ;sense of rotation
A645  00        4020          DEFB 0
A646            4030          DEFS 6
A64C  00        4040          DEFB 0
A64D  00        4050 SCRATC:  DEFB 0                    ;scratch
A64E            4060          DEFS 6
                4070 ;************** end ***************

Pass 2 errors: 00

Table used:   702  from  1383
```

Bebilderte Bauanleitung

Kabelkonfektionierung	62
Verdrahtungsplan Gabellichtschranke	63
Mechanischer Aufbau	64
Verdrahtungsplan Trainingsroboter	91

Illustrated Assembly Instructions

Ribbon Cable Configuration	62
Circuit Layout Photo-Interrupter	63
Mechanical Assembly	64
Circuit Layout Training Robot	91

Kabelkonfektionierung · Ribbon cable configuration · Schéma de câblage

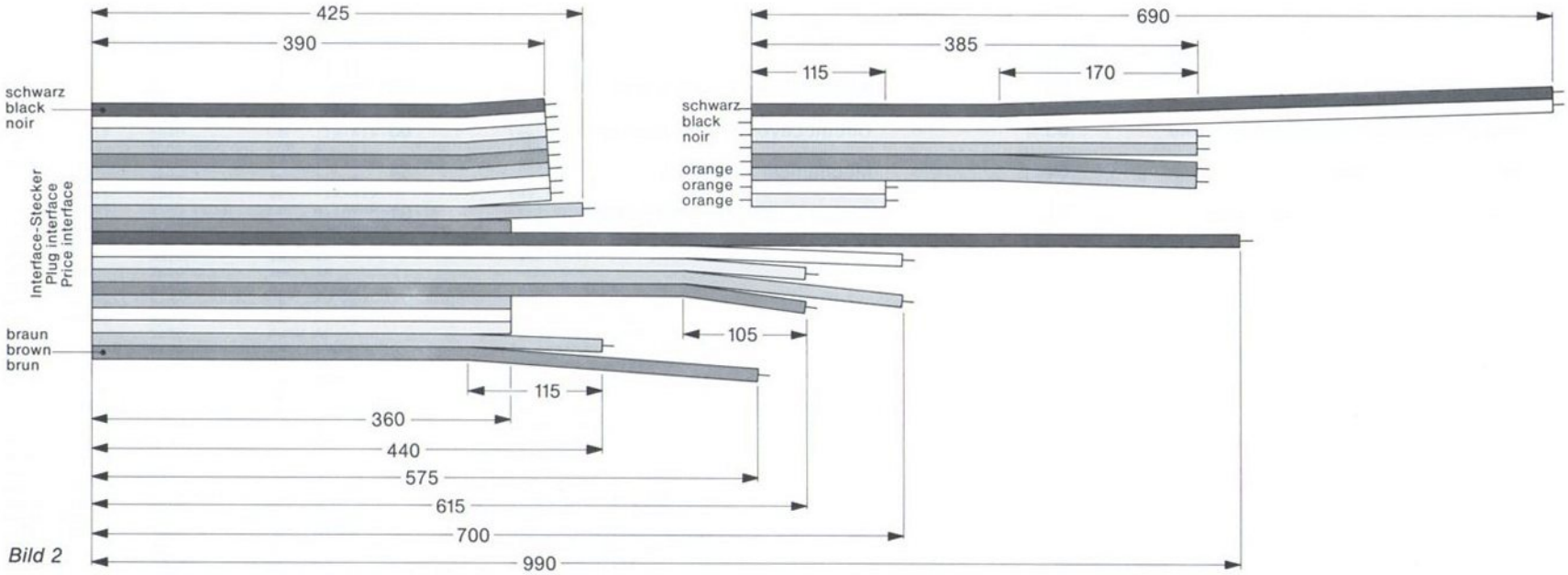


Bild 2

Steckermontage
Plug installation
Assemblage de cables

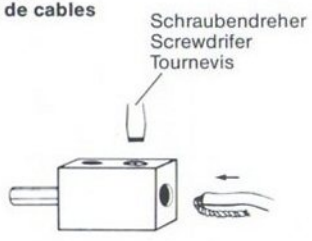


Bild 3

Durchgangsprüfung
Continuity tester
Contrôle du passage

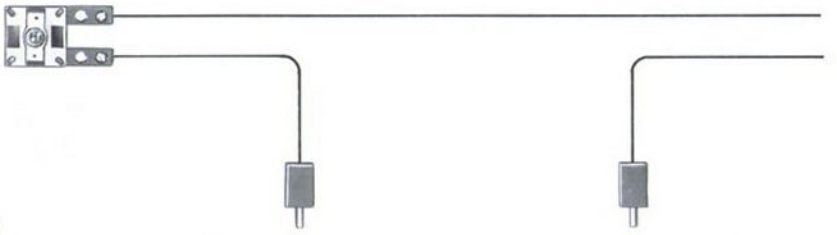
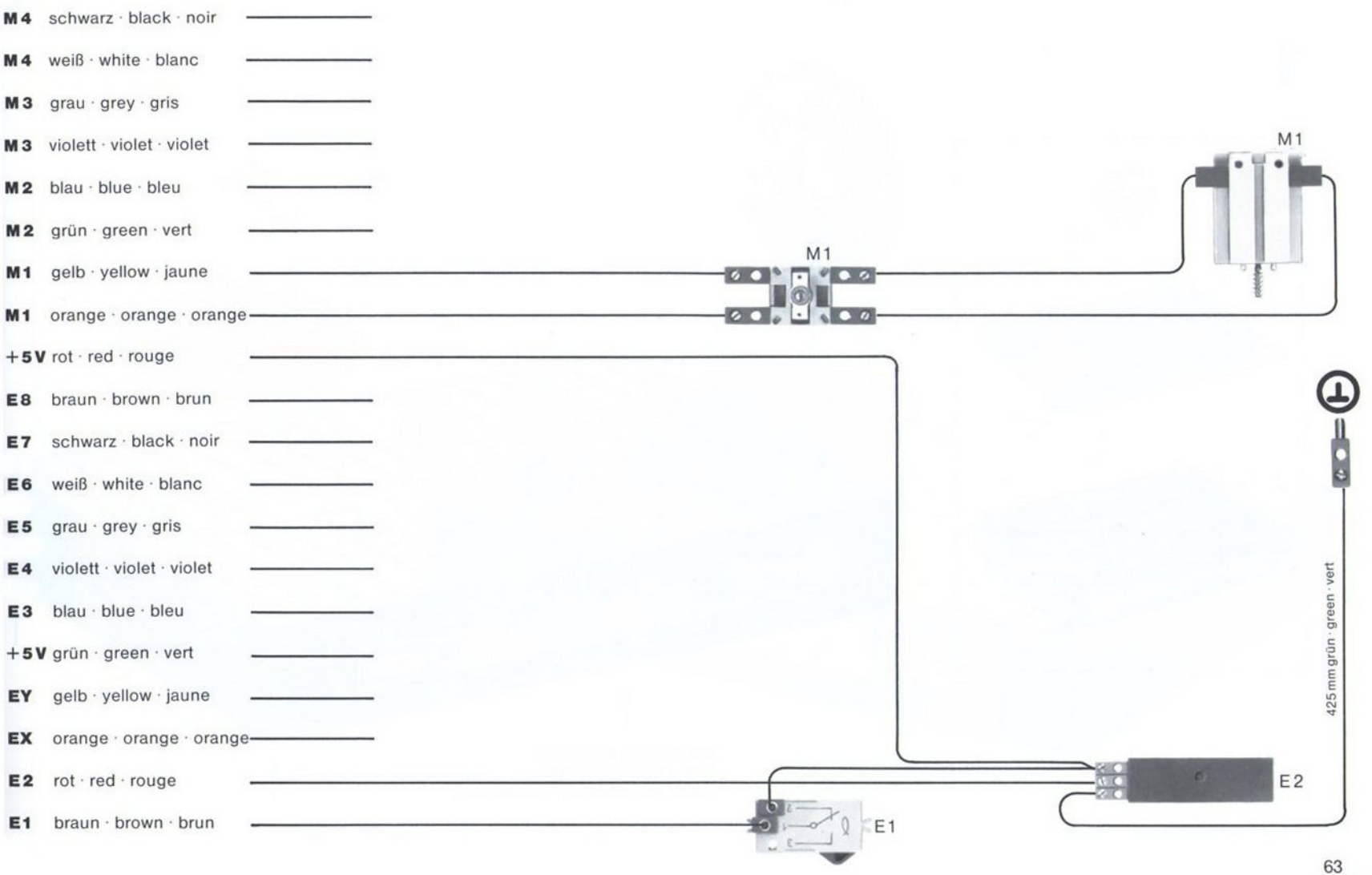


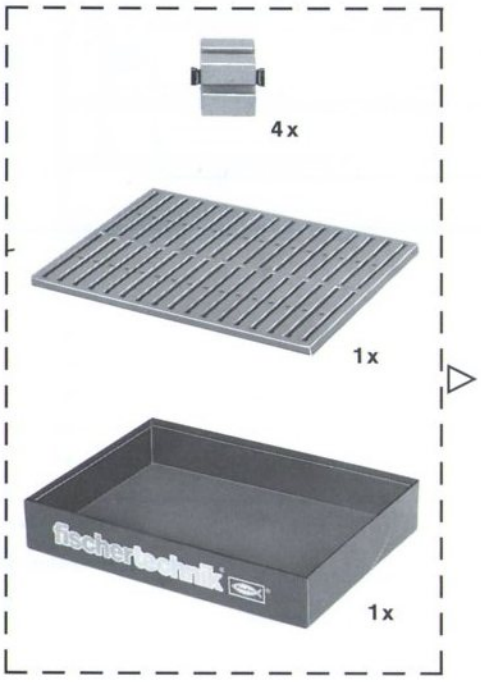
Bild 4

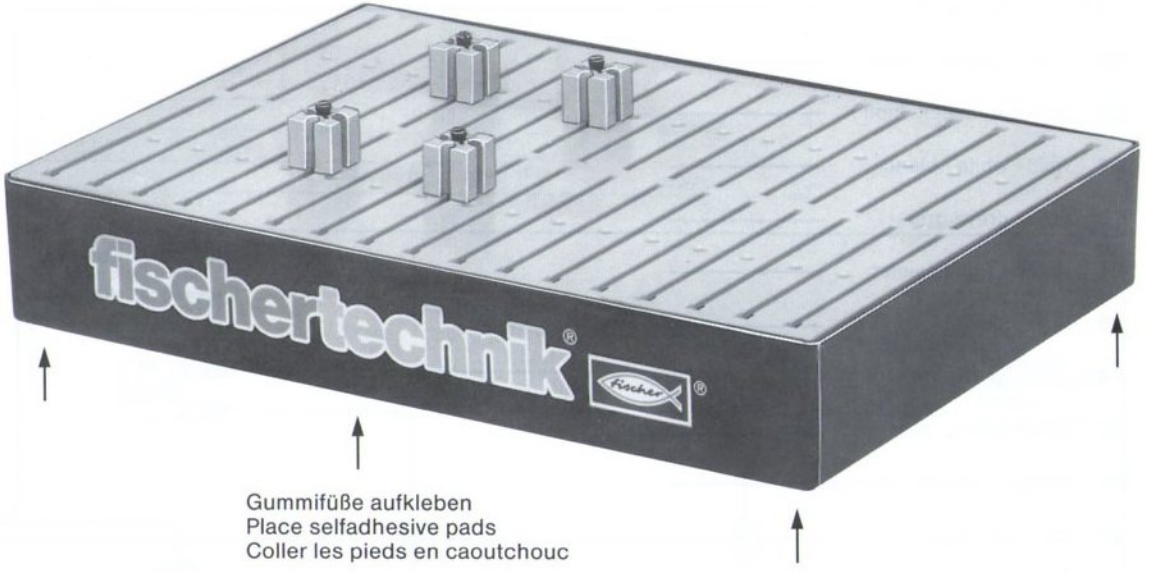
Verdrahtungsplan Gabellichtschranke · Circuit Layout Photo-interrupter · Plan de cablage d'interrupteur photo



Mechanischer Aufbau · Mechanical Assembly · Assemblage des parts mécanique

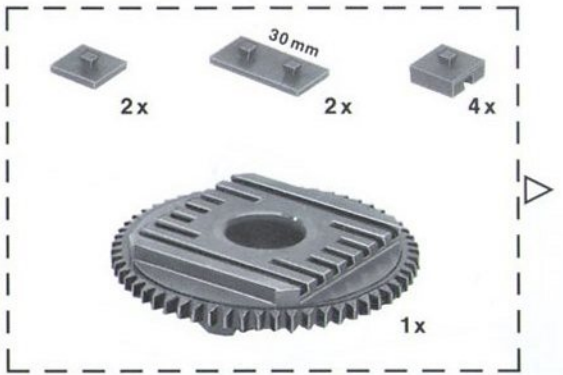
1



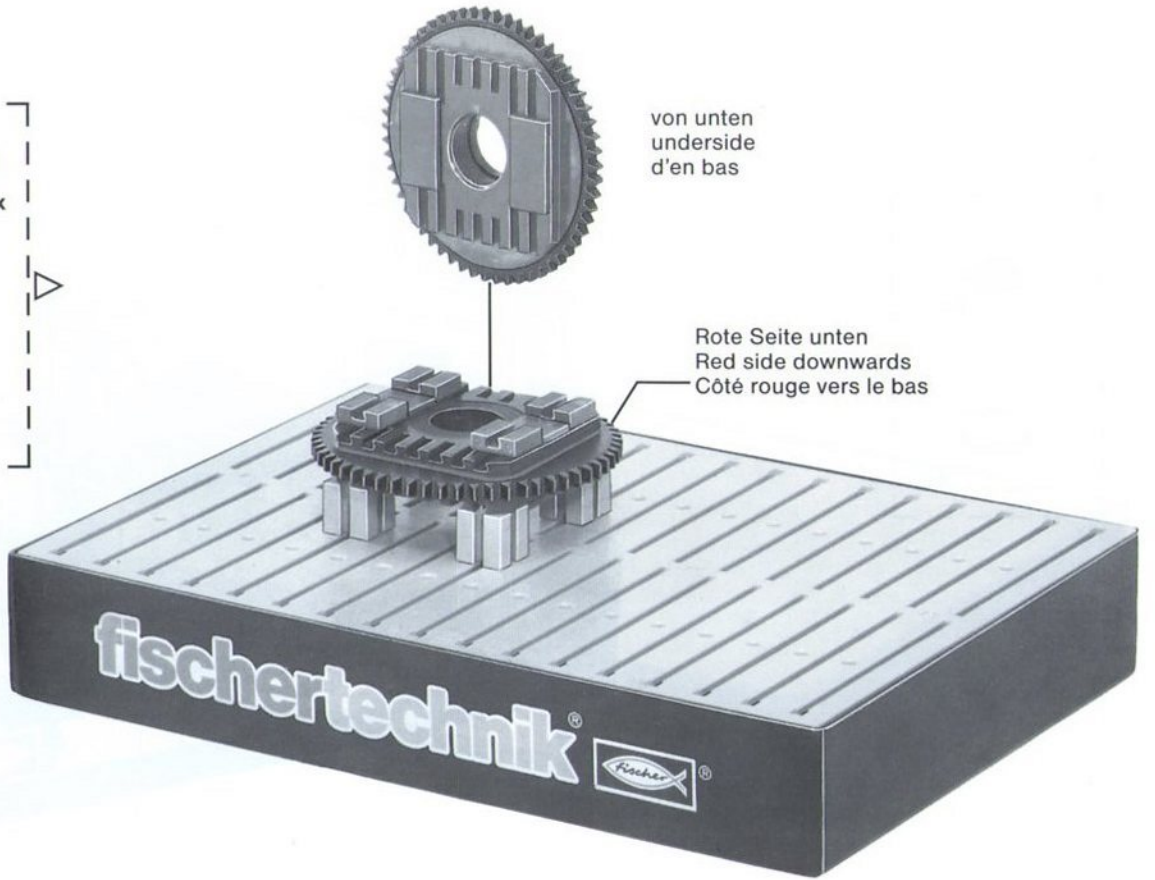


Gummifüße aufkleben
Place selfadhesive pads
Coller les pieds en caoutchouc

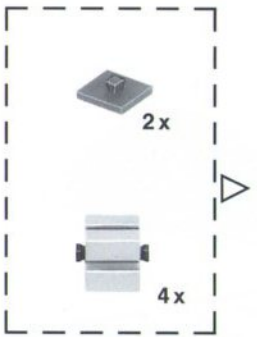
2


30 mm
2 x
2 x
4 x
1 x


0
30 mm


von unten
underside
d'en bas
Rote Seite unten
Red side downwards
Côté rouge vers le bas
fischertechnik

3

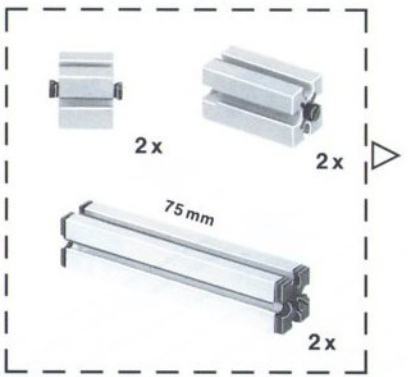


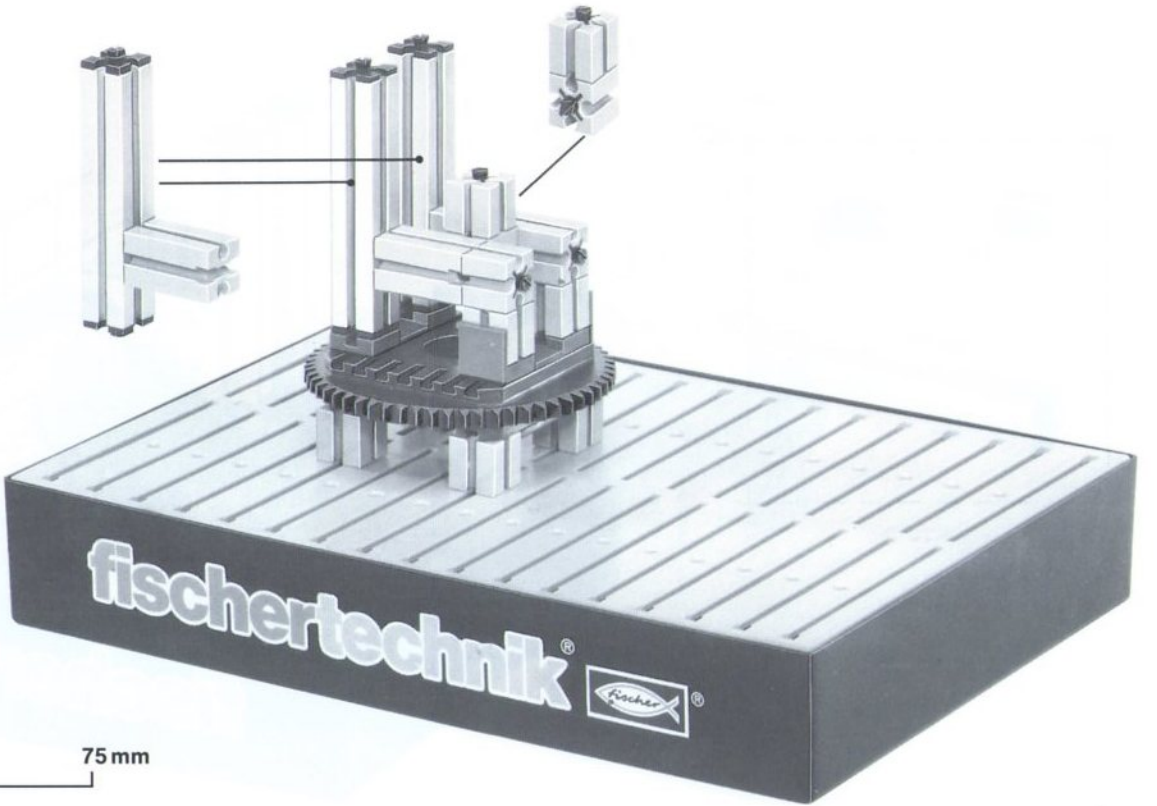
4

Aus technischen Gründen sind die Schiebekräfte der Metallbaustäbe teilweise sehr hoch.

For technical reasons, the shearing forces of the structural metal rods are partially very high.

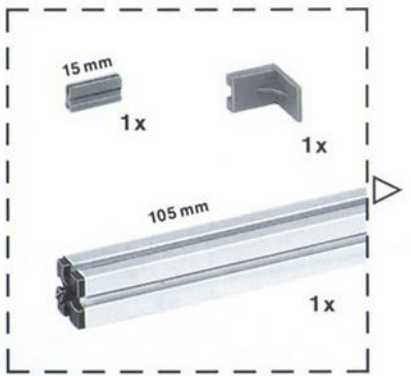
Pour des raisons techniques les forces transversales des barres métalliques de construction sont parfois très importantes.

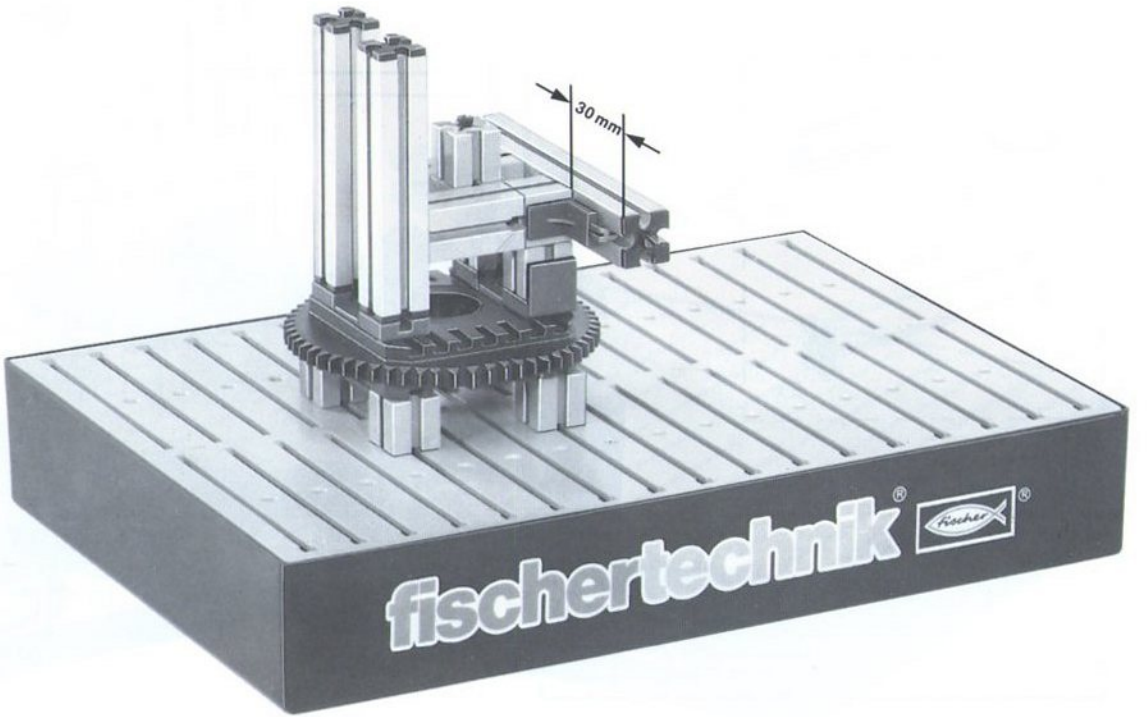




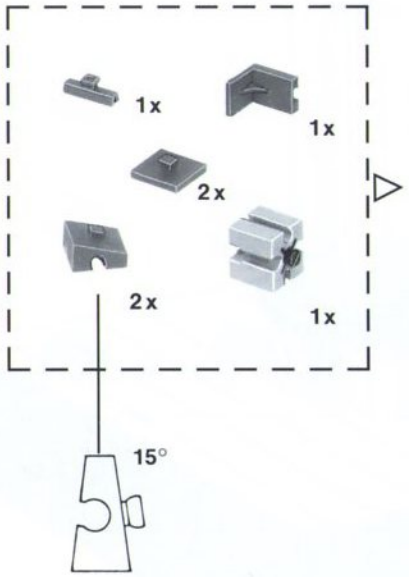
0 75 mm

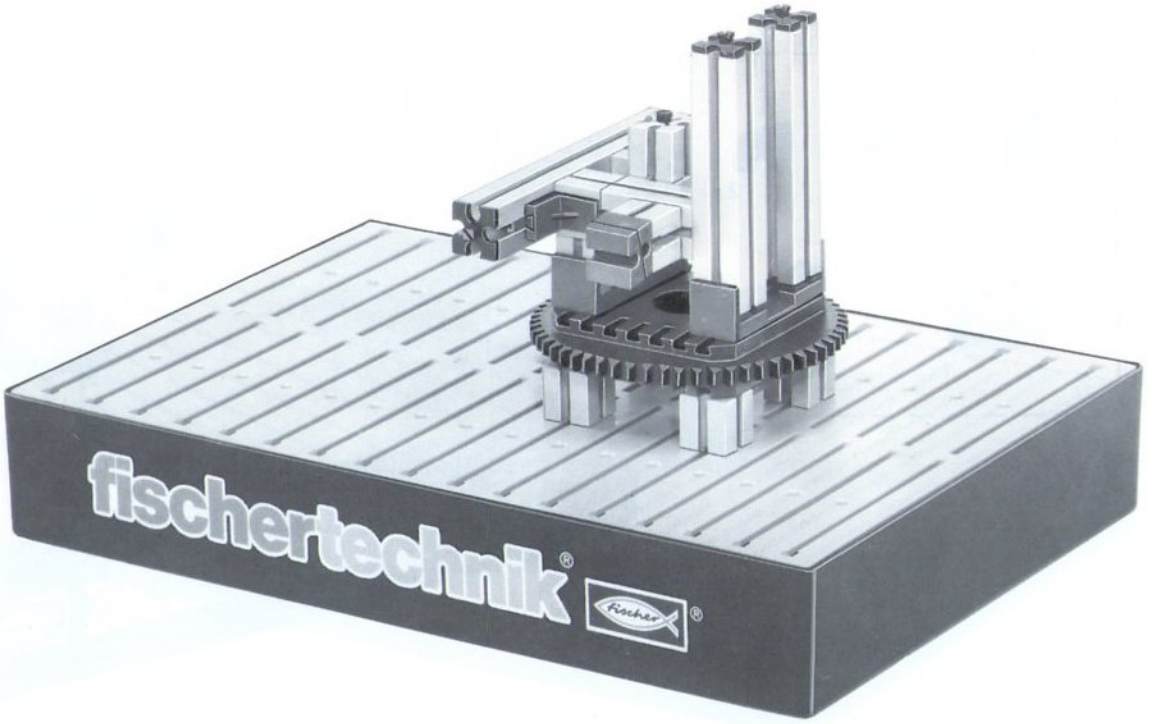
5



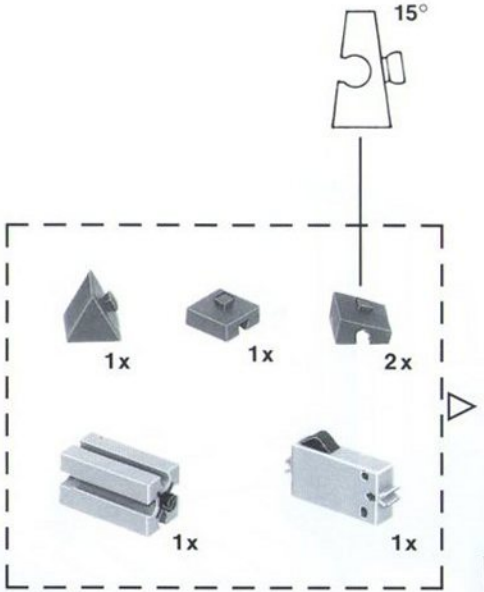


6

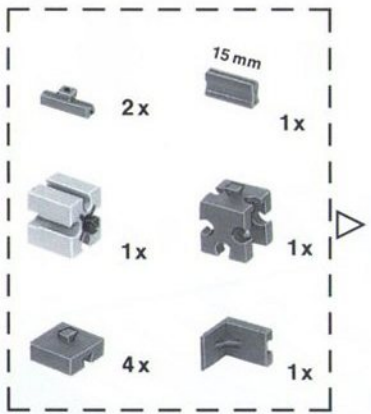


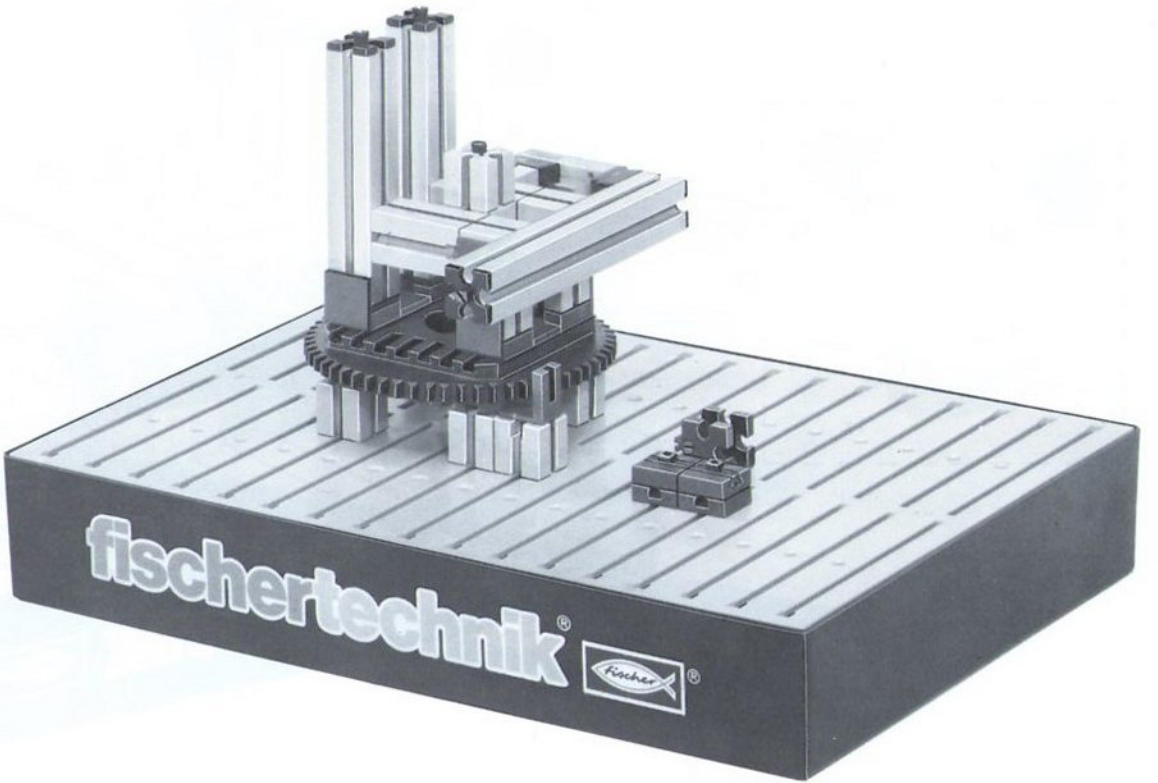


7

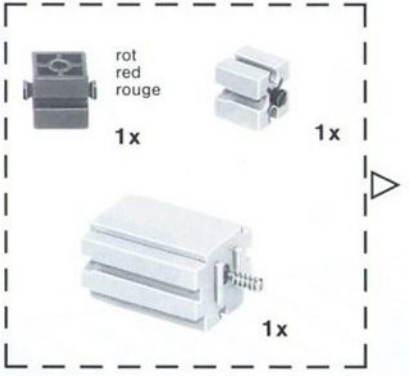


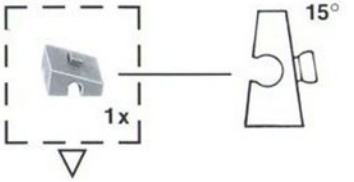
8



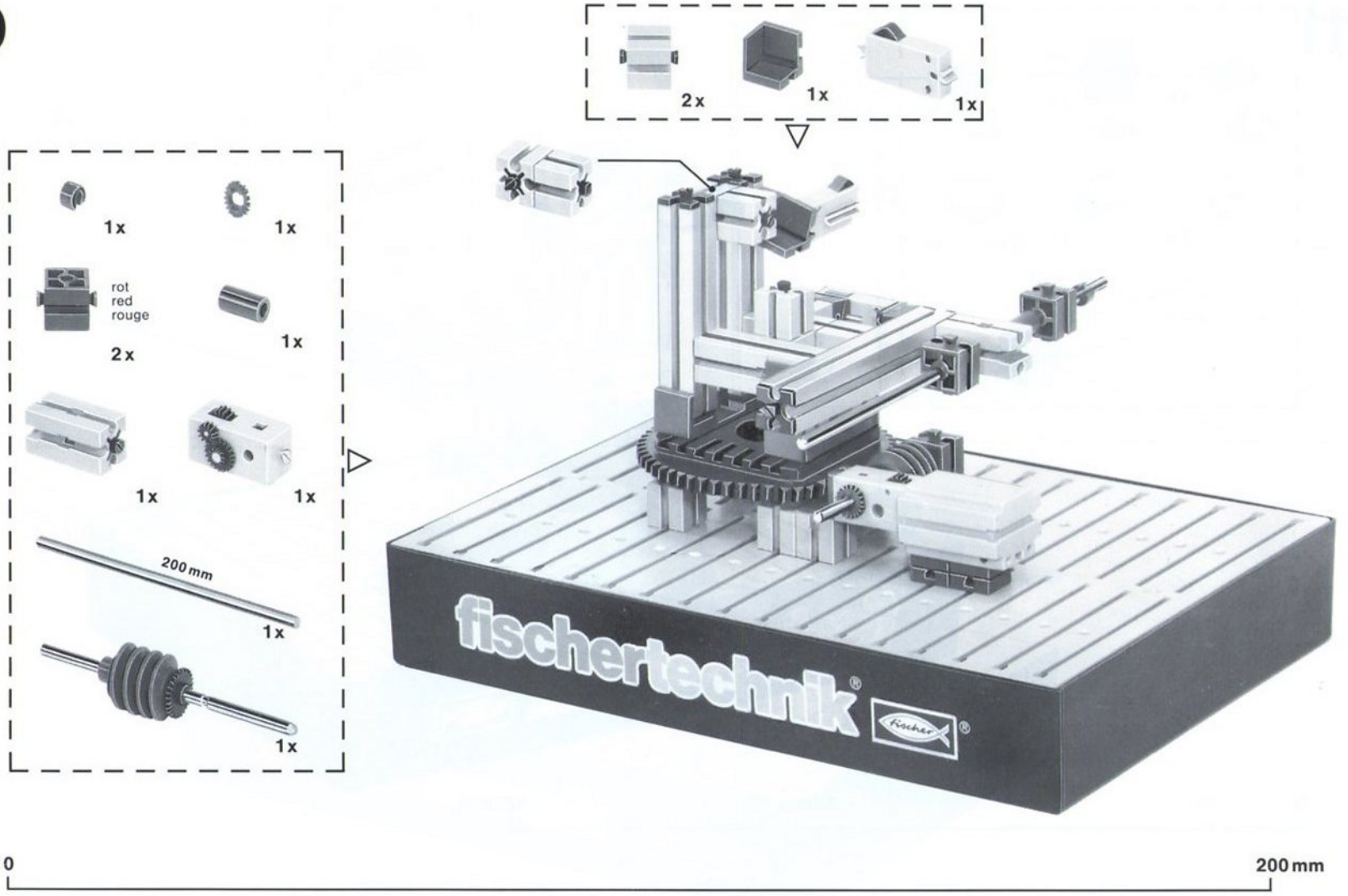


9

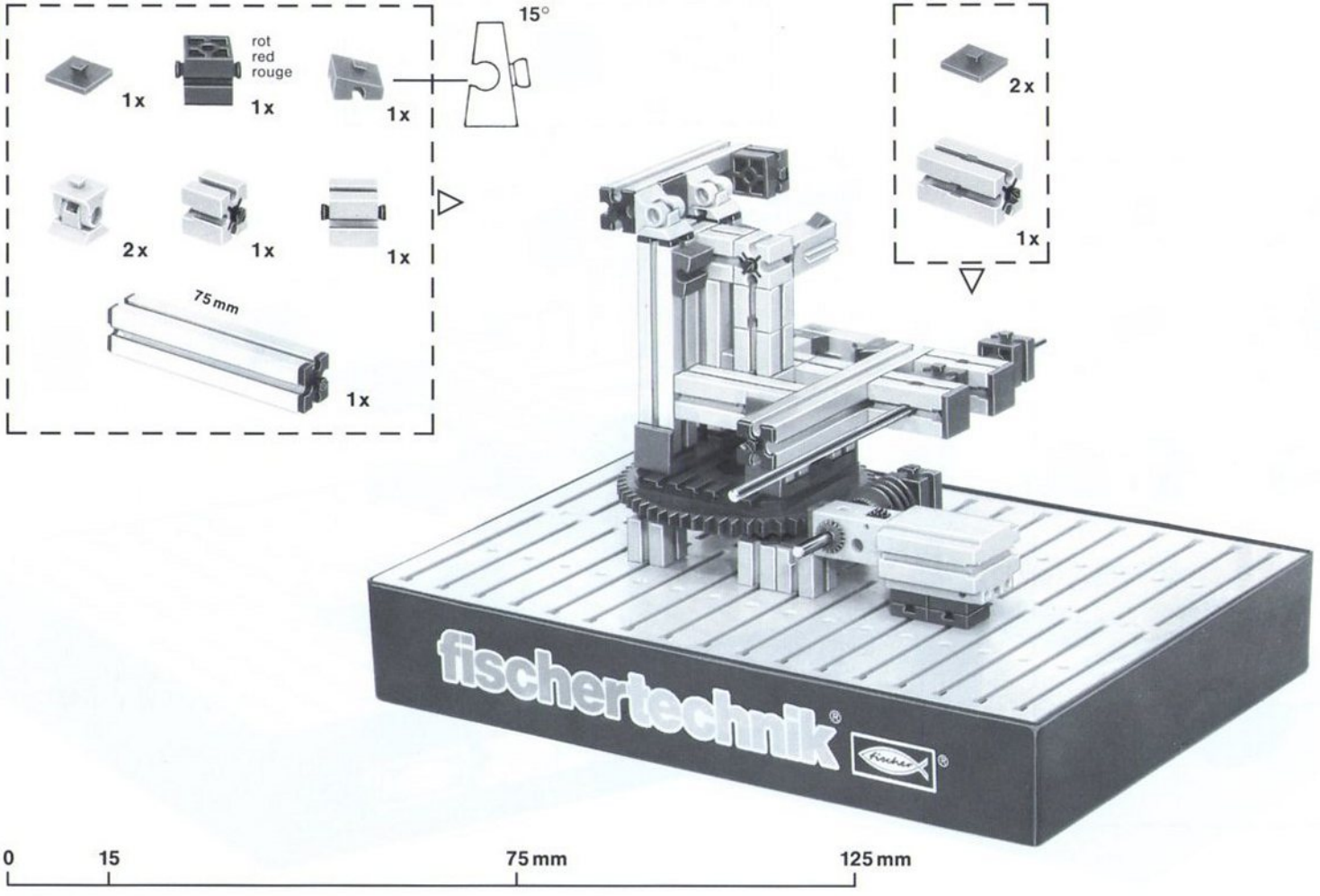




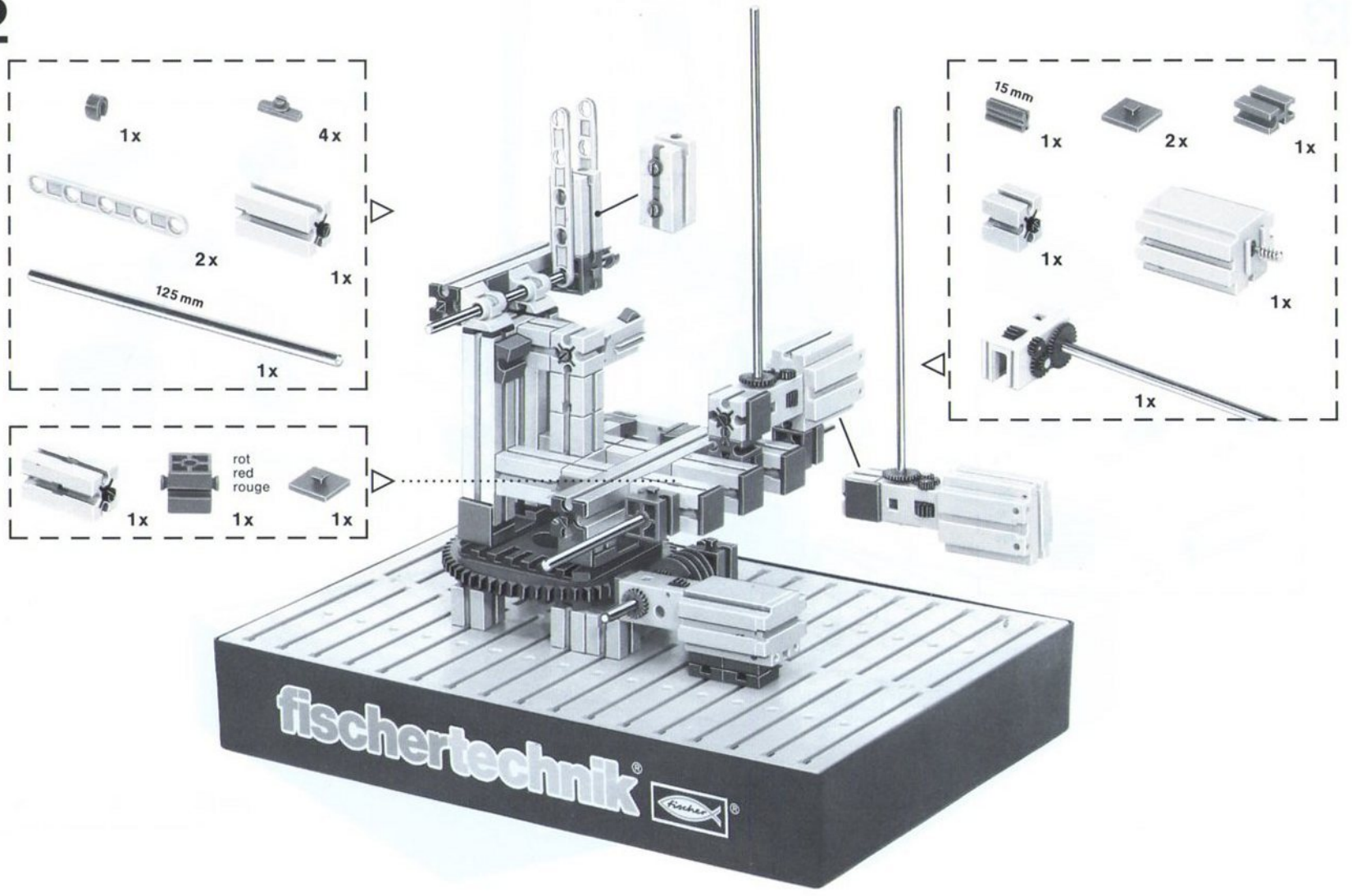
10


2x
1x
1x
1x
1x
rot
red
rouge
2x
1x
1x
1x
200 mm
1x
1x
fischertechnik
0
200 mm

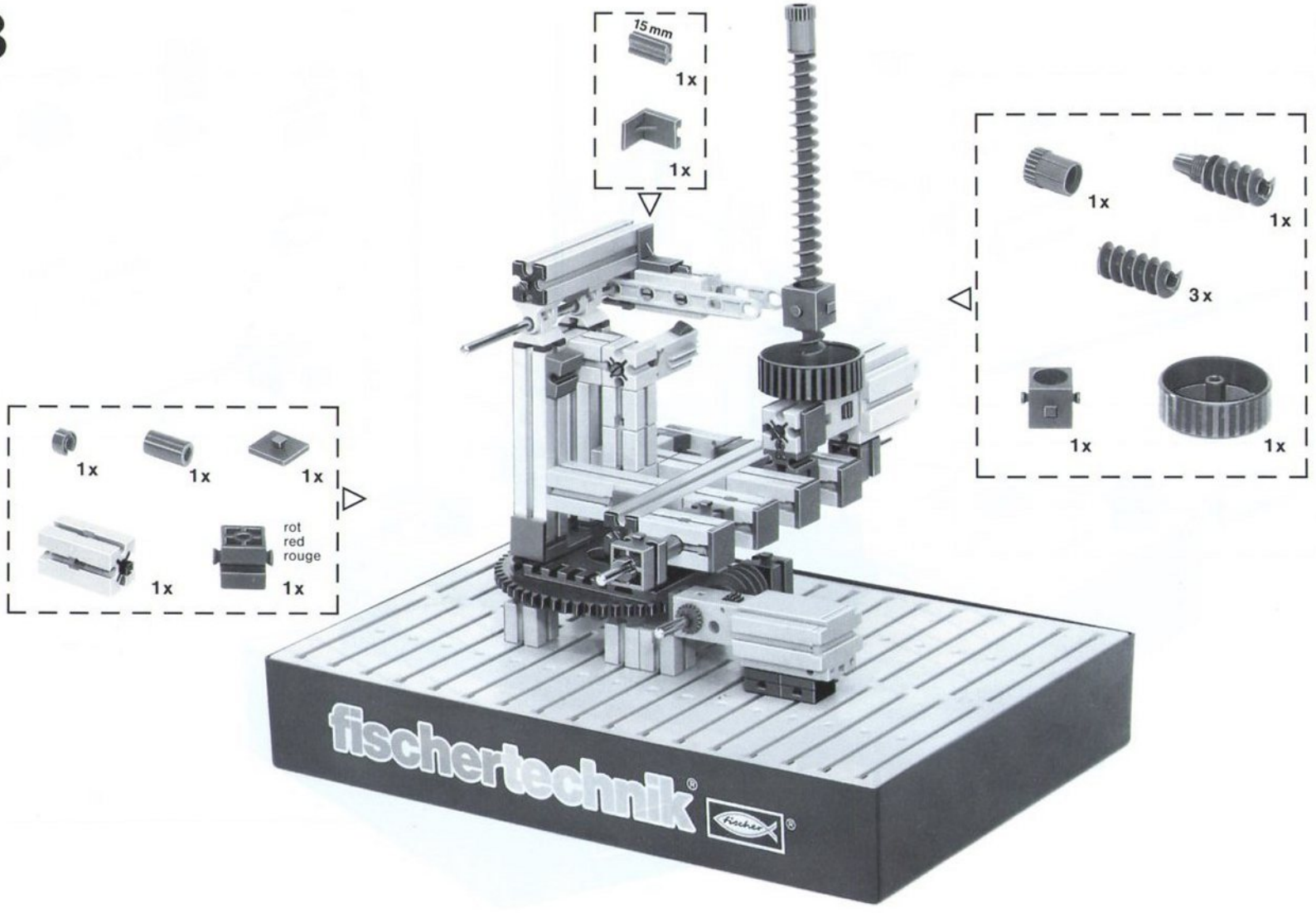
11


1x
rot
red
rouge
1x
1x
15°
2x
1x
1x
75 mm
1x
2x
1x
fischertechnik
0
15
75 mm
125 mm

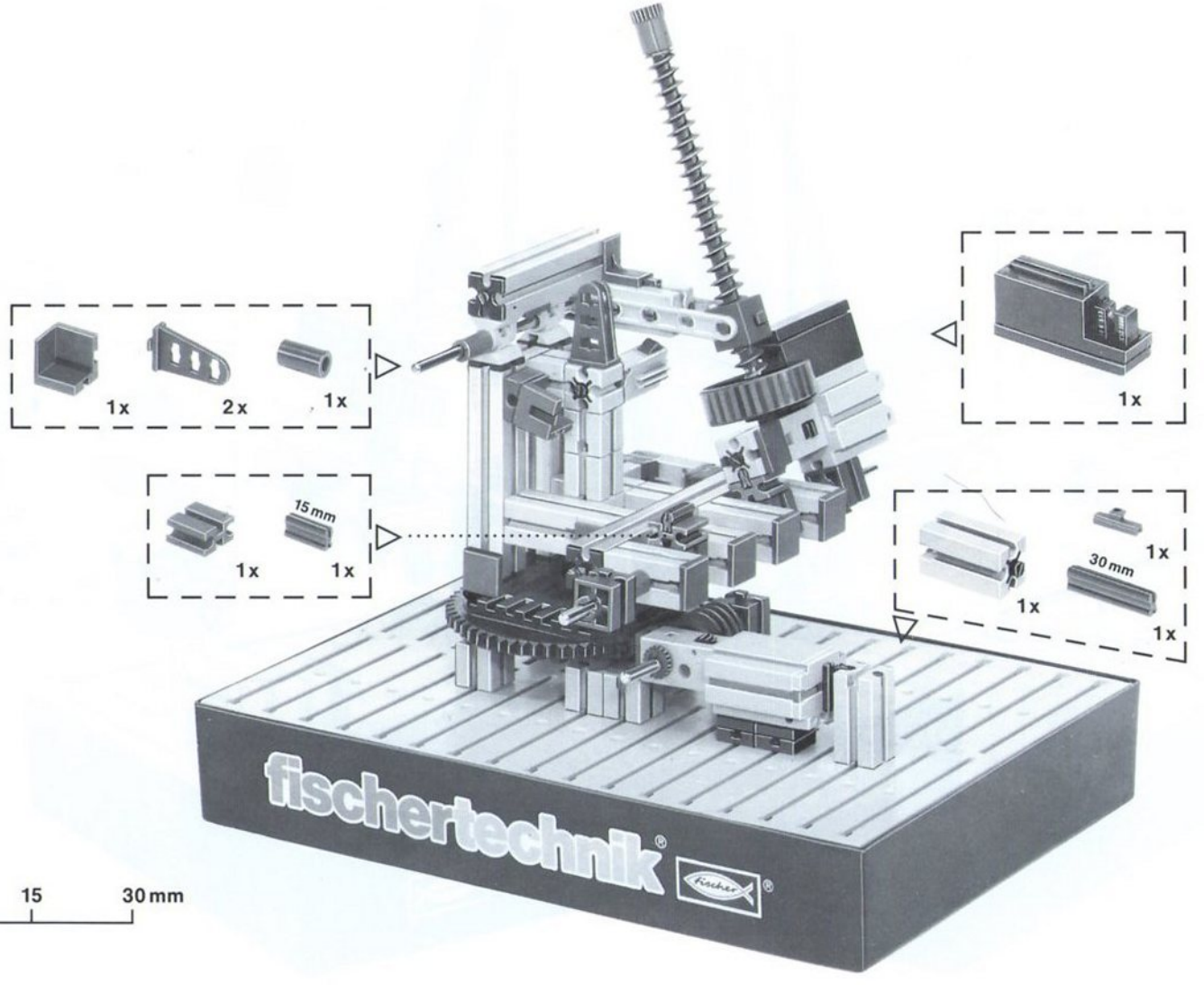
12


1x
4x
2x
1x
125 mm
1x
15 mm
1x
2x
1x
1x
1x
1x
rot
red
rouge
1x
1x
1x
fischertechnik

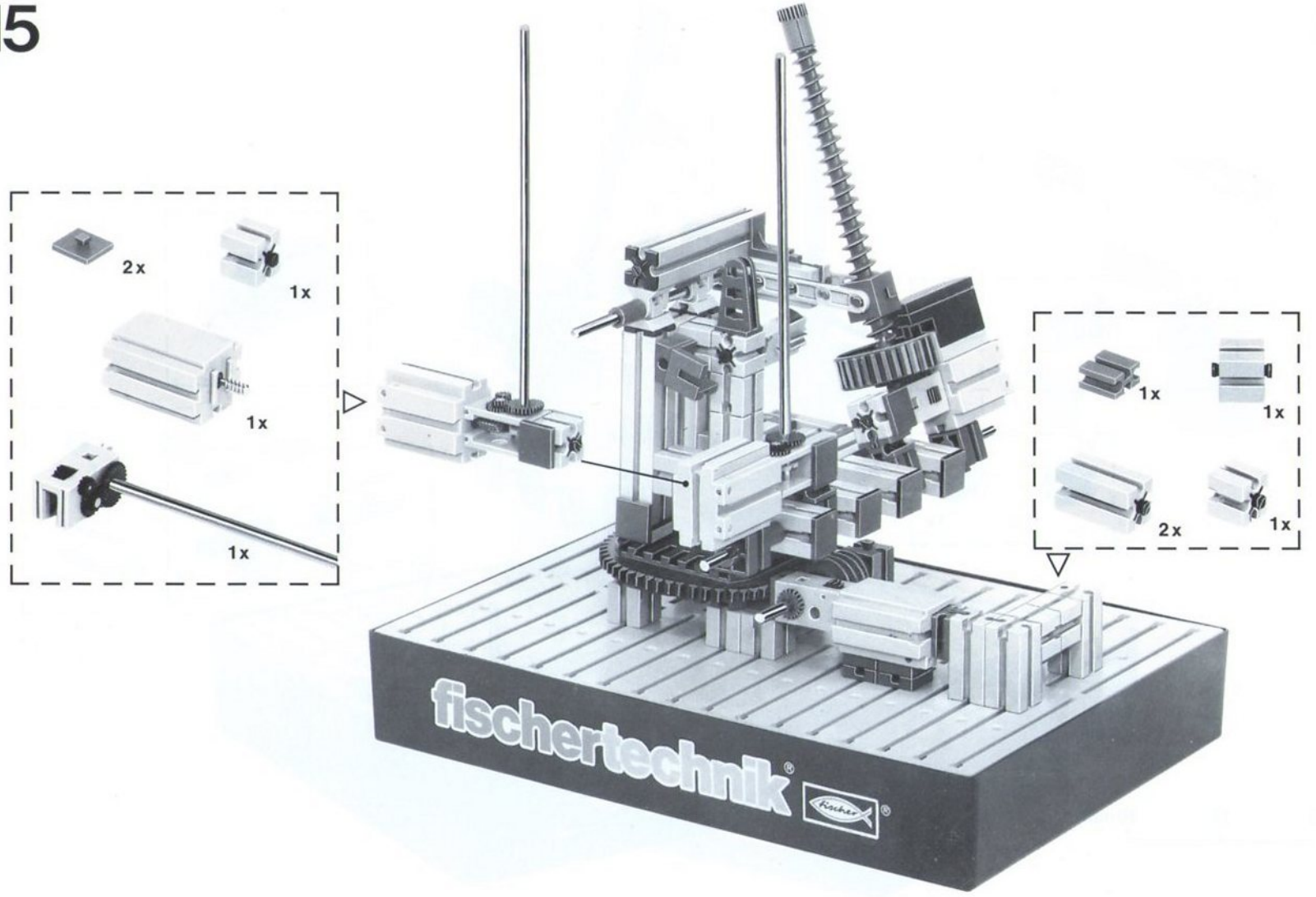
13


15 mm
1x
1x
1x
1x
1x
3x
1x
1x
1x
1x
1x
rot
red
rouge
1x
fischertechnik

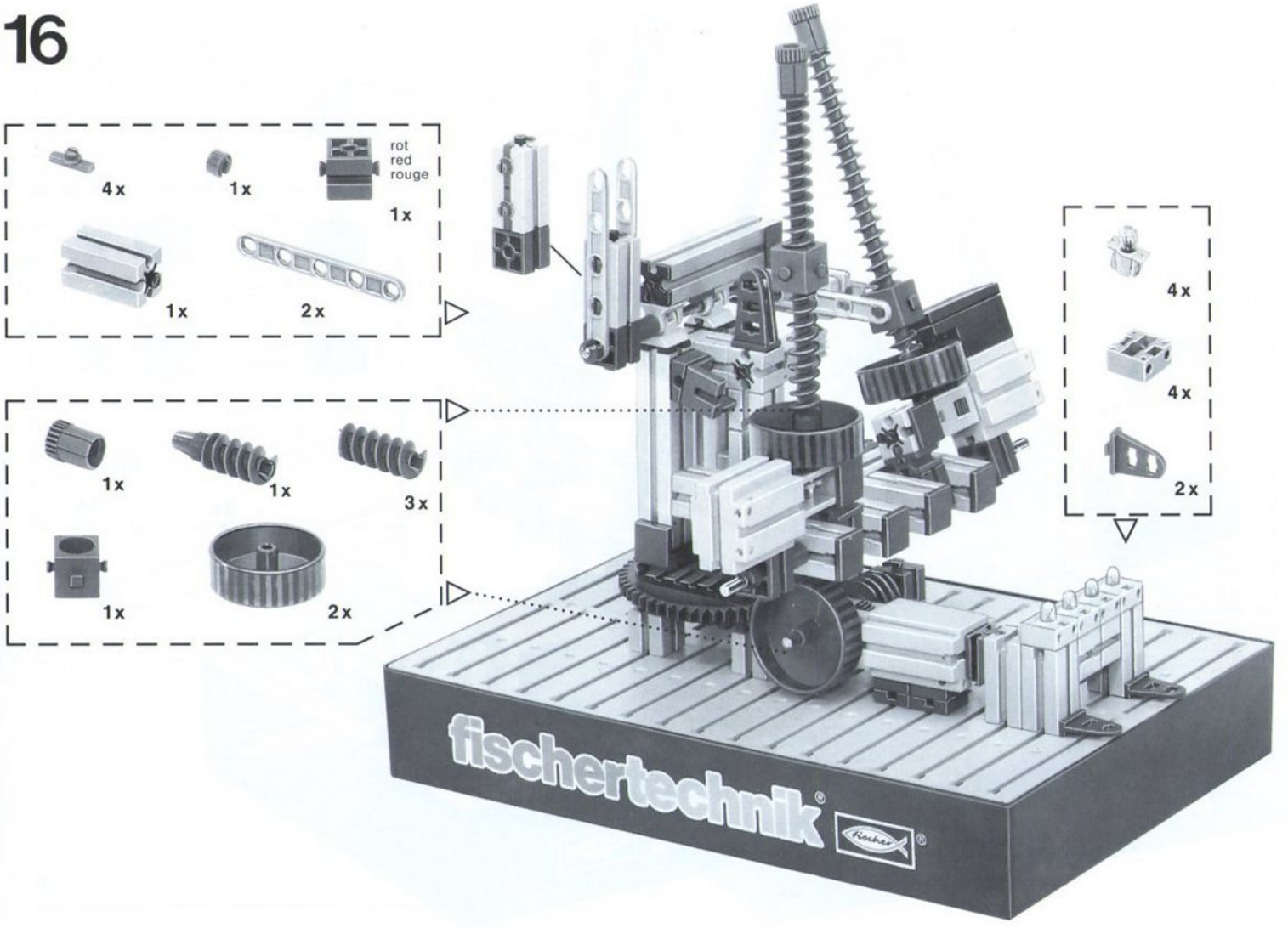
14



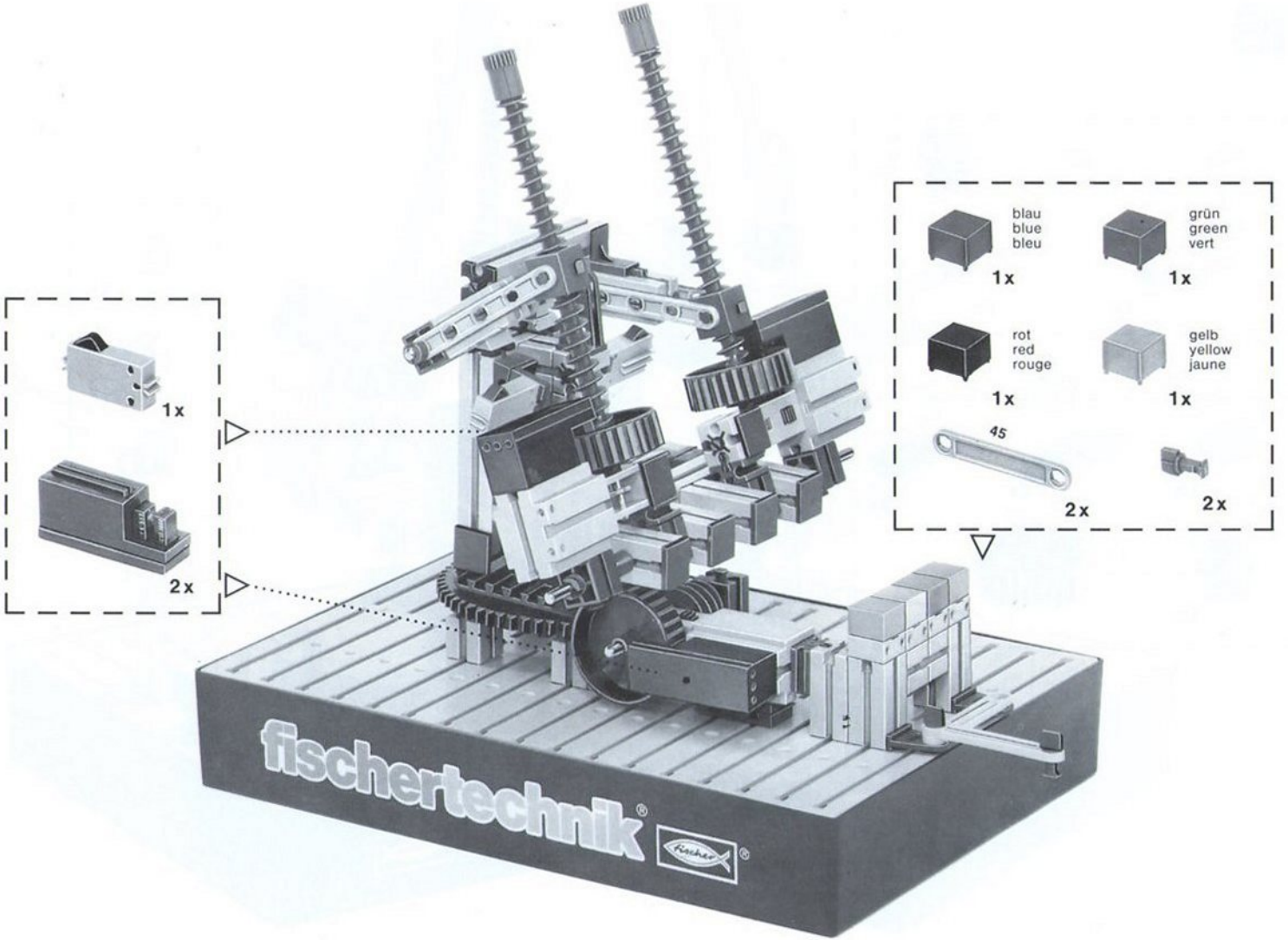
15



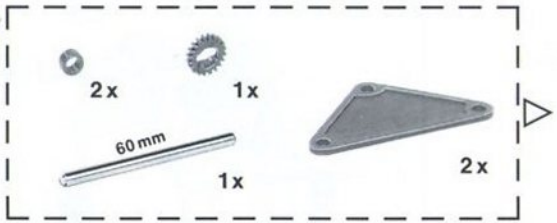
16


4x
1x
rot
red
rouge
1x
1x
2x
1x
1x
3x
1x
2x
4x
4x
2x
fischertechnik

17

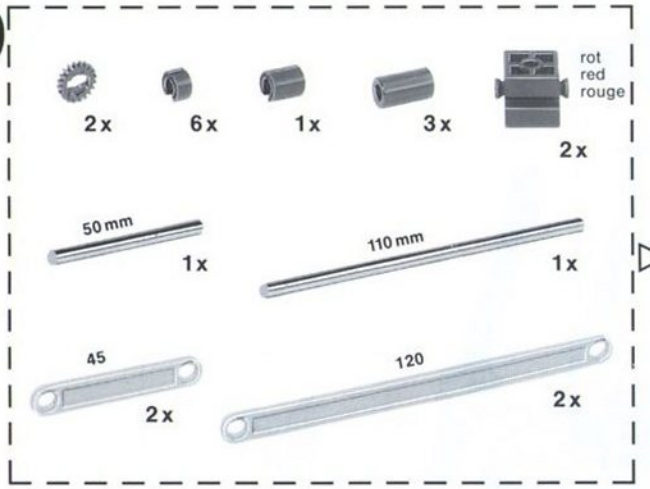

1x
2x
blau
blue
bleu
1x
grün
green
vert
1x
rot
red
rouge
1x
gelb
yellow
jaune
1x
45
2x
2x
fischertechnik

18



0 60 90 mm

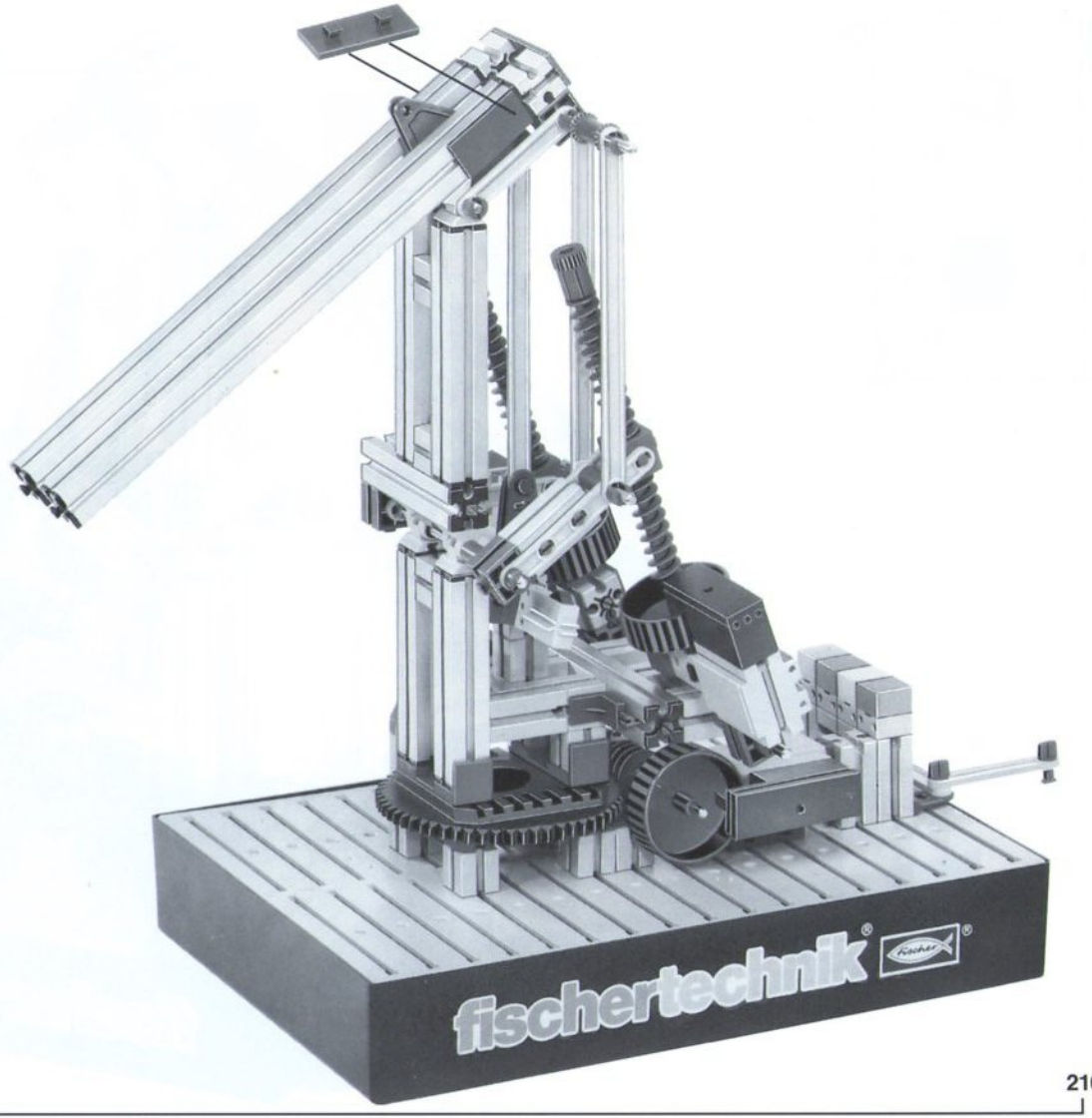
19


2x
6x
1x
3x
rot
red
rouge
2x
50 mm
1x
110 mm
1x
45
2x
120
2x

fischertechnik
fischer

0 50 110 mm

20



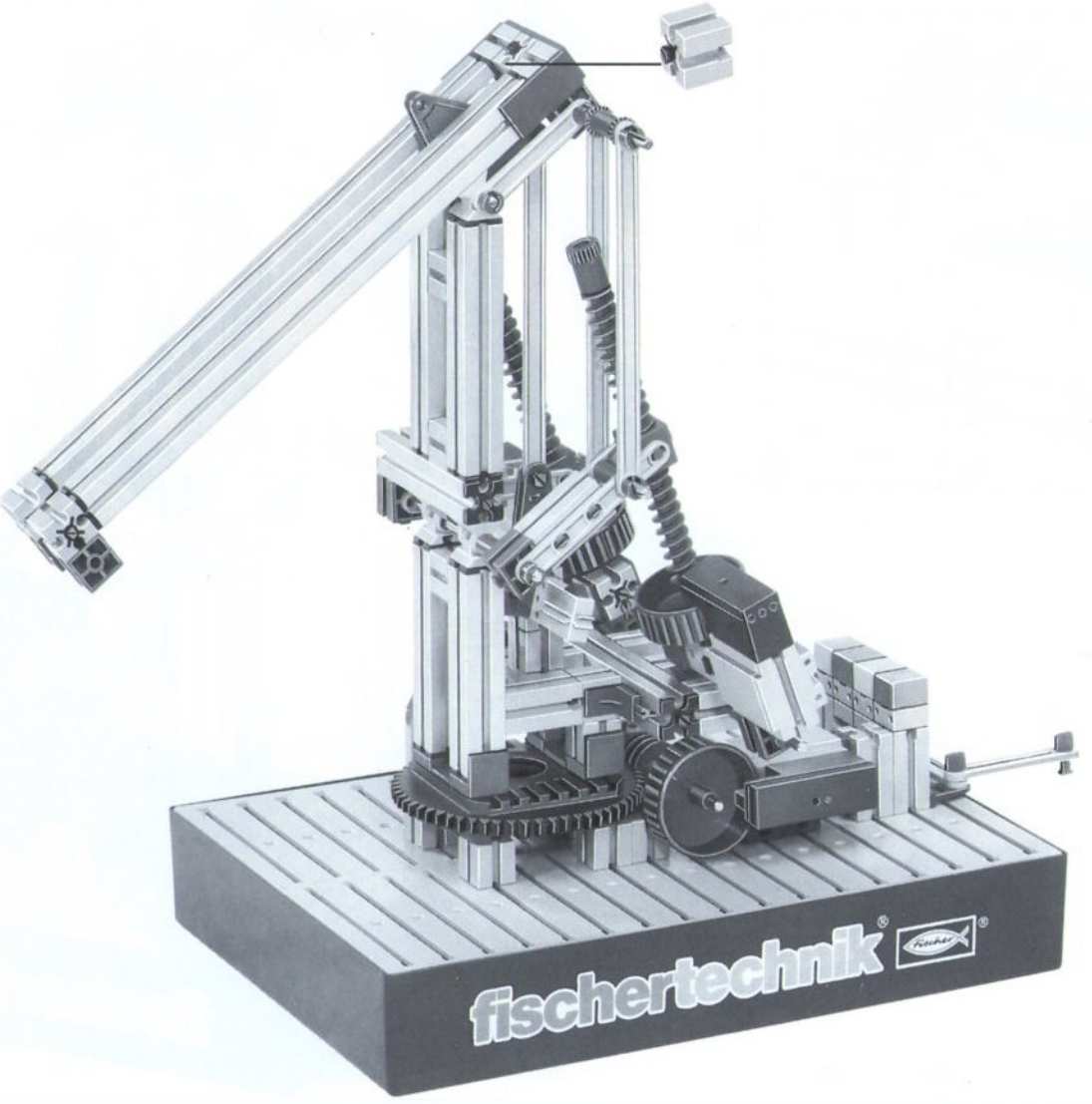
0 30 45 210 mm

21

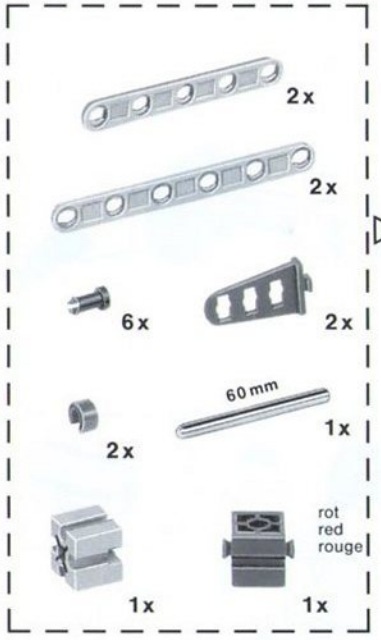
3x

rot
red
rouge

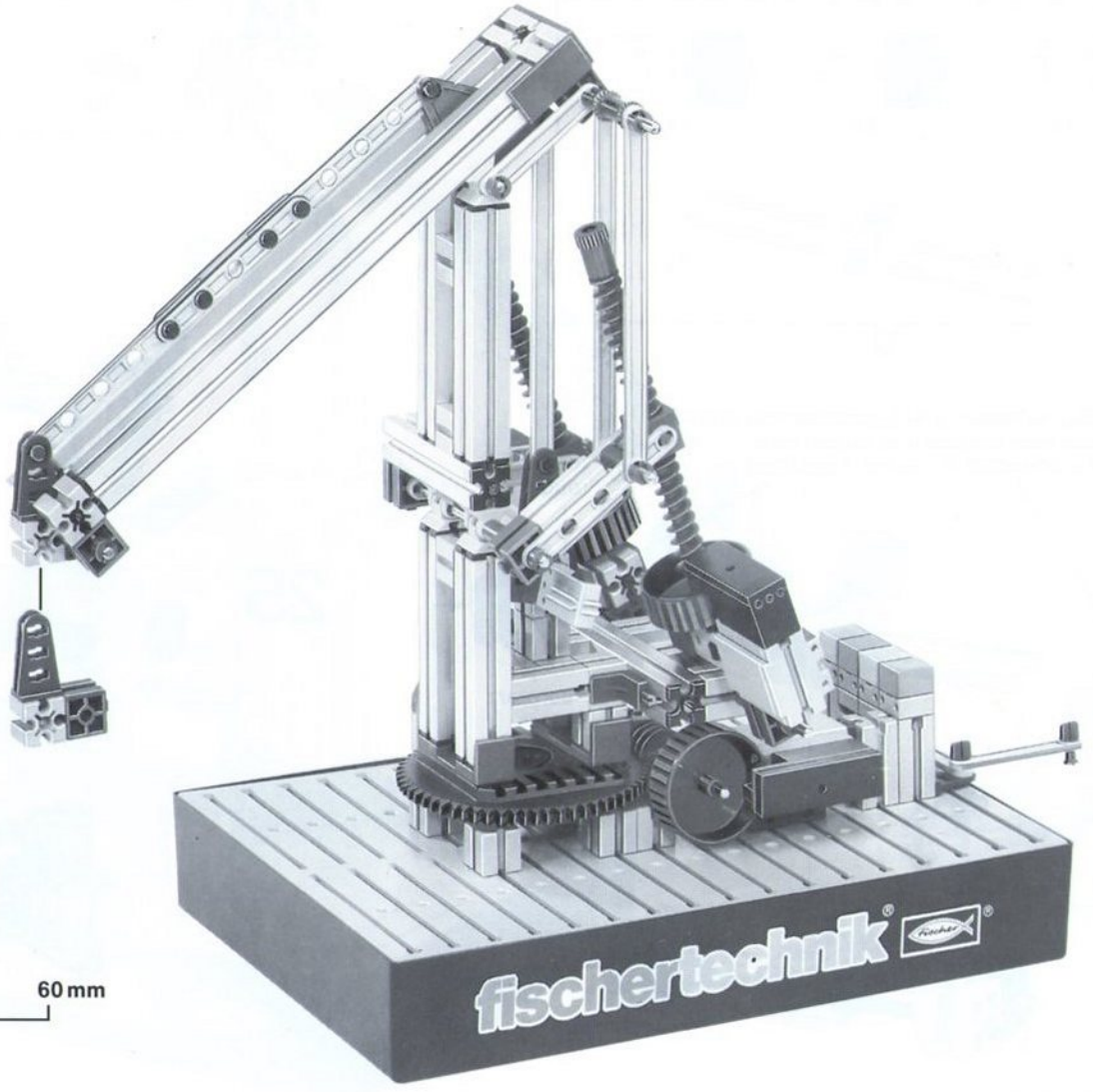
2x



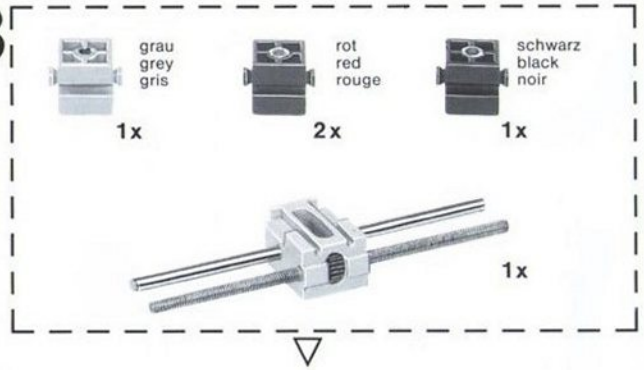
22


2x
2x
6x
2x
60 mm
1x
2x
rot
red
rouge
1x
1x

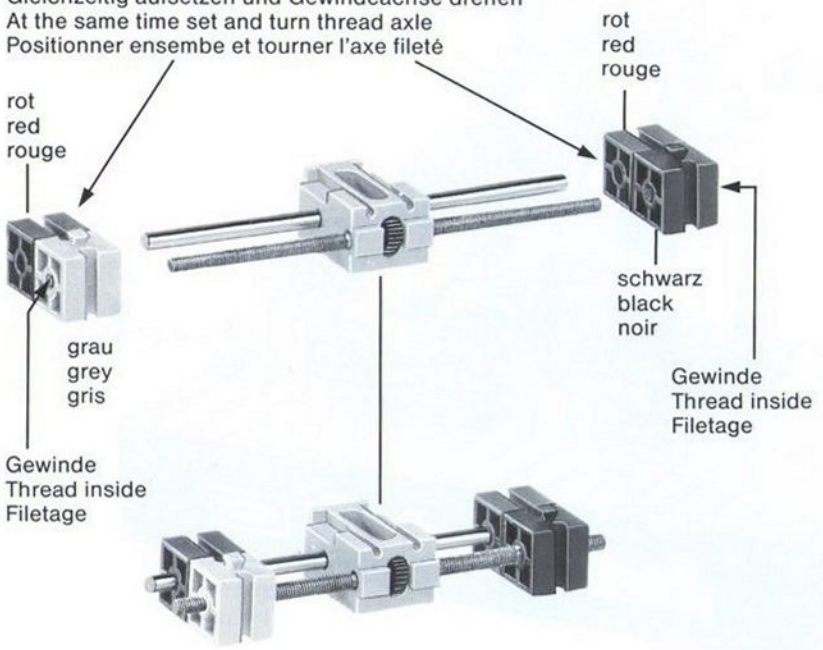

0
60 mm


fischertechnik

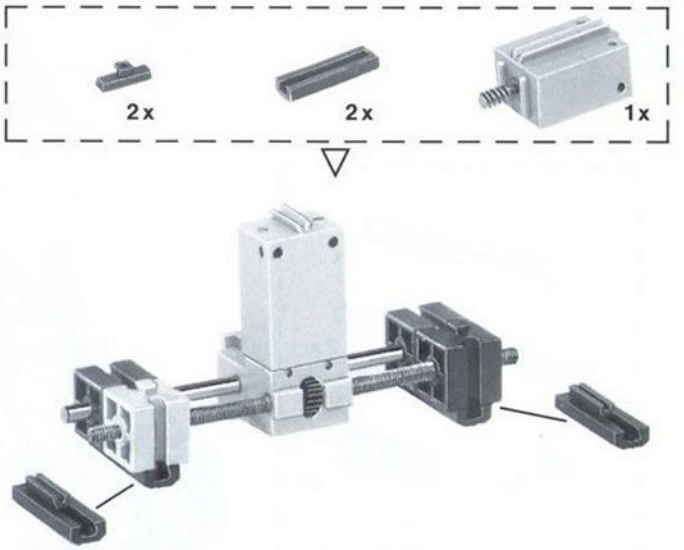
23



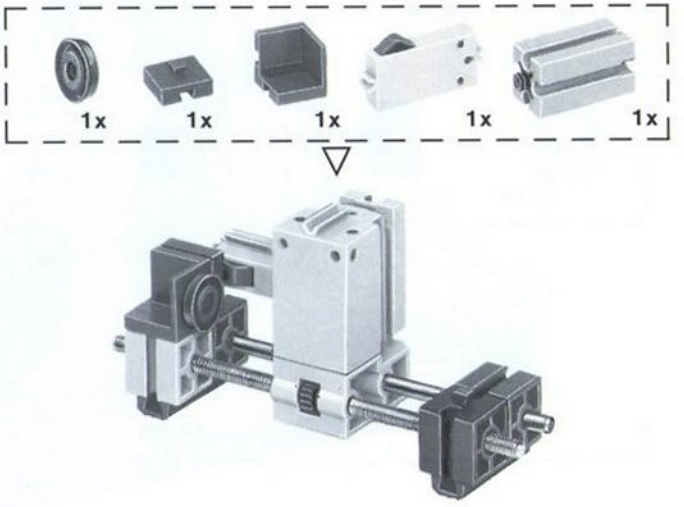
Gleichzeitig aufsetzen und Gewindeachse drehen
At the same time set and turn thread axle
Positionner ensembe et tourner l'axe fileté



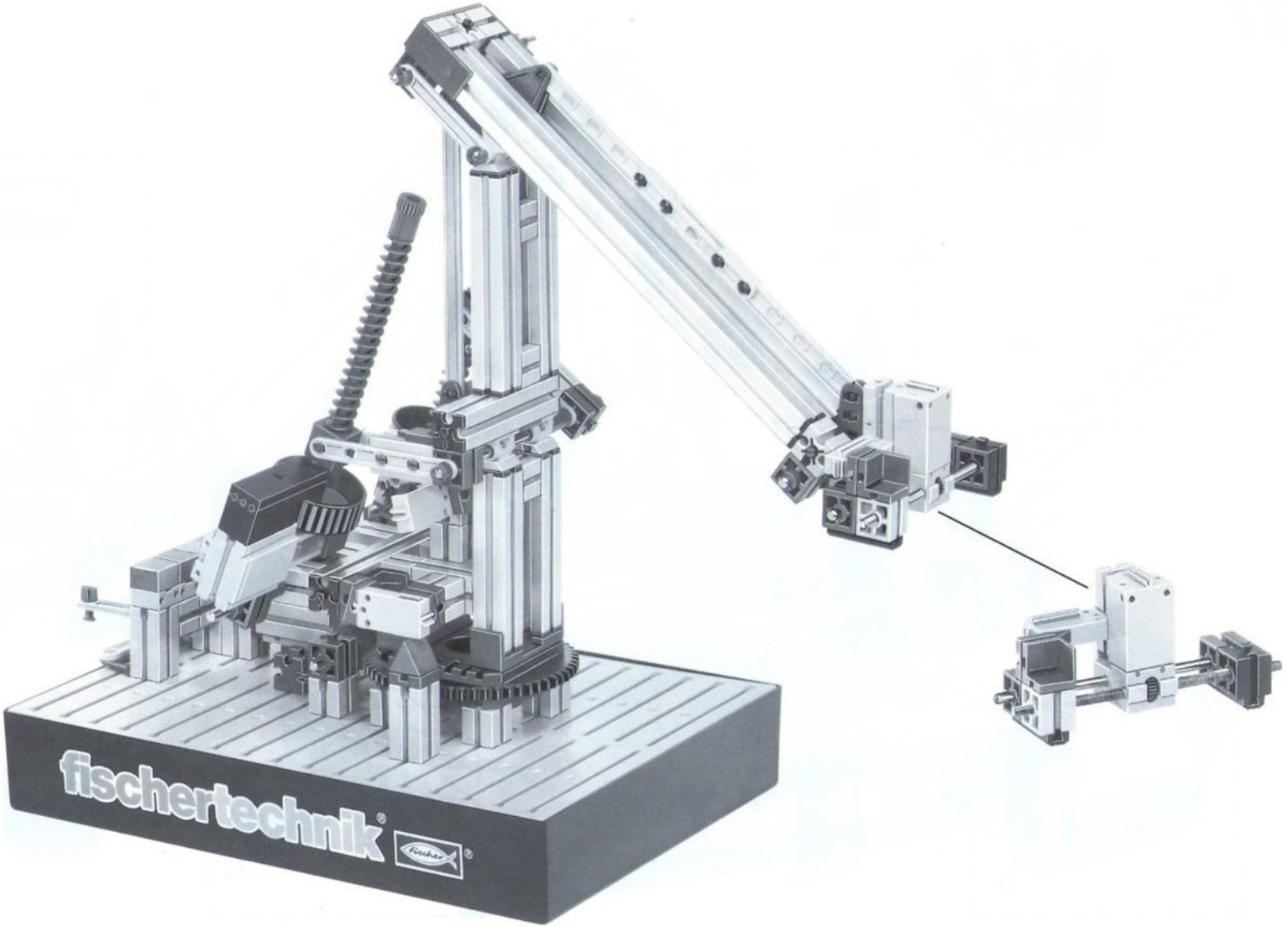
24



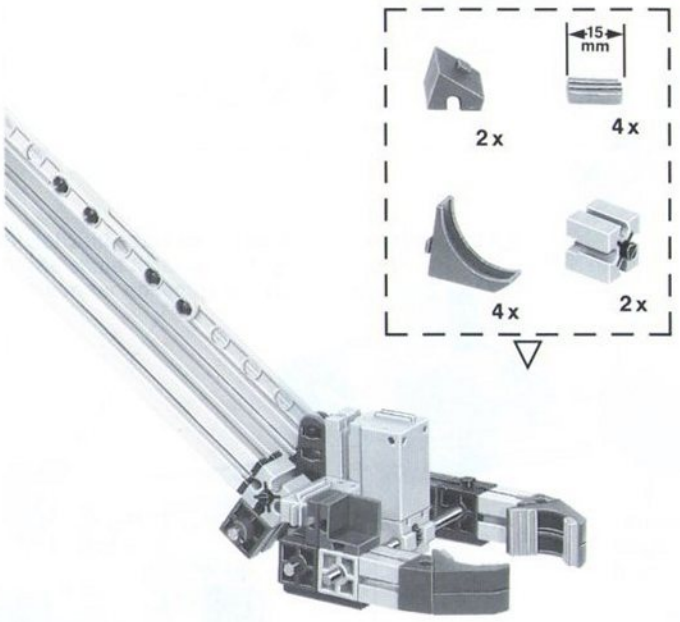
25

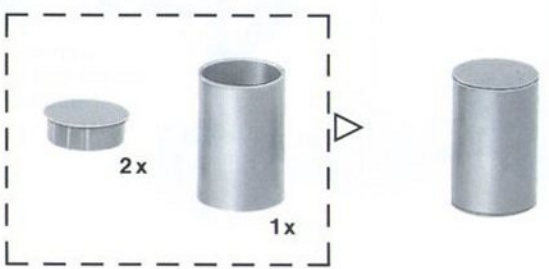


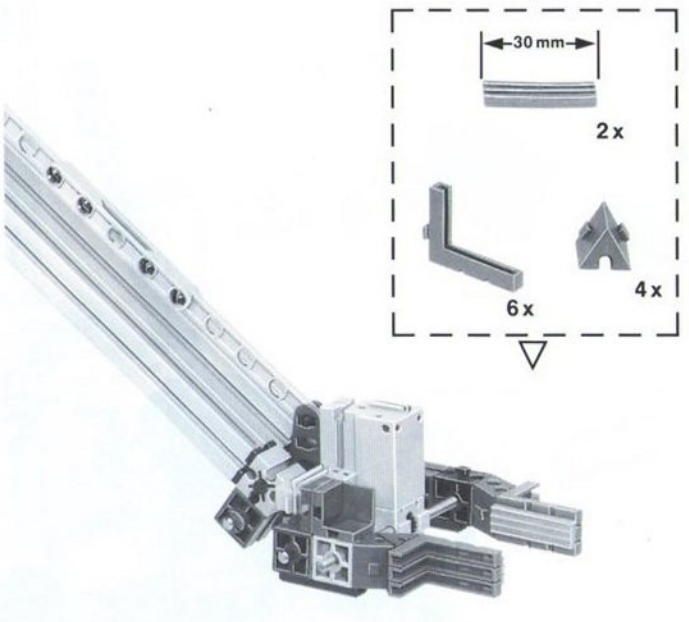
26

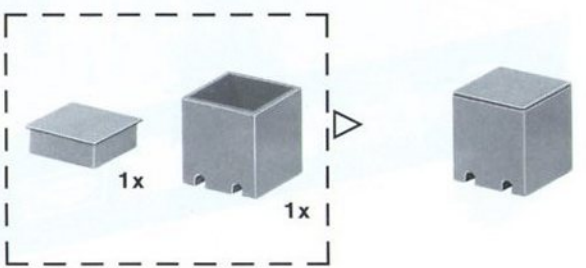


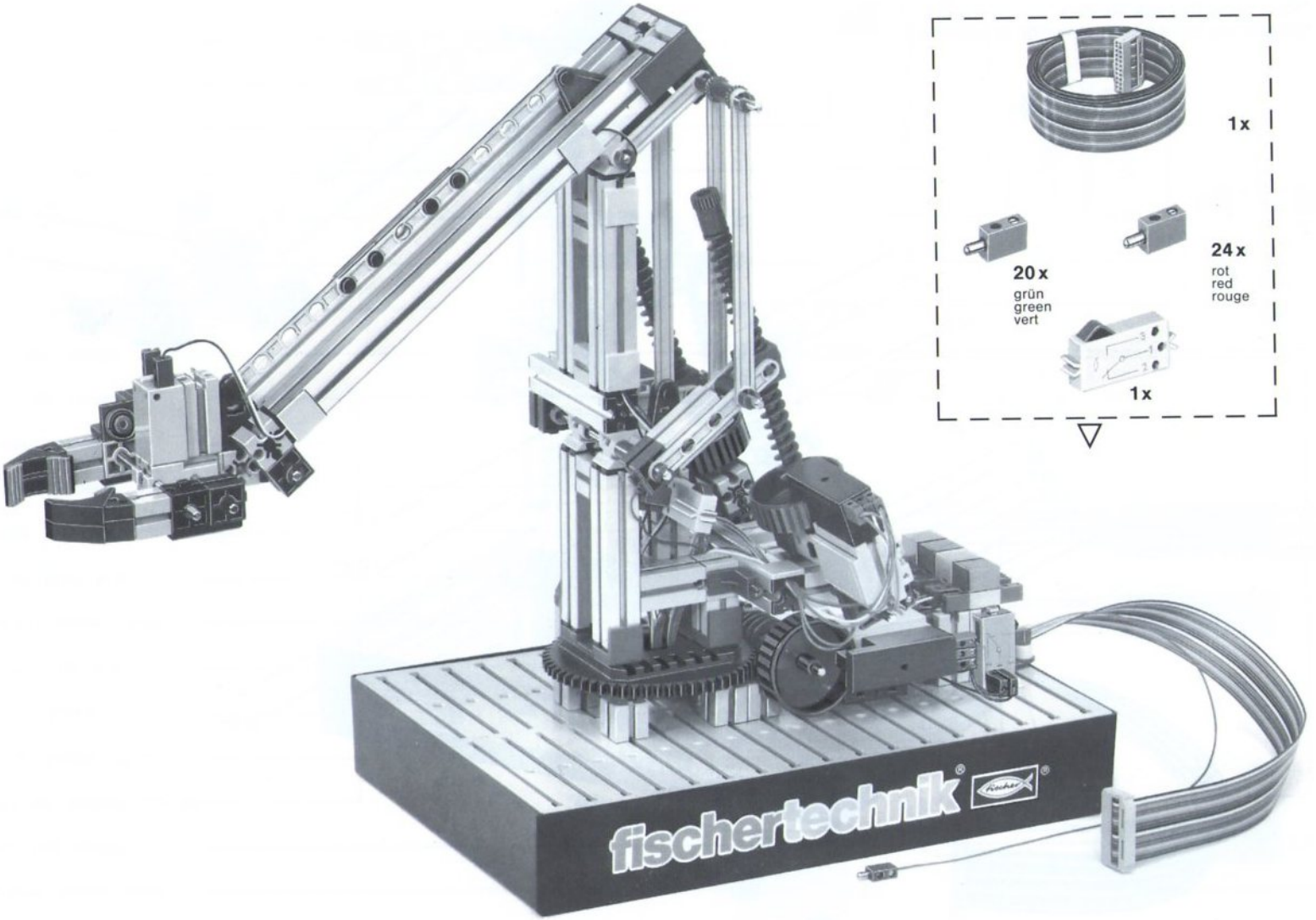
27



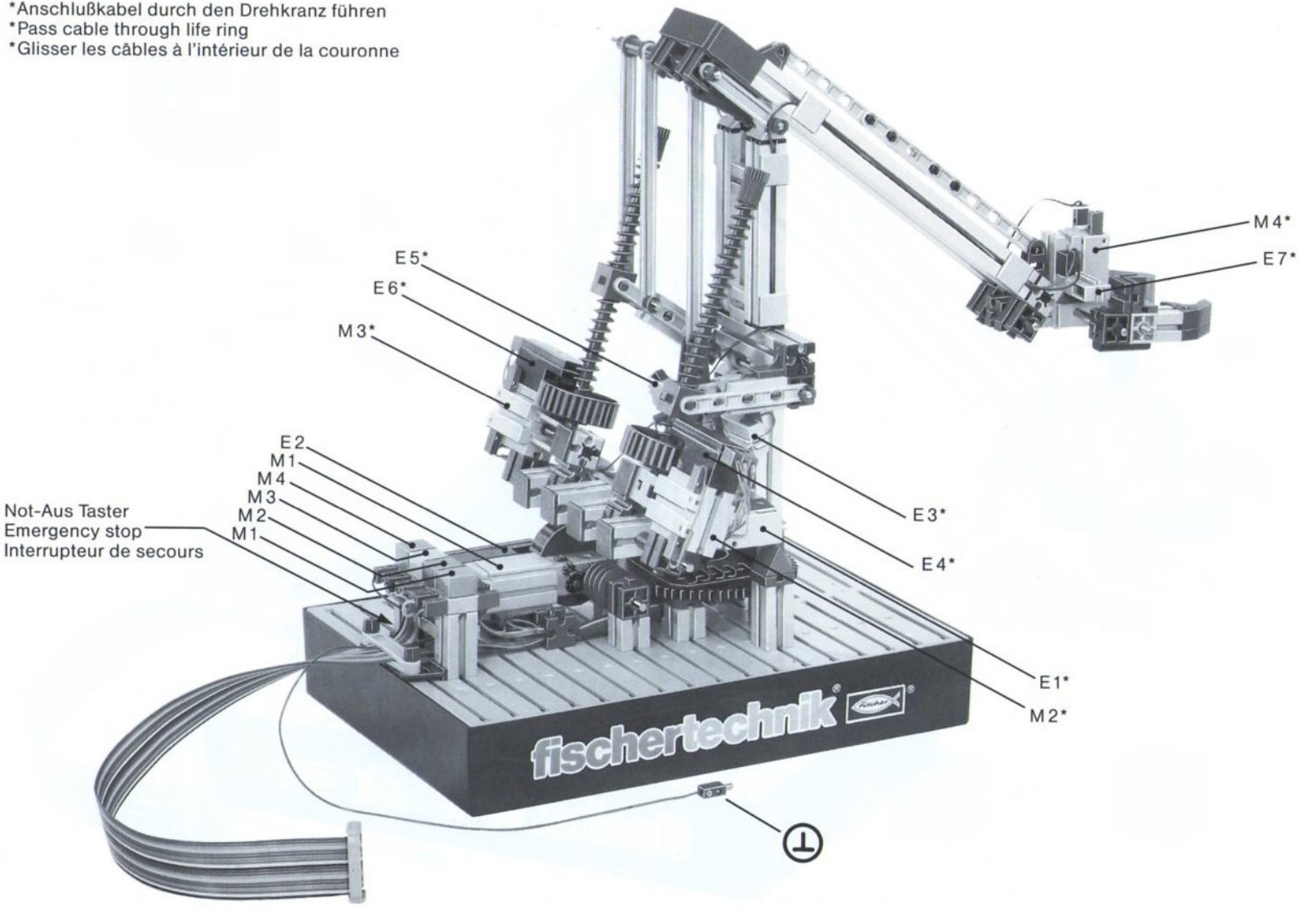




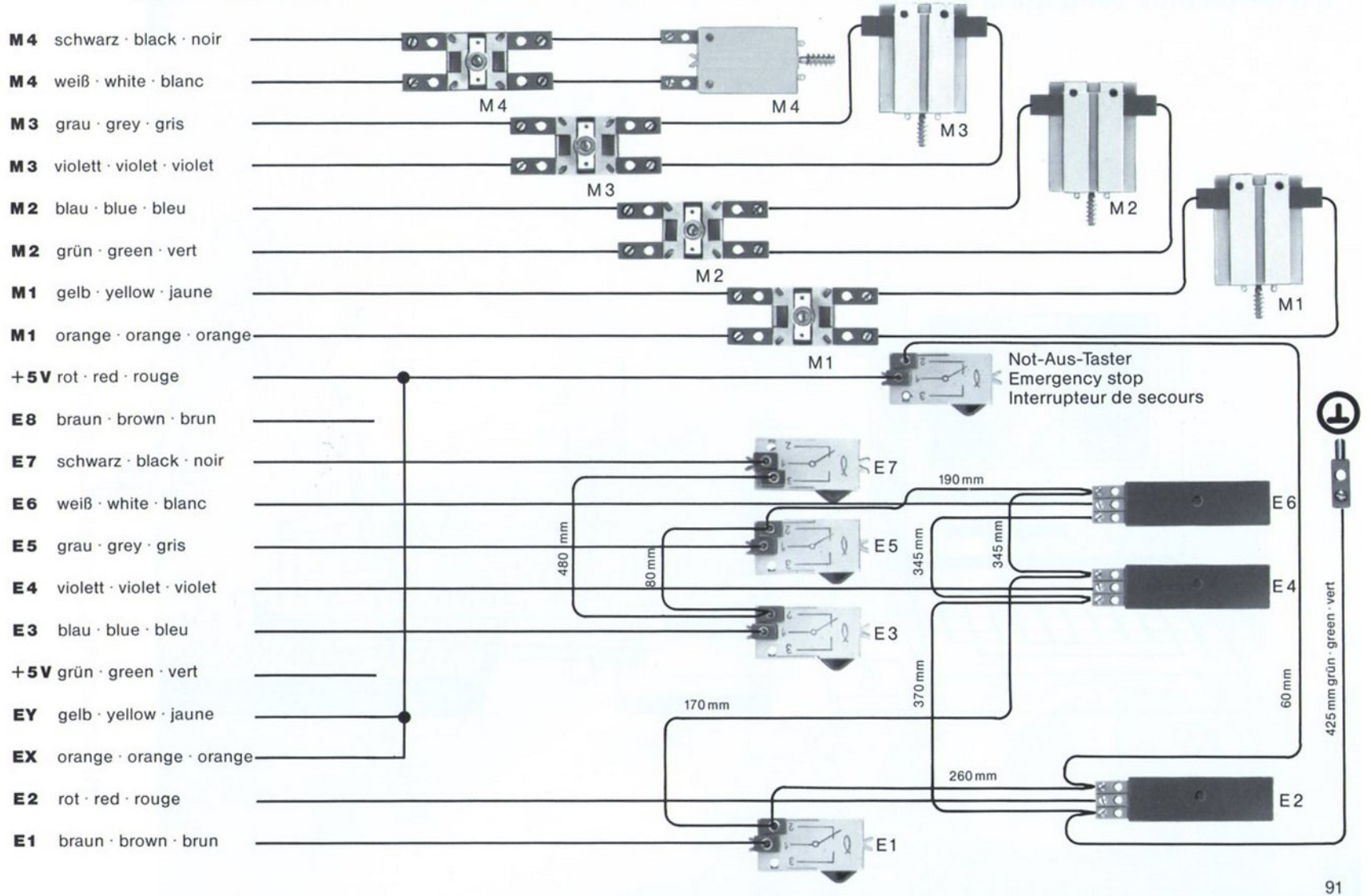


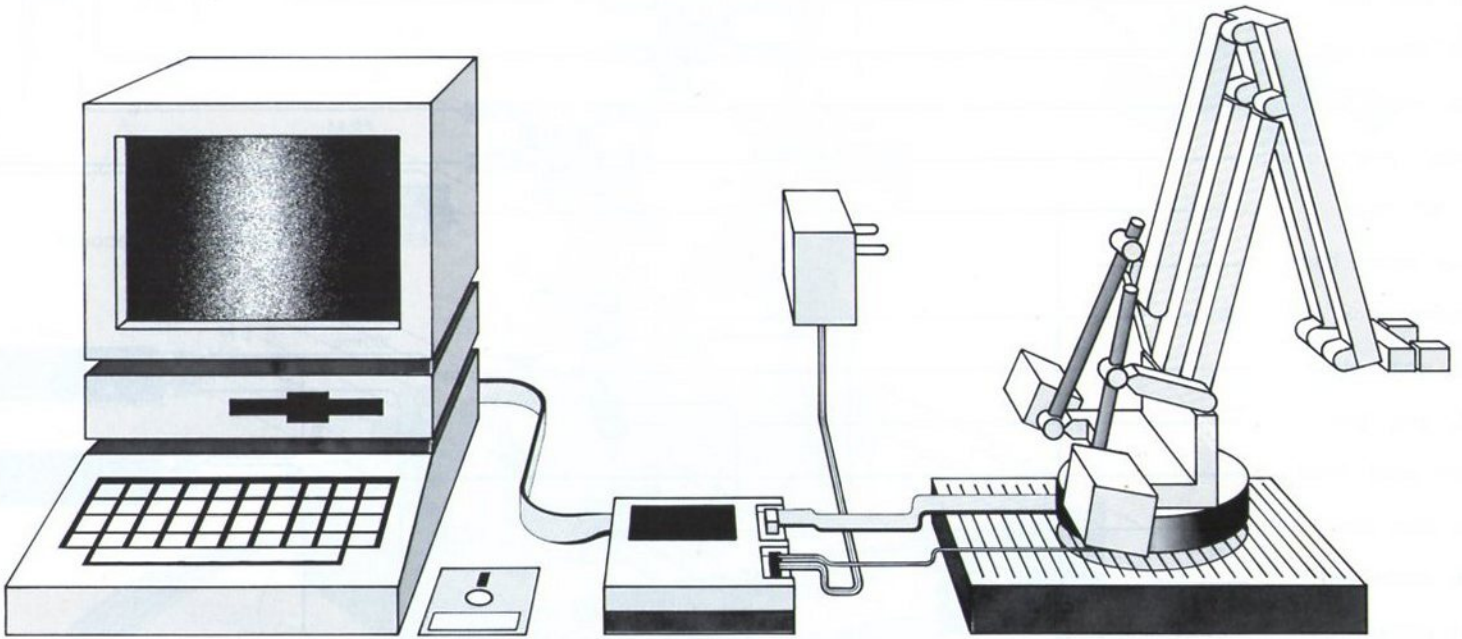

1x
20x
grün
green
vert
24x
rot
red
rouge
1x
fischertechnik

*Anschlußkabel durch den Drehkranz führen
*Pass cable through life ring
*Glisser les câbles à l'intérieur de la couronne



Verdrahtungsplan Trainingsroboter · Circuit layout Training Robot · Plan de câblage du robot d'entraînement





39461 · 0689 LA · Printed in Germany · Technische Änderungen vorbehalten · Subject to technical modifications