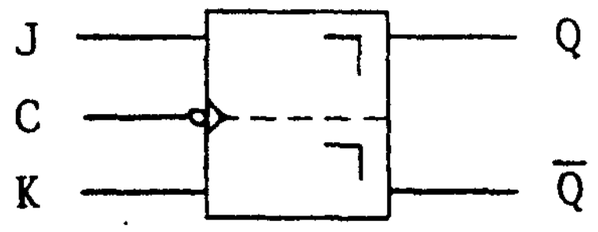
JK-Master-Slave-Flipflop, das bei negativer Taktflanke das Master-Flipflop schaltet:

Schaltzeichen:



Änderungen gegenüber dem JK-Master-Slave-Flipflop, das bei positiver Taktflanke das Master-Flipflop schaltet:

Flußdiagramm: Aus ↓ wird ↑ und umgekehrt.

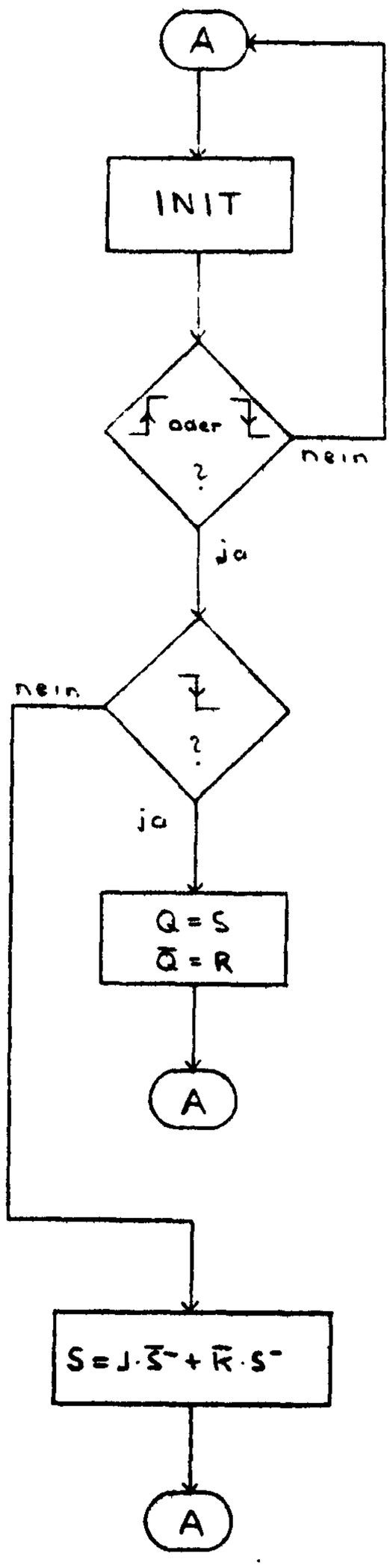
Programm:

```
09 AND  5
15 ANDC 5
```

In diesem Kapitel wurden bisher nur WENN-DANN-Strukturen in den Programmen benutzt. Folgende zwei Programme zur Darstellung von JK-Master-Slave-Flipflops haben eine WENN-DANN-SONST-Struktur und eine SOLANGE-Struktur.

JK-Master-Slave-Flipflop, das bei positiver Taktflanke das Master-Flipflop schaltet:

Flußdiagramm nebenstehend!



Festlegung der Zuordnungen im Programm:

S- = E7, C- = E5, C = E3, K = E2, J = E1, Q = A0, $\overline{Q}$ = A1

Programm:

```
00 INIT
03 LD   5
04 STO  6
05 LD   3
06 STO  5
07 XNOR 6
08 SKZ  x
09 JMP  x
10 LDC  5
11 OEN  0
12 LD   7
13 STO  0
14 STOC 1
15 OEN  5
16 LD   1
17 ANDC 7
18 STO  6
19 LDC  2
20 AND  7
21 OR   6
22 STO  7
23 JMP  x
```

Es muß geprüft werden, ob eine Taktflanke vorliegt. Sind der alte und der neue Taktzustand gleich, so muß die XNOR-Verknüpfung eine "1" ergeben, damit nach SKZ x der Programmschritt JMP x ausgeführt wird. Dann lag nämlich weder eine steigende noch eine fallende Taktflanke vor.

JK-Master-Slave-Flipflop, das bei negativer Taktflanke das Master-Flipflop schaltet:
Änderungen gegenüber dem JK-Master-Slave-Flipflop, das bei positiver Taktflanke das Master-Flipflop schaltet:

Flußdiagramm: Aus wird .

Programm:

```
10 OEN  5
11 LD   7
12 STO  0
13 STOC 1
14 LDC  5
15 OEN  0
```

4.5.4. Programme zu arithmetischen Funktionen

Die Entstehung eines Programmes zu arithmetischen Funktionen:Bei den Programmen zu arithmetischen Funktionen ist das Entwickeln von Flußdiagrammen nicht immer zweckmäßig. Stattdessen stellt man meistens mit Hilfe von Wahrheitstabellen logische Gleichungen auf, die man dann zu vereinfachen versucht. Aus diesen logischen Gleichungen kann dann das Programm geschrieben werden. Alle Programme laufen in Stellung "Schnell-Takt" ab.
Der Übersicht wegen soll für die Äquivalenzfunktion ein neues Rechenzeichen eingeführt werden:

$x_1 * x_2 + \overline{x_1} * \overline{x_2} : x_1 \equiv x_2$

Für die Antivalenzfunktion gilt dann entsprechend:

$\overline{x_1 \equiv x_2} = \overline{x_1} \equiv x_2$

Es gelten im übrigen auch für die logischen Funktionen die Rechenregeln der Algebra, wie z.B. das Distributivgesetz:

$x_1 * (x_2 + x_3) = x_1 * x_2 + x_1 * x_3$

Sehr wichtig sind die Gesetze von De Morgan:

$\overline{x_1 * x_2} = \overline{x_1} + \overline{x_2}$ und $\overline{x_1 + x_2} = \overline{x_1} * \overline{x_2}$

Diese Gesetze gelten auch für Verknüpfungen von mehr als zwei Variablen.

1. Addierer

Halbaddierer:

Ein Halbaddierer kann zwei Dualziffern (D1 und D2) addieren. Die höherwertige Stelle, d.h., das linke Bit, vom Ergebnis ist der Übertrag (Carry out-C out). Die niederwertige Stelle wird Z genannt.

Wahrheitstabelle:

D2	D1	C out	Z
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Logische Gleichungen:

$Z = \overline{D1 \equiv D2}$
C out = D1*D2

Festlegung der Zuordnungen im Programm:

D2 = E2, D1 = E1, Z = A0, C out = A1

Programm:

```
00 INIT
03 LD   1
04 XNOR 2
05 STOC 0
06 LD   1
07 AND  2
08 STO  1
09 JMP  x
```

Volladdierer:

Ein Volladdierer kann drei Dualziffern (D1, D2 und C in) addieren. Bei der Addition von zwei Dualzahlen wird der Übertrag vom letzten Ergebnis (Carry in-C in) mit addiert. Die beiden Stellen vom Ergebnis werden wieder C out (Carry out) und Z genannt.
Durch Verschachteln von mehreren Volladdierern können zwei mehrstellige Dualzahlen addiert werden. Der Carry out eines Volladdierers ist dabei der Carry in für den nächsten Volladdierer. Für das niederwertigste Bit kann auch ein Halbaddierer verwendet werden.

Wahrheitstabelle:

C in	D2	D1	C out	Z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Logische Gleichungen:

Z = D1≡D2≡C in

C out = D1*D2+D1*C in+D2*C in = (D2+C in)*D1+D2*C in

Festlegung der Zuordnungen im Programm:

C in = E3, D2 = E2, D1 - E1, Z = A0, C out - A1

Programm:

```
00 INIT
03 LD   1
04 XNOR 2
05 XNOR 3
06 STO  0
07 LD   2
08 OR   3
09 AND  1
10 STO  7
11 LD   2
12 AND  3
13 OR   7
14 STO  1
15 JMP  x
```

Mit Hilfe des IEN-Befehls kann dieses Programm auch kürzer geschrieben werden:

```
10 IEN  2
11 OR   3
12 STO  1
13 JMP  x
```

Die Programmschritte IEN 2 und OR 3 bewirken folgendes: Der Wert des Eingangs E2 gelangt mit IEN 2 in das IEN-Register des Mikroprozessors und damit an einen Eingang des UND-Gliedes der Eingangsschaltung (Siehe: Kapitel 2.2.1.). Mit OR 3 gelangt der Wert des Eingangs E3 an den anderen Eingang des UND-Gliedes und wird dort mit dem Inhalt des IEN-Registers UND-verknüpft. Das Ergebnis dieser Operation wird mit dem Inhalt des Ergebnisregisters in der Zentralen Logikeinheit ODER-verknüpft. Mit diesen beiden Programmschritten wurde also folgendes durchgeführt:
ER = E2*E3+ER
Da mit IEN 2 der Inhalt des Ergebnisregisters nicht verändert wird, braucht das Ergebnis der letzten Operation (LD 2, OR 3 und AND 1) nicht zwischengespeichert zu werden. Darin liegt auch der große Vorteil des IEN-Befehls. (Für weitere Programmteile muß die Eingabe wieder freigemacht werden.)

Addierer:

Hier wird ein Halbaddierer mit einem Volladdierer verschachtelt. Das folgende Programm ist also eine Kombination der beiden letzten Programme. Es sollen zwei zweistellige Zahlen miteinander addiert werden: Z = D1+D2.
Die beiden höchstwertigen Bits (engl. Most Significant Bit-MSB) liegen an den Eingängen E2 und E4 und die beiden niederwertigsten Bits (engl. Least Significant Bit - LSB) liegen an den Eingängen E1 und E3.

Festlegung der Zuordnungen im Programm:

D2(MSB) = E4, D2(LSB) = E3, D1(MSB) = E2, D1(LSB) = E1, Z1 = A0,
Z2 = A1, C out = A2
Programm:

```
00 INIT
03 LD   1
04 XNOR 3
05 STOC 0
06 LD   1
07 AND  3
08 STO  7
09 XNOR 2
10 XNOR 4
11 STO  1
12 LD   4
13 OR   7
14 AND  2
15 IEN  4
16 OR   7
17 STO  2
18 JMP  x
```

2. Umwandlung positiver und negativer Binärzahlen

Im Dualcode gibt es nur positive Zahlen. Um auch negative Zahlen darstellen zu können, brauchen wir einen neuen Code. Der Zweier-Komplement-Code (2KC) ist dabei ein häufig benutzter Code. Dessen positiver Bereich ist gleich dem Dualcode. Die werthöchste Stelle wird als Vorzeichenstelle angesehen. Positive Zahlen sind durch eine 0, negative Zahlen durch eine 1 in der ersten Stelle von links gekennzeichnet. (Computer arbeiten bei der Zahlendarstellung stets mit festgelegter Stellenzahl. Die mögliche werthöchste Stelle ist somit stets bekannt und kann als Vorzeichenstelle verwendet werden, ohne daß Irrtümer entstehen.)
Bei der Umrechnung von positiven in negative Zahlen und umgekehrt braucht nur das Komplement der umzurechnenden Zahl gebildet werden, d.h., die Zahl wird invertiert und mit 1 addiert.

Dezimal	Dual	2KC
7	111	0111
6	110	0110
5	101	0101
4	100	0100
3	011	0011
2	010	0010
1	001	0001
0	000	0000
-1	-	1111
-2	-	1110
-3	-	1101
-4	-	1100
-5	-	1011
-6	-	1010
-7	-	1001
-8	-	1000

Im folgenden Programm sollen positive Zahlen in negative Zahlen und umgekehrt umgewandelt werden. Die umzuwandelnde Zahl liegt an den Eingängen E1 ... E4. Die umgewandelte Zahl wird an den Ausgängen A0 ... A3 angezeigt.

Wahrheitstabelle:

E4	E3	E2	E1	A3	A2	A1	A0
0	1	1	1	1	0	0	1
0	1	1	0	1	0	1	0
0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	0
0	0	1	1	1	1	0	1
0	0	1	0	1	1	1	0
0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1
1	1	1	0	0	0	1	0
1	1	0	1	0	0	1	1
1	1	0	0	0	1	0	0
1	0	1	1	0	1	0	1
1	0	1	0	0	1	1	0
1	0	0	1	0	1	1	1
(1	0	0	0	1	0	0	0)

Logische Gleichungen:

$A0 = E1$
$A1 = \overline{E1} \equiv E2 = \overline{E1 \equiv E2}$
$A2 = \overline{E1} * \overline{E2} \equiv E3$
$A3 = \overline{E1} * \overline{E2} * \overline{E3} \equiv E4$

Programm:

```
00 INIT
03 LD   1
04 STO  0
05 XNOR 2
06 STOC 1
```

```
07 LDC  1
08 ANDC 2
09 XNOR 3
10 STO  2
11 LDC  1
12 ANDC 2
13 ANDC 3
14 XNOR 4
15 STO  3
16 JMP  x
```

Hat die umzuwandelnde Zahl mehr als vier Stellen, so wird das Verfahren mit einer Wahrheitstabelle unhandlich. Dann empfiehlt es sich, das Programm nach der Umrechnungsvorschrift (Komplementbildung) aufzubauen: Für jede Stelle der umzurechnenden Zahl wird eine Wahrheitstabelle aufgestellt, beginnend mit der niederwertigsten Stelle. Die Wahrheitstabelle für die zweitniederwertigste Stelle kann auch für alle höherwertigen Stellen verwendet werden.

3. Subtrahierer

Da beim Subtrahieren auch negative Zahlen entsteh n können, werden die Ergebnisse im Zweier-Komplement-Code dargestellt. Wie beim Addierer gibt es auch hier Halbsubtrahierer und Vollsubtrahierer. Das folgende Programm stellt eine Kombination dieser beiden Schaltungen dar. Es sollen zwei zweistellige Dualzahlen voneinander abgezogen werden:
z = D2-D1.

Festlegung der Zuordnungen im Programm:

D2(MSB) = E4, D2(LSB) = E3, D1(MSB) = E2, D1(LSB) = E1, Z1 = A0, Z2 = A1, Vorzeichen = A2

Wahrheitstabelle:

E4	E3	E2	E1	A2	A1	A0
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	1	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	0	0	1
1	0	1	0	0	0	0
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	0	1	0
1	1	1	0	0	0	1
1	1	1	1	0	0	0

Logische Gleichungen:

A0 = $\overline{E1 \equiv E3}$ = $\overline{E1} \equiv E3$ C out = E1*$\overline{E3}$ = 00*E1

A1 = C in$\equiv$E2$\equiv$E4 (C in =E1*$\overline{E3}$)

A2 = C out = (C in+E2)*$\overline{E4}$+C in*E2 (C in = E1*$\overline{E3}$)

Programm:

```
00 INIT
03 LDC  1
04 XNOR 3
05 STO  0
06 AND  1
07 STO  7
08 XNOR 2
09 XNOR 4
10 STO  1
11 LD   7
12 OR   2
13 ANDC 4
14 IEN  2
15 OR   7
16 STO  2
17 JMP  x
```

4. Inkrementieren

Inkrementieren heißt eine 1 addieren.
Die an den Eingängen E1 ... E4 liegende Dualzahl wird mit 1 addiert. Das Ergebnis erscheint an den Ausgängen A0 ... A3. Liegt an den Eingängen eine 15 an, so ist das Ergebnis 16. Da 16 aber nicht dargestellt werden kann, erscheint an den Ausgängen eine 0. Zusätzlich wird dieser Überlauf durch eine "1" am Ausgang A5 angezeigt.

Wahrheitstabelle für die niederwertigste Stelle:

E1	C out	A0
0	0	1
1	1	0

Wahrheitstabelle für die zweitniederwertigste Stelle:

C in	E2	C out	A1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Für alle höherwertigen Stellen gilt die letzte Wahrheitstabelle.

Logische Gleichungen:

A0 = $\overline{E1}$ C out = E1
A1 = $\overline{C\ in \equiv E2}$ C out = C in*E2
A2 = $\overline{C\ in \equiv E3}$ C out = C in*E3
A3 = $\overline{C\ in \equiv E4}$ A5 = C out = C in*E4

Das C in einer Gleichung ist gleich dem C out der vorherigen Zeile.

Programm:

```
00 INIT
03 LD   1
04 STOC 0
05 XNOR 2
06 STOC 1
07 LD   1
08 AND  2
09 STO  7
10 XNOR 3
11 STOC 2
12 LD   7
13 AND  3
14 STO  7
15 XNOR 4
16 STOC 3
17 LD   7
18 AND  4
19 STO  5
20 JMP  x
```

5. Dekrementieren

Dekrementieren heißt 1 subtrahieren.
Das Programm zum Dekrementieren läuft im Prinzip wie das Programm zum Inkrementieren ab. Der Überlauf, der bei dem Übergang von 0 auf 15 entsteht, wird wieder durch eine "1" am Ausgang A5 angezeigt.

Wahrheitstabellen:

E1	C out	A0
0	1	1
1	0	0

C in	E2	C out	A1
0	0	0	0
0	1	0	1
1	0	1	1
1	1	0	0

Logische Gleichungen:

A0 = $\overline{E1}$ C out = $\overline{E1}$
A1 = $\overline{C\ in \equiv E2}$ C out = C in*$\overline{E2}$
A2 = $\overline{C\ in \equiv E3}$ C out = C in*$\overline{E3}$
A3 = $\overline{C\ in \equiv E4}$ A5 = C out = C in*$\overline{E4}$

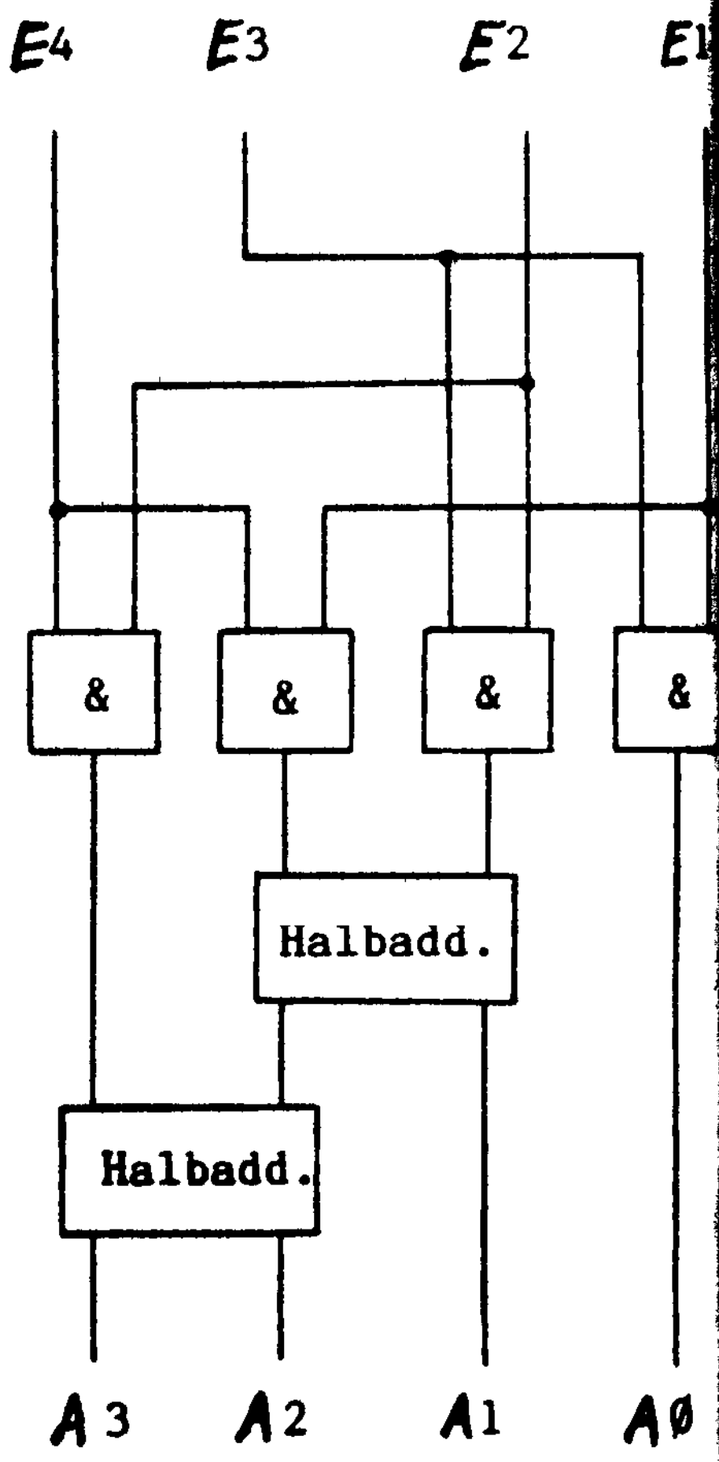
Programm:

```
00 INIT
03 LDC  1
04 STO  0
05 XNOR 2
06 STOC 1
07 LDC  1
08 ANDC 2
09 STO  7
10 XNOR 3
11 STOC 2
12 LD   7
13 ANDC 3
14 STO  7
15 XNOR 4
16 STOC 3
17 LD   7
18 ANDC 4
19 STO  5
20 JMP  x
```

6. Multiplizierer

Es sollen zwei zweistellige Dualzahlen (D1 und D2) miteinander multipliziert werden. D1 liegt an den Eingängen E1 und E2 und D2 liegt an den Eingängen E3 und E4. Das Ergebnis erscheint an den Ausgängen A0...A3.

Schaltungsprinzip nebenstehend!



Wahrheitstabelle:

E3	E1	A0
0	0	0
0	1	0
1	0	0
1	1	1

E1*E4	E2*E3	C out	A1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

C in	E2*E4	A3A	A2
0	0	0	0
0	1	0	1
1	0	-	-
1	1	1	0

Logische Gleichungen:

$A0 = E1*E3$

$A1 = \overline{E1*E4} \equiv E2*E3$

$A2 = \overline{C\ in}*E2*E4 = C\ in + \overline{E2} + \overline{E4}$

$A3 = C\ in*E2*E4 = C\ in$

$C\ out = E1*E4*E2*E3$

$(C\ in = E1*E4*E2*E3)$

$(C\ in = E1*E4*E2*E3)$

Programm:

```
00 INIT
03 LD   1
04 AND  3
05 STO  0
06 LD   1
07 AND  4
08 STO  7
09 LD   2
10 AND  3
11 XNOR 7
12 STOC 1
13 LD   7
14 AND  2
15 AND  3
16 STO  3
17 ORC  2
18 ORC  4
19 STOC 2
20 JMP  x
```

7. Paritätsprüfung

Es soll die Parität einer vierstelligen Dualzahl ermittelt werden. Ist die Anzahl der "Einsen", die an den vier Eingängen E1...E4 liegen, gerade, so soll auf dem Ausgang A0 eine "0" ausgegeben werden. Ansonsten soll dort eine "1" erscheinen.

Wahrheitstabelle:

E4	E3	E2	E1	A0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Logische Gleichung:
$A0 = \overline{E1 \equiv E2 \equiv E3 \equiv E4}$

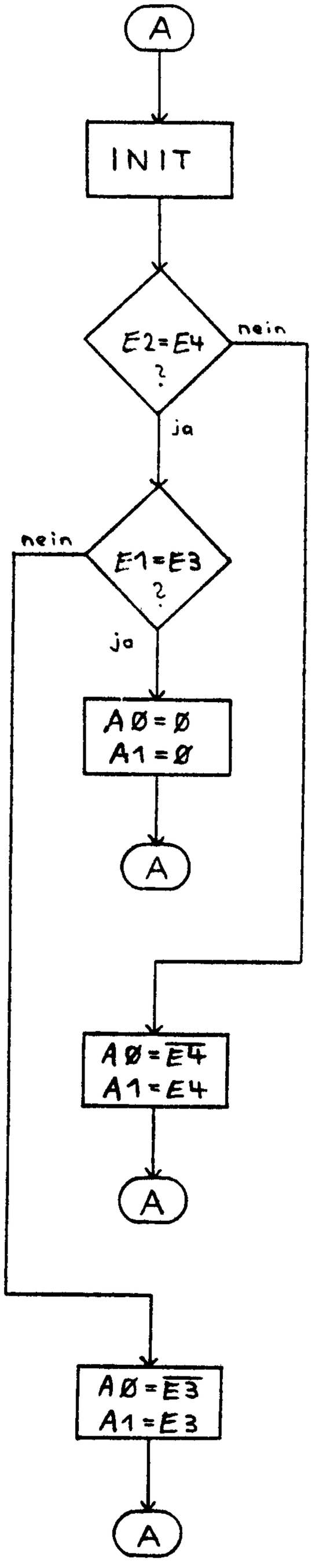
Programm:

```
00 INIT
03 LD   1
04 XNOR 2
05 XNOR 3
06 XNOR 4
07 STOC 0
08 JMP  x
```

8. Komparator

Es sollen zwei zweistellige Dualzahlen (D1 und D2) miteinander verglichen werden. D1 liegt an den Eingängen E1 und E2 und D2 liegt an den Eingängen E3 und E4. Ist D1 größer als D2, so soll das der Ausgang A0 durch eine "1" anzeigen. Ist D1 kleiner als D2, so soll das der Ausgang A1 durch eine "1" anzeigen. Sind D1 und D2 gleich groß, so sollen beide Ausgänge eine "0" anzeigen.
Das Programm beginnt mit der Untersuchung der höherwertigen Stellen E2 und E4. Ist einer der beiden größer als die andere, so brauchen die niederwertigen Stellen E1 und E3 nicht zu untersucht werden. Bei Gleichheit werden E1 und E3 untersucht. Daher eignet sich hier ein Flußdiagramm zur Erstellung des Programmes.

Flußdiagramm:



Programm:

```
00 INIT
03 LDC  4
04 XNOR 2
05 STOC 7
06 OEN  0
07 LD   4
08 STO  1
09 STOC 0
10 OEN  7
11 LD   1
12 ANDC 3
13 STO  0
14 LDC  1
15 AND  3
16 STO  1
17 JMP  x
```

9. Umcodierer

Die an den Eingän_en E1...E3 liegende Dualzahl soll in die entsprechende Dezimalzahl umcodiert werden. Die Nummer des Ausgangs, an dem eine "1" erscheint, ist gleich dieser Dezimalzahl.

Wahrheitstabelle:

E3	E2	E1	A0	A1	A2	A3	A4	A5	A6	A7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Logische Gleichungen:

A0 = $\overline{E1}*\overline{E2}*\overline{E3}$
A1 = $E1*\overline{E2}*\overline{E3}$
A2 = $\overline{E1}*E2*\overline{E3}$
A3 = $E1*E2*\overline{E3}$
A4 = $\overline{E1}*\overline{E2}*E3$
A5 = $E1*\overline{E2}*E3$
A6 = $\overline{E1}*E2*E3$
A7 = E1*E2*E3

Programm:

```
00 INIT
03 LDC  1
04 ANDC 2
05 ANDC 3
06 STO  0
07 LD   1
08 ANDC 2
09 ANDC 3
10 STO  1
11 LDC  1
12 AND  2
13 ANDC 3
14 STO  2
15 LD   1
16 AND  2
17 ANDC 3
18 STO  3
19 LDC  1
20 ANDC 2
21 AND  3
22 STO  4
23 LD   1
24 ANDC 2
25 AND  3
26 STO  5
27 LDC  1
28 AND  2
29 AND  3
30 STO  6
31 LD   1
32 AND  2
33 AND  3
34 STO  7
35 JMP  x
```

10. Zähler

Es soll nun ein Programm geschrieben werden, in dem die Anzahl der Eingänge (E1, E2, E3 und E4), die eine "1" anliegen haben, festgestellt werden soll. Die Nummer des Ausgangs, an dem eine "1" erscheint, soll gleich dieser Anzahl sein.

Wahrheitstabelle:

E4	E3	E2	E1	A0	A1	A2	A3	A4
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	0	0
0	0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0	0
0	1	0	1	0	0	1	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	1	0	0
1	0	1	0	0	0	1	0	0
1	0	1	1	0	0	0	1	0
1	1	0	0	0	0	1	0	0
1	1	0	1	0	0	0	1	0
1	1	1	0	0	0	0	1	0
1	1	1	1	0	0	0	0	1

Logische Gleichungen:

$$A0 = \overline{E1}*\overline{E2}*\overline{E3}*\overline{E4}$$

$$A1 = E1*\overline{E2}*\overline{E3}*\overline{E4}+\overline{E1}*E2*\overline{E3}*\overline{E4}+\overline{E1}*\overline{E2}*E3*\overline{E4}+\overline{E1}*\overline{E2}*\overline{E3}*E4 =$$
$$= (E1*\overline{E2}+\overline{E1}*E2)*\overline{E3}*\overline{E4}+(E3*\overline{E4}+\overline{E3}*E4)*\overline{E1}*\overline{E2} =$$
$$= (\overline{E1 \equiv E2})*\overline{E3}*\overline{E4}+(\overline{E3 \equiv E4})*\overline{E1}*\overline{E2} =$$
$$= \overline{(E1 \equiv E2)+E3+E4}+\overline{(E3 \equiv E4)+E1+E2} =$$
$$= \overline{((E1=E2)+E3+E4)*((E3=E4)+E1+E2)}$$

$$A2 = E1*E2*\overline{E3}*\overline{E4}+E1*\overline{E2}*E3*\overline{E4}+\overline{E1}*E2*E3*\overline{E4}+E1*\overline{E2}*\overline{E3}*E4+\overline{E1}*E2*$$
$$\overline{E3}*E4++\overline{E1}*\overline{E2}*E3*E4 =$$
$$= (E2*\overline{E3}+\overline{E2}*E3)*E1*\overline{E4}+(E2*\overline{E4}+\overline{E2}*E4)*\overline{E1}*E3+(E1*\overline{E2}+\overline{E1}*E2)*\overline{E3}*E4$$
$$= (\overline{E2 \equiv E3})*E1*\overline{E4}+(\overline{E2 \equiv E4})*\overline{E1}*E3+(\overline{E1 \equiv E2})*\overline{E3}*E4 =$$
$$= \overline{(E2 \equiv E3)+\overline{E1}+E4}+\overline{(E2 \equiv E4)+E1+\overline{E3}}+\overline{(E1 \equiv E2)+E3+\overline{E4}} =$$
$$= \overline{((E2=E3)+\overline{E1}+E4)*((E2=E4)+E1+\overline{E3})*((E1=E2)+E3+\overline{E4})}$$

$$A3 = E1*E2*E3*\overline{E4}+E1*E2*\overline{E3}*E4+E1*\overline{E2}*E3*E4+\overline{E1}*E2*E3*E4 =$$
$$= (E3*\overline{E4}+\overline{E3}*E4)*E1*E2+(E1*\overline{E2}+\overline{E1}*E2)*E3*E4 =$$
$$= (\overline{E3 \equiv E4})*E1*E2+(\overline{E1 \equiv E2})*E3*E4 =$$

$$A4 = E1*E2*E3*E4$$

Die logischen Gleichungen werden nacheinander in ein Programm umgesetzt. Zwischenergebnisse werden auf dem Ausgang A7 gespeichert. Da die Operation E1=E2 und E3=E4 mehrmals vorkommen, werden ihre Ergebnisse auf den Ausgängen A5 und A6 festgehalten.

Programm:

```
00 INIT ¡
03 LDC  1
04 ANDC 2
05 ANDC 3
06 ANDC 4 ⁿ
07 STO  0
08 LD   1
```

```
109 XNOR 2
 10 STO  5
 11 OR   3
 12 OR   4
 13 STO  7
 14 LD   3
 15 XNOR 4
 16 STO  6
 17 OR   1
 18 OR   2
 19 AND  7
 20 STOC 1
 21 LD   2
 22 XNOR 3
 23 ORC  1
 24 OR   4
 25 STO  7
 26 LD   2
 27 XNOR 4
 28 OR   1
 29 ORC  3
 30 AND  7
 31 STO  7
 32 LD   5
 33 OR   3
 34 ORC  4
 35 AND  7
 36 STOC 2
 37 LDC  6
 38 AND  1
 39 AND  2
 40 STO  7
 41 LDC  5
 42 AND  3
 43 AND  4
 44 OR   7
 45 STO  3
 46 LD   1
 47 AND  2
 48 AND  3
 49 AND  4
 50 STO  4
 51 JMP  x
```

4.5.5. Simulationsprogramme

In diesem Kapitel sollen weitere Programme wie z.B. Schaltungen behandelt werden.

1. Wechselschaltung

Eine Wechselschaltung ist eine Schaltung mit zwei Schaltern (S1 und S2). Wird ein Schalter betätigt, so ändert sich der Ausgangszustand Z der Schaltung. Der Anfangszustand ist willkürlich gewählt.

Festlegung der Zuordnungen im Programm:

S2 = E2, S1 = E1, Z = A0

Wahrheitstabelle:

E2	E1	A0
0	0	0
0	1	1
1	1	0
1	0	1

Logische Gleichung:

$A0 = \overline{E1 \equiv E2}$

Programm:

```
00 INIT
03 LD   1
04 XNOR 2
05 STOC 0
06 JMP  x
```

Stellung: Schnell-Takt

2. Kreuzschaltung

Eine Kreuzschaltung ist eine Schaltung mit mehreren Schaltern (hier drei: S1, S2 und S3). Wird ein Schalter betätigt, so ändert sich der Ausgangszustand Z der Schaltung. Der Anfangszustand ist wieder willkürlich gewählt.

Festlegung der Zuordnungen im Programm:

S3 = E3, S2 = E2, S1 = E1, Z = A0

Wahrheitstabelle:

E3	E2	E1	A0
0	0	0	0
0	0	1	1
0	1	1	0
0	1	0	1
1	1	0	0
1	1	1	1
1	0	1	0
1	0	0	1

Logische Gleichung:

$A0 = E1 \equiv E2 \equiv E3$

Programm:

```
00 INIT
03 LD   1
04 XNOR 2
05 XNOR 3
06 STO  0
07 JMP  x
```

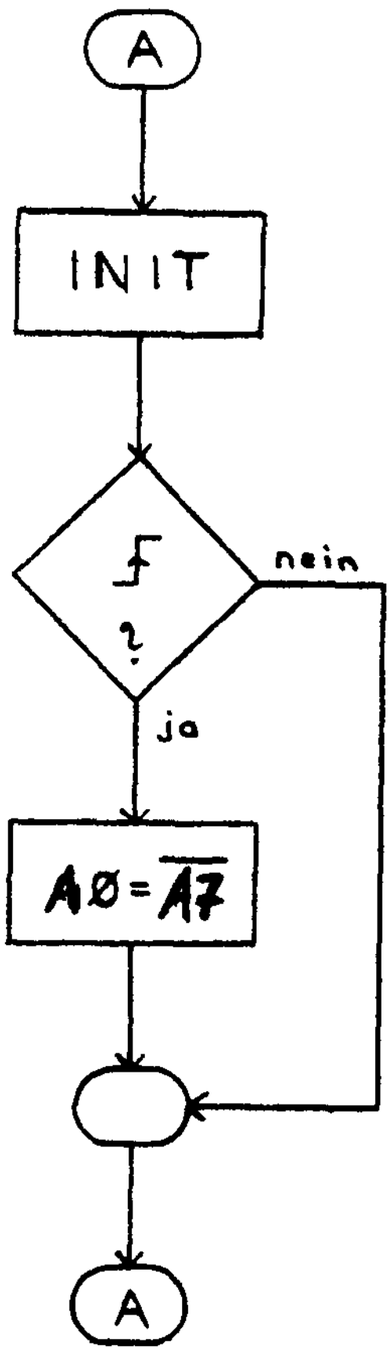
Stellung: Schnell-Takt

3. Tasterschaltung

Die Tasterschaltung ist eine Schaltung mit mehreren Tastern (hier vier: T1, T2, T3 und T4). Wird ein Taster kurz geschlossen, so ändert sich der Ausgangszustand Z der Schaltung.

Festlegung der Zuordnungen im Programm:

T4 = E4, T3 = E3, T2 = E2, T1 = E1, Z = A0



Flußdiagramm nebenstehend!

Programm:

```
00 INIT
03 LD   5
04 STO  6
05 LD   1
06 OR   2
07 OR   3
08 OR   4
09 STO  5
10 ANDC 6
11 OEN  0
12 LD   7
13 STOC 7
14 STOC 0
15 JMP  x
```

Stellung: Schnell-Takt

4. Zwei-aus-drei-Schaltung

Zwei-aus-drei-Schaltungen werden bei mit Risiken behafteten Anlagen verwendet. Eine Abschaltung der Anlage soll nur dann erfolgen, wenn mindestens zwei der drei Gefahrenmelder (G1, G2 und G3) die Gefahr anzeigen, da Gefahrenmelder auch defekt sein können. Die Gefahrenmelder geben bei Gefahr eine "1". Die Abschaltung der Anlage soll erfolgen, wenn am Ausgang Z der Zwei-aus-drei-Schaltung eine "1" erscheint.

Festlegung der Zuordnungen im Programm:

G3 = E3, G2 = E2, G1 = E1, Z = A0

Wahrheitstabelle:

E3	E2	E1	A0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Logische Gleichungen:

$A0 = (E1+E2)*E3+\overline{E1*E2}$

Programm:

```
00 INIT
03 LD   1
04 OR   2
05 AND  3
06 IEN  1
07 OR   2
08 STO  0
09 JMP  x
```

Stellung: Schnell-Takt

5. Multivibrator

Flipflops sind bistabile Kippstufen, d.h., sie haben zwei stabile Ausgangszustände (Q und $\bar{Q}$). Multivibratoren sind astabile Kippstufen, d.h., sie haben keinen stabilen Ausgangszustand. Sie kippen von einem nichtstabilen Zustand in den anderen nichtstabilen Zustand und wieder zurück. Zum Kippen ist kein von außen kommendes Signal erforderlich.

Festlegung der Zuordnungen im Programm:

Q = A0, $\bar{Q}$ = A1

Programm:

```
00    INIT
03    LD    /
04    STOC  7
05    STOC  0
06    STO   1
07    NOP   x
.
.
.
yy    NOP   x
yy+1  JMP   x
```

yy ist eine beliebige Programmadresse. yy+1 ist die auf yy folgende Programmadresse.
Durch die Anzahl der Leerbefehle NOP kann die Frequenz variiert werden. Bei hohen Frequenzen ist die Stellung "Schnell-Takt" zu wählen und bei niedrigen Frequenzen ist die Stellung "Langsam-Takt" zu wählen, wobei im letzteren Fall die Frequenz noch zusätzlich durch das Potentiometer verändert werden kann. Das Ausgangssignal ist ein Rechtecksignal , da die beiden Zustände Q = "0" und Q = "1" gleich lange dauern (Siehe: Kapitel 4.3.2.).

6. Monoflop

Monoflops sind monostabile Kippstufen, d.h., sie haben einen stabilen und einen nichtstabilen Ausgangszustand. Ein Kippen der Schaltung in den nichtstabilen Zustand ist nur durch ein von außen zugeführtes Signal S möglich. Im stabilen Zustand liegt am Ausgang Q eine "0". Im nichtstabilen Zustand liegt dort eine "1".

Festlegung der Zuordnungen im Programm:

S = E1, Q = A0, $\overline{Q}$ = A1

Programm:

```
00 INIT
03 LD   5
04 STO  6
05 LDC  1
06 STO  5
07 ORC  6
08 SKZ  x
09 JMP  x
10 STOC 0
11 STO  1
12 NOP  x
.
.
.
yy    NOP  x
yy+1 STO  0
yy+2 STOC 1
yy+3 JMP  x
```

Durch die Anzahl der Leerbefehle NOP kann die Dauer des nichtstabilen Zustandes variiert werden. Danach kippt die Schaltung selbständig in den stabilen Zustand zurück. Es kann "Schnell-Takt" oder "Langsam-Takt" gewählt werden.

7. Elektronischer Würfel

Es soll ein elektronischer Würfel dargestellt werden. Ein Schalter soll den Würfelvorgang ersetzen. Die Nummer des Ausgangs, an dem beim Betätigen des Schalters (E1) eine "1" erscheint, ist gleich der "gewürfelten" Zahl. Zunächst wird ein Zähler gebildet, der immer von 1 bis 6 zählt. Wird dann der Eingang E1 auf "1" gesetzt, so soll der derzeitige Zählerstand erhalten bleiben. Damit alle Ausgangszustände, d.h. alle Zahlen, gleich wahrscheinlich sind, muß jeder Zählerstand gleich lange an den Ausgängen stehen.

Programm:

```
00 INIT
03 LD   1
04 SKZ  x
05 JMP  x
06 STOC 1
07 STO  6
08 NOP  x
09 NOP  x
10 NOP  x
11 NOP  x
12 LD   1
13 SKZ  x
```

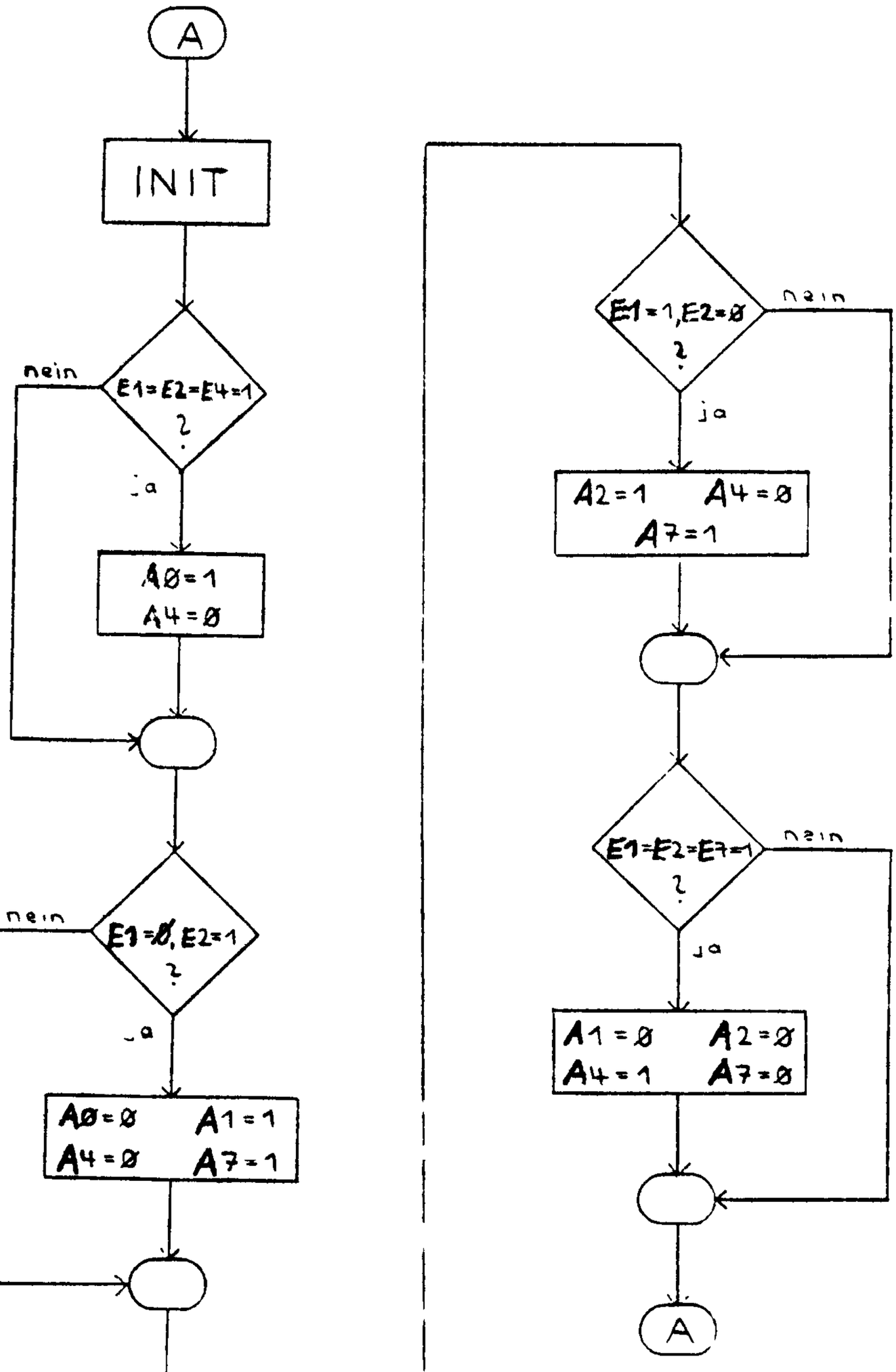
```
14 JMP   x
15 STOC  2
16 STO   1
17 NOP   x
18 NOP   x
19 NOP   x
20 NOP   x
21 LD    1
22 SKZ   x
23 JMP   x
24 STOC  3
25 STO   2
26 NOP   x
27 NOP   x
28 NOP   x
29 NOP   x
30 LD    1
31 SKZ   x
32 JMP   x
33 STOC  4
34 STO   3
35 NOP   x
36 NOP   x
37 NOP   x
38 NOP   x
39 LD    1
40 SKZ   x
41 JMP   x
42 STOC  5
43 STO   4
44 NOP   x
45 NOP   x
46 NOP   x
47 NOP   x
48 LD    1
49 SKZ   x
50 JMP   x
51 STOC  6
52 STO   5
53 JMP   x
Stellung: Schnell-Takt
```

8. Fließbandsteuerung

Der Antrieb eines Fließbandes soll so gesteuert werden, daß Pakete, die auf diesem Fließband laufen, zu einer Entnahmestelle gebracht werden. Die Entnahmestelle befindet sich auf dem Fließband zwischen zwei Lichtschranken (L1 und L2). Die Steuerung des Fließbandes geschieht mit Hilfe dieser beiden Lichtschranken und einem Entnahmegerät. Der Abstand der beiden Lichtschranken voneinander ist etwas größer als die größtmögliche Breite eines Paketes. Normalerweise läuft das Fließband schnell vorwärts bis ein Paket die erste Lichtschranke unterbricht. Dann läuft das Fließband langsam vorwärts. Ist die Lichtschranke wieder geschlossen, so stoppt das Fließband und das Entnahmegerät bekommt einen Impuls zum Entnehmen des Paketes vom Fließband. Ist das Paket richtig entnommen, so wird das Fließband durch einen Impuls vom Entnahmegerät auf "Schnell-

Vorwärts" gestellt. Ist jedoch das Paket wieder auf das Fließband gefallen bzw. gar nicht entnommen worden, so muß es wieder an die Entnahmestelle gebracht werden, wenn eine der beiden Lichtschranken unterbrochen wurde. Die erste Lichtschranke (L1) steuert das Paket langsam vorwärts zur Entnahmestelle und die zweite Lichtschranke (L2) steuert das Paket langsam rückwärts.
Ist eine Lichtschranke geschlossen, entspricht das einer "1".
Ist eine Lichtschranke unterbrochen, entspricht das einer "0". Impulse werden durch eine "1" dargestell .

Festlegung der Zuordnungen im Programm:
Impuls vom Entnahmegerät = E4, Lichtschranke 2 (L2) = E2, Lichtschranke 1 (L1) = E1
Fließband:
Schnell-Vorwärts = A0, Langsam-Vorwärts = A1, Langsam-Rückwärts = A2, Impuls für Entnhmegerät = A4
Flußdiagramm:



```
Programm:
00 INIT
03 LD   1
04 AND  2
05 AND  3
06 OEN  0
07 STOC 4
08 STO  0
09 LD   2
10 ANDC 1
11 OEN  0
12 STOC 0
13 STOC 4
14 STO  1
15 STO  7
16 LD   1
17 ANDC 2
18 OEN  0
19 STOC 4
20 STO  2
21 STO  7
22 LD   7
23 AND  1
24 AND  2
25 OEN  0
26 STOC 1
27 STOC 2
28 STO  4
29 STOC 7
30 JMP  x
```

Stellung: Schnell-Takt

4.6. PROGRAMME ZUR STEUERUNG ANGESCHLOSSENER PERIPHERIE

In Kapitel 4.5.5. haben wir unter anderem eine Fließbandsteuerung simuliert. Wir haben also kein Fließband angeschlossen, sondern nur die Steuerungsvorgänge an den Leuchtdioden beobachtet.
In diesem Kapitel wollen wir nun z. B. einen Motor an den Computer anschließen und die Steuerungsvorgänge direkt beobachten. Geräte, die an einen Computer angeschlossen werden, nennt man Peripherie.

Wie schon vorher dargelegt, ist es eines der wesentlichen Kennzeichen der dritten industriellen Revolution, daß Computer die Steuerung von Maschinen übernehmen. Genau das kann auch unser Computer. Grundlage für den Antrieb von Robotern sind Elektromotoren, genauer gesagt, Schrittmotoren. Beginnen wir daher mit der Steuerung eines Elektromotors.
Alle Elektromotoren müssen mit weit höheren elektrischen Strömen betrieben werden, als unser Computer sie an seinen Ausgängen liefert. Daher schalten wir einen oder auch mehrere Transistorverstärker, ein sogenanntes Interface, zwischen Computer und Motor. Das Interface, an dem der Motor angeschlossen ist, wird mit dem Expansionsstecker verbunden.

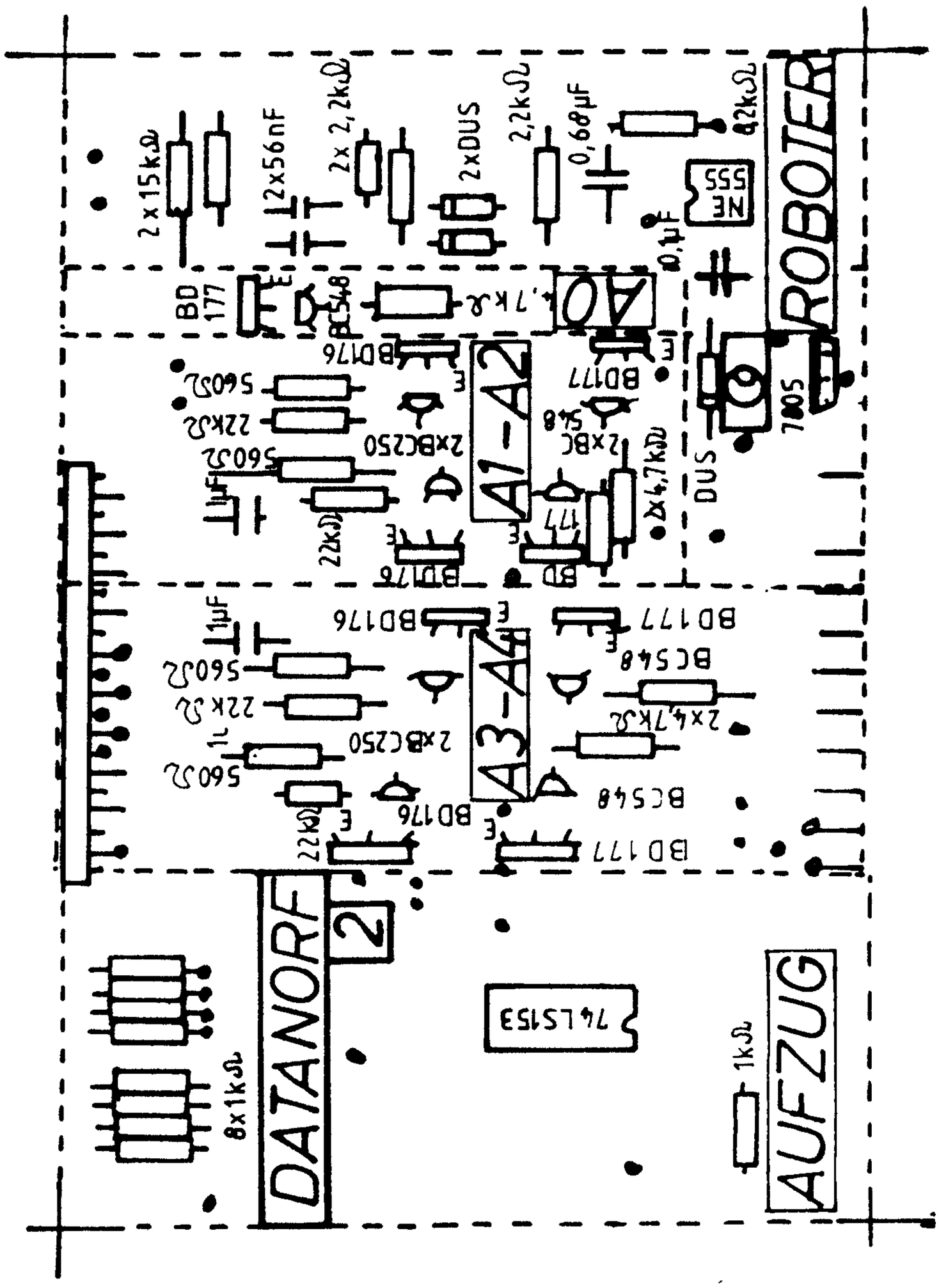


Abb. 42: Verstärker für den Anschluß an den WDR-1-Bit Computer

Um möglichst vielseitig zu sein, benutzen wir ein speziell für den WDR-1-Bit-Computer entwickeltes Interface für Elektromotoren, das DATANorf-Interface (vergl. Bezugsquellen). Dieses Interface bietet den Vorteil, daß es je nach Bedarf ausgebaut bzw. erweitert werden kann. In der ersten Ausbaustufe "das Schalten eines Gerätes" wird das Feld A0 bestückt. Die zweite Ausbaustufe, Feld A1/A2 ist schon leistungsfähiger. Beispielsweise kann damit ein Motor umpolet werden. Insgesamt können wir mit den vier vorhandenen Ausbaustufen alle Geräte aus dem Fischertechnik computing- Baukasten steuern.

4.6.1. MOTORSTEUERUNG

Ziel ist es zunächst, eine Radarantenne zu steuern. Eine Radarantenne dreht sich ein Stück in eine bestimmte Richtung, stoppt dann und dreht sich ein Stück in die entgegengesetzte Richtung. Dazu müssen wir einen Elektromotor ein- und ausschalten, sowieumpolen. Umpolen heißt, wir machen aus einer Linksdrehung des Motors eine Rechtssdrehung, bzw. umgekehrt. Die Radaranlage bauen wir aus Legobauteilen (Siehe Abb. 43) auf.

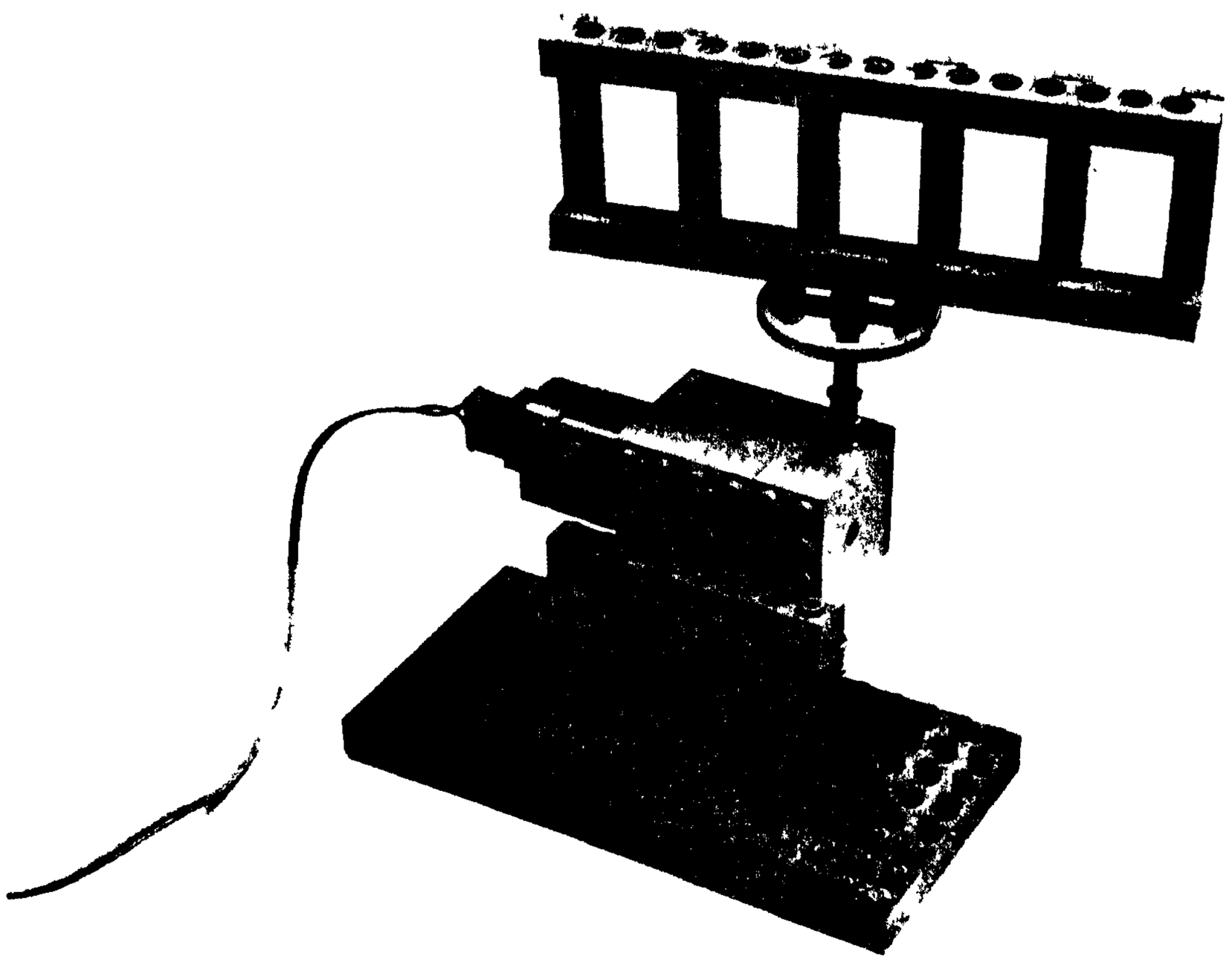


Abb. 43 Radarantenne

Den Motor verdrahten wir mit dem Ausgang A0 des Interfaces. Zunächst schalten wir den Motor an und aus.

```
** Motor schalten **
* Betriebsart: Handtakt *
00 INIT
03 STO  0     Motor wird eingeschaltet
04 STOC 0     Motor wird ausgeschaltet
05 JMP  x
```

Soll das Programm automatisch ablaufen, stellen wir den Taktschalter auf Langsamtakt und fügen "Verzögerungen" ein, da der Motor sonst den schnellen Befehlsänderungen nicht folgen kann:

```
** Motor schalten **
* Betriebsart: Langsamtakt *
00    INIT
03    STO  0     Motor wird eingeschaltet
04    NOP  0     Bei diesem Schritt verändert sich nichts: es
                 entsteht eine Pause.
zz    NOP  0
zz+1  STOC 0   Motor wird ausgeschaltet
zz+2  JMP  x
```

(Es ist darauf zu achten, daß die Anzahl der Programmschritte hier zz+2, 256 nicht überschreitet)

Mit Hilfe des nächsten Programms können wir den Motor umpolen. Da das Feld A0 des Interfaces einen Motor oder Elektromagne en nur ein- bzw. ausschalten kann, verdrahten wir nun den Motor mit den Ausgängen des Feldes A1-A2.
Damit sind die Zuordnungen im Programm festgelegt:

Motor linksdrehen = A1, Motor rechtsdrehen = A2.

```
** Motor umpolen **
* Betriebsart: Langsamtakt *
00 INIT
03 STO  1     Motor linksdrehen an
04 STOC 1     Motor aus
05 STO  2     Motor rechtsdrehen an
06 STOC 2     Motor aus
07 JMP  x
```

(Sollte die Drehrichtung des Motors nicht mit dem Programm übereinstimmen, muß der Motor umgepolt werden.)
In dieses Programm können natürlich wieder "Verzögerungen" eingebracht werden, so daß eine Radarantenne richtig simuliert wird. Außerdem ist darauf zu achten, daß der Motor vor dem Umschalten durch das Programm ausgeschaltet wird. Andernfalls zerstören wir die Transistoren unseres Interfaces.

4.6.2. SORTIERANLAGE

Eine Anlage, die aus Fischertechnikbauteilen zusammensetzt ist (Siehe: Abb. 44), soll "Werkstücke" mit zwei verschiedenen Längen sortieren. Sortiert wird, indem ein Schlitten mit Hilfe eines Motors bewegt wird.
Die Längenabfrage geschieht mittels zwei parallel geschalteter Schalter S1 und S2 mit einem gemeinsamen Ausgang. Der Ausgang ist über diese Schalter mit 5 V verbunden. In Ruhestellung liegt der Ausgang auf "1", da beide Schalter in diesem Zustand

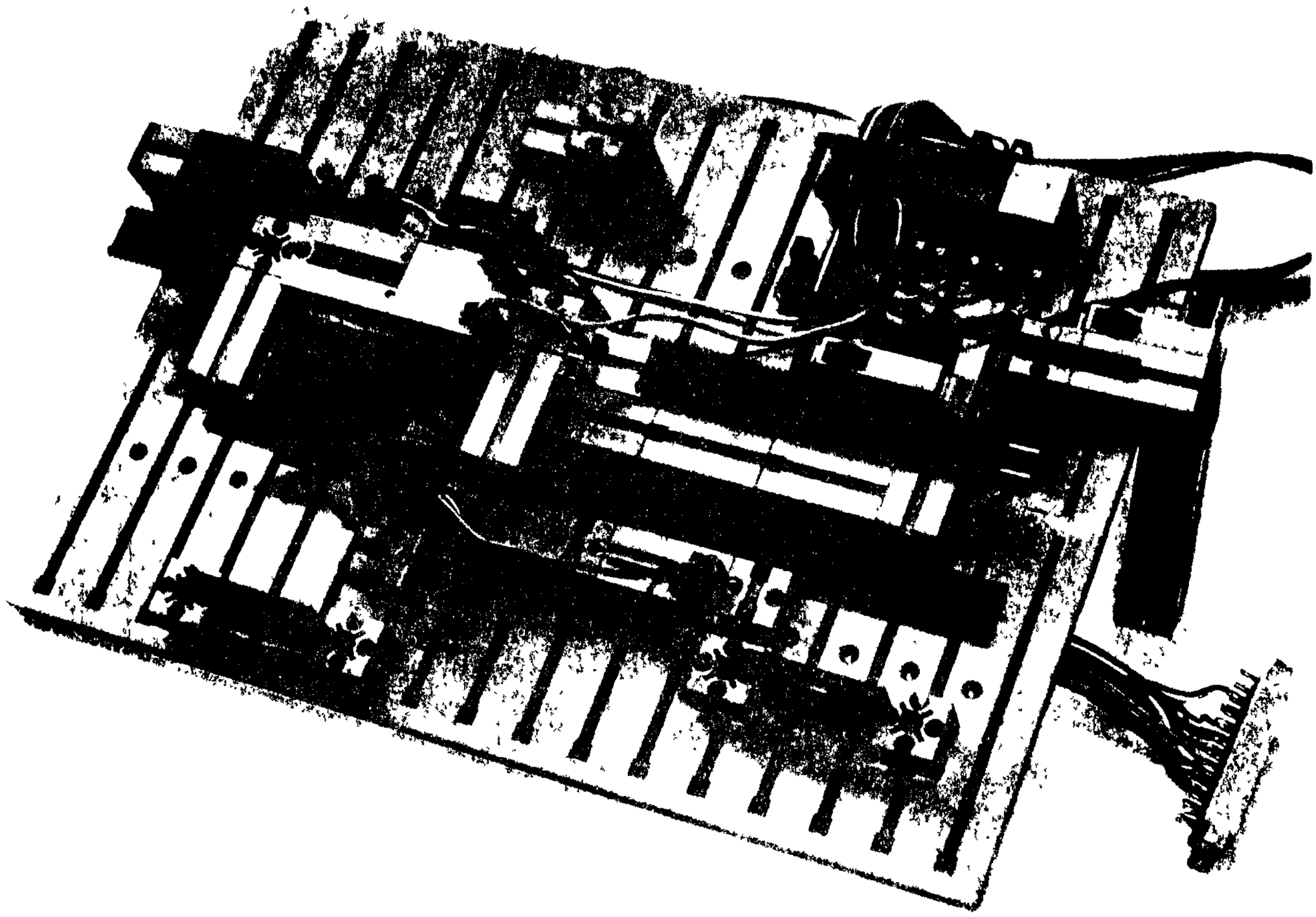


Abb. 44: Sortieranlage

geschlossen sind. Wird einer von ihnen geöffnet, ändert sich am Ausgang zunächst nichts, da sie im Sinne einer ODER-Schaltung miteinander verdrahtet sind. Damit am Ausgang eine "0" anliegt, wenn beide Schalter geöffnet sind, muß der Ausgang noch über einen Pull down Widerstand von der Größe 1 Kiloohm mit Masse verbunden sein. Der Ausgang dieser Schaltung ist mit dem Eingang E3 des Computers verbunden. Abb. 45 zeigt die Schaltung für die Längenabfrage.

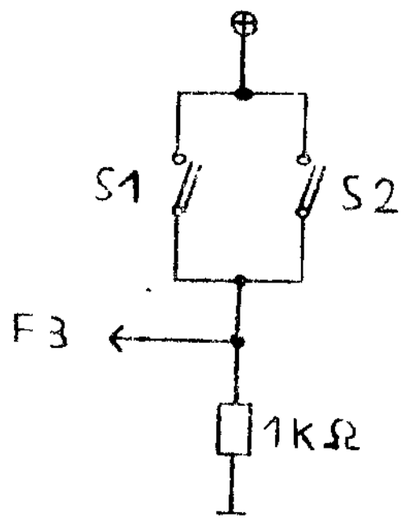


Abb. 45: Schaltung für die Längenabfrage

Die Längenabfrage erkennt einen langen Baustein daran, daß beim Transport des Bausteins beide Schalter betätigt werden, also eine "0" anliegt. Beim Transport eines kurzen Bausteins ist immer einer der beiden Schalter geöffnet, es liegt also eine "1" an.

Hier die Festlegung der Zuordnungen für die Abfrage des Schlittens im Programm:

Start des Schlittens und stoppen rechts = E4, Längenabfrage = E3,
Stoppen des Schlitten links = E2, Schalter zum Abschalten = E1,

Motor linksdrehen = A1, Motor rechtsdrehe = A2.

```
** Sortieranlage **

* Definition der Prozeduren *
* In Prozedur WARTEN            sind        E/A6 = 0 und E/A7 = 0
* In Prozedur TESTEN            sind        E/A6 = 1 und E/A7 = 0
* In Prozedur LÄNGE ERKENNEN    sind        E/A6 = 0 und E/A7 = 1
* In Prozedur ZURÜCKFAHREN      sind        E/A6 = 1 und E/A7 = 1

* Werkstücke sortieren *
00 INIT
* Prozedur WARTEN
03 LDC  7
04 ANDC 6
05 LD   4
06 STO  1
07 STO  6
* In Prozedur TESTEN
08 LDC  7
09 AND  6
10 ANDC 3
11 OEN  0
12 STOC 1
13 STO  2
14 STOC 6
15 STO  7
16 LD   2
* In Prozedur LÄNGE ERKENNEN
17 AND  6
18 ANDC 7
19 OEN  0
20 STOC 1
21 STO  2
22 STO  6
23 STO  7
* In Prozedur ZURÜCKFAHREN
24 LDC  6
25 AND  7
26 AND  4
27 OEN  0
28 STOC 2
29 STO  1
30 STO  6
31 LD   7
32 AND  6
33 AND  1
34 OEN  0
```

```
35 STOC 1
36 STOC 2
37 STOC 6
38 STOC 7
39 JMP  x
```

4.6.3. STEUERUNG EINES ROBOTERS

Bauen wir unsere Sortieranlage zu einem "Teach-in"-Roboter um. Er soll lernen, Werkstücke an einem bestimmten Platz aufzunehmen (Materialeinkauf), sie zu einer anderen Stelle hinzutransportieren (Produktion) und sie von hier an eine dritte Stelle zu bringen (Versand).
Der Teach-in-Roboter hat für diese Aufgabenstellung zwei Motoren, einen Elektromagneten und zwei Potentiometer. Gegenüber dem von Fischertechnik angegebenen Aufbauvorschlag haben wir ihn jedoch leicht verändert: Wir haben die beiden Potentiometerabfragen weggelassen, da unser Computer nicht in der Lage ist, die Ergebnisse von Analog-Digital-Wandlern zu verarbeiten. Wir müssen daher einen anderen Weg gehen. Für die Abfrage des Drehwinkels setzen wir auf die Schneckenwelle eine Seiltrommel mit Doppelnocke, die einen darunter montierten Schalter betätigt. Um die Lage des Schlittens abzufragen, fügen wir unter die Aluminiumschiene 10 Winckelstücke ein (vgl. Bezugsquellen), die wiederum einen Schalter betätigen und so die Lage des Schlittens dem Computer mitteilen. Beiden Abfragen ist gemeinsam, daß sie nicht mehr den Anfangs- bzw. Endzustand des zu bewegenden Teils registrieren, sondern daß die Anzahl der Schalterbetätigungen abgezählt wird. Allerdings besitzt der WDR-1-Bit-Computer hierfür keinen speziellen Zähler. Sein einziger Zähler ist der Programmzähler, der für diese Vorgänge mitbenutzt wird, wollen wir keinen zusätzlichen Zähler aufbauen. Es reicht aus, das DATANorf-Interface im Feld "Robot" um einen Timer NE 555 mit dazugehörenden passiven Bauteilen zu ergänzen. Der NE 555 ist hierbei so beschaltet, daß er die beiden Schalter des Roboters entprellt und so nur Einzelimpulse an den Eingang des Computerprogrammzählers gelangen. Wenn wir den Programmzähler als Impulszähler verwenden wollen, müssen wir dafür sorgen, daß der automatisch ablaufende Langsam-Takt des Computers abgestellt wird. Dies erreichen wir, indem wir über den Anschluß 20 des Expansionssteckers an den Pin 4 des Taktgenerators eine "0" legen. Dies geschieht automatisch durch Anstecken des DATANorf-Interfaces, das eine Verbindung zum Ausgang A5 des Computers herstellt.
Nach der Initialisierung muß dann zuerst der Langsam-Takt wieder freigegeben werden. Dies geschieht mit dem Befehl STO 5. Jetzt arbeitet der Computer wieder wie gewohnt.
Hier das Prinzip des TEACH-IN: Mit STOC 5 stellen wir den Langsamtakt ab. Jeder Zählimpuls aus einem der beiden Schalter gelangt nun über den Anschluß 18 des Expansionssteckers an den Prozessor und damit an seinen Programmzähler. Jedem Zählimpuls wird ein NOP-Befehl zugeordnet. Soll dieser Vorgand beendet werden, geben wir STO 5 ein. Auf diese Art und Weise läßt sich jeder "Koordinatenpunkt" im Bewegungsraum des Elektromagneten erreichen, wir müssen nur die entsprechenden Programmschritte eingeben.

Das folgende Programm veranlaßt den Roboter zwei verschieden große Metallstücke mit Hilfe des Elektromagneten zu versetzen.
Damit wir den Roboter leichter in eine durch ein Programm definierte Ausgangslage bringen können (in unserem Beispiel ist die Startposition links unten), enthält er noch zwei weitere Schalter, die die beide Motoren ausschalten können.

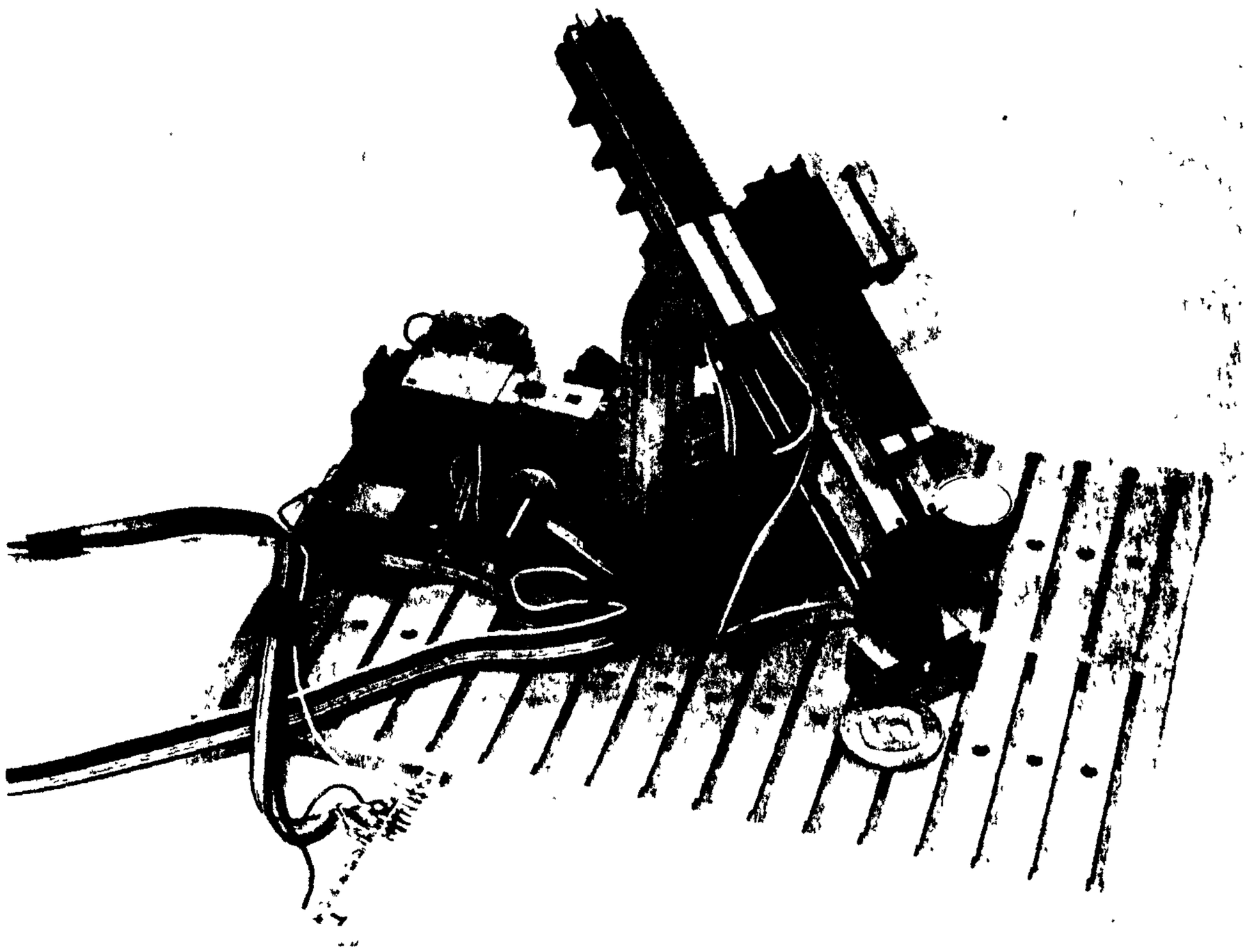


Abb. 46: Teach-in-Roboter mit digitaler Abfrage

Da das Programm sehr lang ist, ist die Erklärung des Programmes gleich bei den Programmschritten beigefügt.

```
***"TEACH-IN"-ROBOTER ***
** Zuordnungen **
* Startposition     :links unten
* Elektromagnet     :A0
* Rechtsdrehen      :A1
* Linksdrehen       :A2
* Tieffahren        :A3
* Hochfahren        :A4
* Wenn A5 = 1       :Takt vom Timer des Computers
* Wenn A5 = 0       :Takt vom Roboter
*
00 INIT                  * Magnet an              * Magnet ein
* Magnet ein              * hoch ein               * hoch ein
* hoch ein                73 STO  5                145 STO  5
03 STO  5                74 STOC 3                146 STOC 3
04 STO  0                75 STO  0                147 STO  0
05 STO  4                76 STO  4                148 STO  4
06 STOC 5                77 STOC 5                149 STOC 5
07 NOP  x /4 mal         78 NOP  x /4 mal         150 NOP  x /4 mal
```

```
* hoch aus
* rechts ein
11 STO  5
12 STOC 4
13 STO  1
14 STOC 5
15 NOP  x /20 mal
* rechts aus
* tief an
35 STO  5
36 STOC 1
37 STO  3
38 STOC 5
39 NOP  x /4 mal
* tief aus
* Magnet aus
* hoch an
43 STO  5
44 STOC 3
45 STOC 0
46 STO  4
47 STOC 5
48 NOP  x /4 mal
* hoch aus
* links an
52 STO  5
53 STOC 4
54 STO  2
55 STOC 5
56 NOP  x /9 mal
* links aus
* tief an
65 STO  5
66 STOC 2
67 STO  3
68 STOC 5
69 NOP  x /4 mal
* tief aus
```

```
* hoch aus
* links ein
82 STO  5
83 STOC 4
84 STO  2
85 STOC 5
86 NOP  x /10 mal
* links aus
* tief an
 96 STO  5
 97 STOC 2
 98 STO  3
 99 STOC 5
100 NOP  x /4 mal
* tief aus
* Magnet aus
* hoch an
104 STO  5
105 STOC 3
106 STOC 0
107 STO  4
108 STOC 5
109 NOP  x /4 mal
* hoch aus
* rechts an
113 STO  5
114 STOC 4
115 STO  1
116 STOC 5
117 NOP  x /20 mal
* rechts aus
* tief ein
137 STO  5
138 STOC 1
139 STO  3
140 STOC 5
141 NOP  x /4 mal
* tief aus
```

```
* hoch aus
* links ein
154 STO  5
155 STOC 4
156 STO  2
157 STOC 5
158 NOP  x /9 mal
* links aus
* tief an
167 STO  5
168 STOC 2
169 STO  3
170 STOC 5
171 NOP  x /4 mal
* tief aus
* Magnet aus
* hoch an
175 STO  5
176 STOC 3
177 STOC 0
178 STO  4
179 STOC 5
180 NOP  x /4 mal
* hoch aus
* links an
184 STO  5
185 STOC 4
186 STO  2
187 STOC 5
188 NOP  x /10 mal
* links aus
* tief ein
198 STO  5
199 STOC 2
200 STO  3
201 STOC 5
202 NOP  x /4 mal
* tief aus
205 STO  5
206 STOC 3
207 JMP  x
```

Ein Programm mit nur wenig verschiedenen Programmbefehlen, aber vielen Programmschritten. Es eignet sich für die Durchführung eines Projekts "Robotics". Die einzelnen Programmabschnitte können arbeitsteilig geschrieben, eingegeben, ausgetestet und koordiniert werden. Das Programm kann in einem EPROM gespeichert bezogen werden, (vergl. Bezugsquellen). Dann ist der Gesamtablauf des Programms im Unterricht realistisch.
Ein Programm, das ohne ein EPROM zu benutzen in eine Unterichtsreihe paßt, wenn es verkürzt eingegeben wird, ist das folgende.

4.6.4. AMPELSTEUERUNG

LS241 → LS240

Natürlich können wir eine Ampelanlage auf unseren Ausgangsleuchtdioden simulieren. Allerdings fällt die Zuordnung zu den einzelnen Signallampen schwer. Um also ein möglichst naturgetreues Modell zu erhalten, greifen wir wieder auf Bauelemente des Fischertechnik computing-Baukastens zurück. Als Treiber für die Signallampen benutzen wir z.B. das IC SN 74LS241 auf einer Pufferplatine (vgl. Bezugsquellen).
Unsere Ampel ist eine Fußgängerampel, d.h. sie stoppt die Autofahrer, wobei sie gleichzeitig den Fußgängern grünes Licht gibt und umgekehrt. Ein Taster ermöglicht die Betätigung durch einen Fußgänger, sie läuft aber auch automatisch auf Langsamtakt, wenn wir den Eingangsschalter E1 betätigen. Jede Signallampe ist so geschaltet, daß eine "0" an einem Ausgang diese anschaltet! Wird die Ampelanlage mit der Pufferplatine verbunden, sind somit zunächst alle Lampen angeschaltet. Dies ändert sich erst mit Ablauf des Programms .

```
*** Ampel 1 ***
**Festlegung der Zuordnungen im Programm **
* Eingang E1       :Tasterabfrage
* Schalter E1      :Dauerbetrieb
* Druckknopf für den Fußgänger = E1
* Signale fur die Fußgänger
* Ausgang A0       :Grün
* Ausgang A1       :Gelb
* Ausgang A2       :Rot
*
* Signale für die Autofahrer
* Ausgang A3       :Grün
* Ausgang A4       :Gelb
* Ausgang A5       :Rot
*
00  INIT
* Tasterabfrage *
03  STOC 3      Grün für die Autofahrer
04  STOC 2      Rot für die Fußgänger
05  LDC  1      Taster abfragen
06  SKZ  x      War der Taster nicht gedrückt, ist das
                Ergebnisregister = 1, der nächste Befehl wird
                 usgeführt. Dadurch wird an den Programmanfang
                "Tasterabfrage" gesprungen. Wurde der Taster
                gedrückt oder wurde der Schalter E1 betätigt, so
                ist das Ergebnisregister = 0, der nächste Befehl
                wird ignoriert. Dadurch wird zur Routine
                "Signalumschaltung" gesprungen.
07  JMP  x
* Signalumschaltung *
08  ORC  0      Das Ergebnisregister ist "0", wenn der vorherige
                Zweig durchlaufen wird.
09  STO  3      Grün für die Autofahrer ausschalten
10  STOC 4      Gelb für die Autofahrer
11  STOC 1      Gelb für die Fußgänger
12  NOP  x      10 Takte Pause
21  NOP  x
```

```
22  STO  4      Gelb für die Autofahrer ausschalten
23  STOC 5      Rot für die Autofahrer
24  STO  2      Rot für die Fußgänger ausschalten
25  STO  1      Gelb für die Fußgänger ausschalten
26  STOC 0      Grün für die Fußgänger
27  NOP  x      60 Takte Pause
86  NOP  x
87  STO  0      Grün für die Fußgänger ausschalten
88  STOC 1      Gelb für die Fußgänger
89  STOC 4      Gelb für die Autofahrer
90  NOP  x      10 Takte Pause
99  NOP  x
100 STO  1      Gelb für die Fußgänger ausschalten
101 STOC 2      Rot für die Fußgänger
102 STO  5      Rot für die Autofahrer ausschalten
103 STO  4      Gelb für die Autofahrer ausschalten
104 STOC 3      Grün für die Autofahrer
105 NOP  x      25 Takte Pause
129 NOP  x
130 JMP  x
```

Läßt man die Pausen weg, so kann das Programm, die Pausen für die einzelnen Signalphasen berücksichtigend, im Hand-Takt durchlaufen werden. Soll dagegen das komplette Programm in den Computer geladen werden, empfiehlt sich wieder die EPROM-Version (siehe Bezugsquellen).

Das folgende Ampelprogramm berücksichtigt die Tatsache, daß der Fußgänger kein Gelb bekommt.

```
***Ampel 2***
**Festlegung der Zuordnungen im Programm**
*Druckknopf für den Fußgänger = E1
*Lichtsignale für den Fußgänger: Grün = A0. Rot = A1
*Lichtsignale für die Autofahrer: Grün = A2, Gelb = A3, Rot = A4
*Bemerkungen: F = Fußgänger, A = Autofahrer

00 INIT
03  STO  0
04  STO  3      A. Grün an
05  STO  4      F. Rot an
06  LDC  1
07  SKZ  x
08  JMP  x
09  ORC  0
10  STO  2      A. Grün aus
11  STOC 3      A. Gelb an
12  NOP  x
.
.   NOP  x      Gelbphase (20 Takte)
.
31  NOP  x
32  STO  3      A. Gelb aus
33  STOC 4      A. Rot an
34  NOP  x
.
.   NOP  x      Fußgängerampel springt nicht sofort um (10 Takte)
```

falsch

```
43  NOP  x
44  STO  1     F. Rot aus
45  STOC 0     F. Grün an
46  NOP  x
.
.   NOP  x     F. Grünphase (60 Takte)
.
105 NOP  x
106 STO  0     F. Grün aus
107 STOC 1     F. Rot an
108 NOP  x
.
.   NOP  x     Autofahrerampel springt nicht sofort um, damit
               auch Fußgänger, die sich noch auf der Straße
               befinden, noch gefahrlos über die Straße können
               (30 Takte).
137 NOP  x
138 STOC 3     A. Gelb an
139 NOP  x
.
.   NOP  x     Rot-Gelb-Phase (10 Takte)
.
148 NOP  x
149 STO  4     A. Rot aus
150 STO  3     A. Gelb aus
151 STOC 2     A. Grün an
152 NOP  x
.
.   NOP  x     A. Grünphase (104) Takte
.
255 NOP  x     Lange Grünphase, damit nicht durch dauerndes
               Betätigen des Druckknopfs der Autoverkehr
               blockiert wird.
```

Auch für dieses Programm gilt wieder: läßt man die Pausen weg, so kann das Programm, die Pausen für die einzelnen Programmphasen berücksichtigend, im Handtakt durchlaufen werden. Soll dagegen das komplette Programm in den Computer geladen werden, empfiehlt sich wieder die EPROM-Version (vergl. Bezugsquellen).

4.6.5. MORSEZEICHENGENERATOR

Ein Morsezeichengenerator (vgl. Bezugsquellen) arbeitet mit dem bekannten NE 555. Die bestückte Platine wird mit einer Buchsenleiste versehen (Siehe: Abb.47). Die Stromversorgung für den Morsezeichengenerator wird von den Anschlüssen 1 und 2 des Peripheriesteckers an die mit "+" und "-" gekennzeichneten Stellen geführt, der Anschluß 4 des NE 555 wird z.B. mit Ausgang A0 des Computers verbunden. STO 0 schaltet den Tonerzeuger ein und STOC 0 schaltet ihn aus. Nach der Initialisierung bewirkt ein STO 0 und dahinter ein STOC 0 einen Morsepunkt, dreimal STO 0 und dahinter ein STOC 0 ergibt einen Morsestrich. Eine Buchstabenpause entsteht durch Anfügen von zwei STOC 0-Programmschritten. Der Morsezeichengenerator in Folge 6 der Fernsehreihe "Bit und Byte-Wir bauen einen Computer" gibt das Amateurfunkrufzeichen "DDOEU" aus.

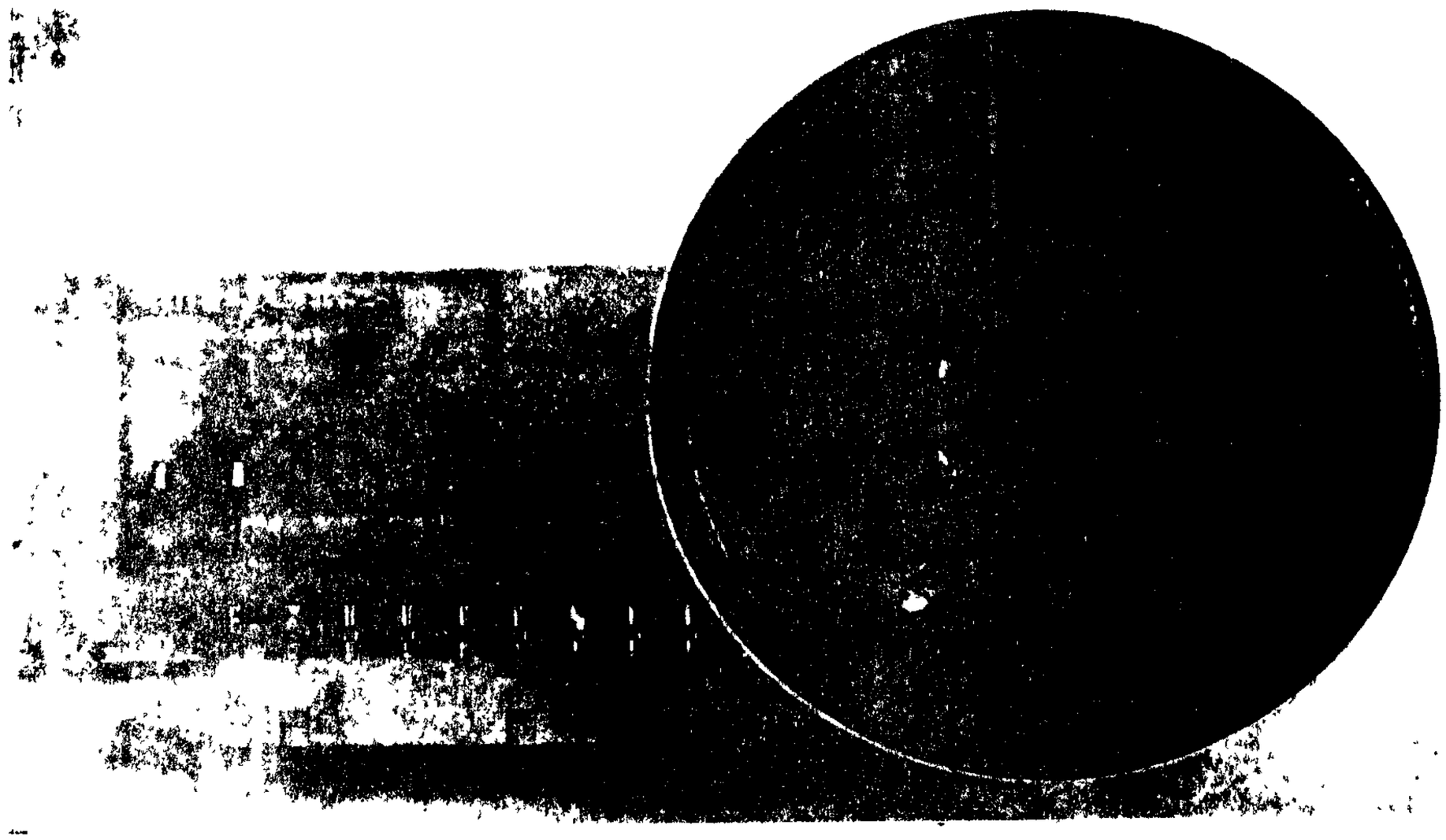


Abb 4/ ... chengenerator

4.6.6. SIEBENSEGMENT-ANZEIGE

Abb. 48 zeigt die aufgebaute Platine für eine Siebensegmentanzeige (vergl. Bezugsquellen). In dem IC sind 8 invertierende Treiber enthalten, deren Eingänge mit den Ausgängen A0-A7 des Computers verbunden sind. 7 Treiber (A1 - A7) sind Verstärker für die 7 Leuchtdiodensegmente, denen jeweils ein Widerstand vorgeschaltet ist. Der 8. Treiber (A0) schaltet den Leistungstransistor, dessen Kollektor die gemeinsame Kathode der Siebensegmentanzeige auf "0" setzt. Durch diese Beschaltung ist ein flackerfreier Bildaufbau für die Darstellung alphanumerischer Zeichen gewährleistet.

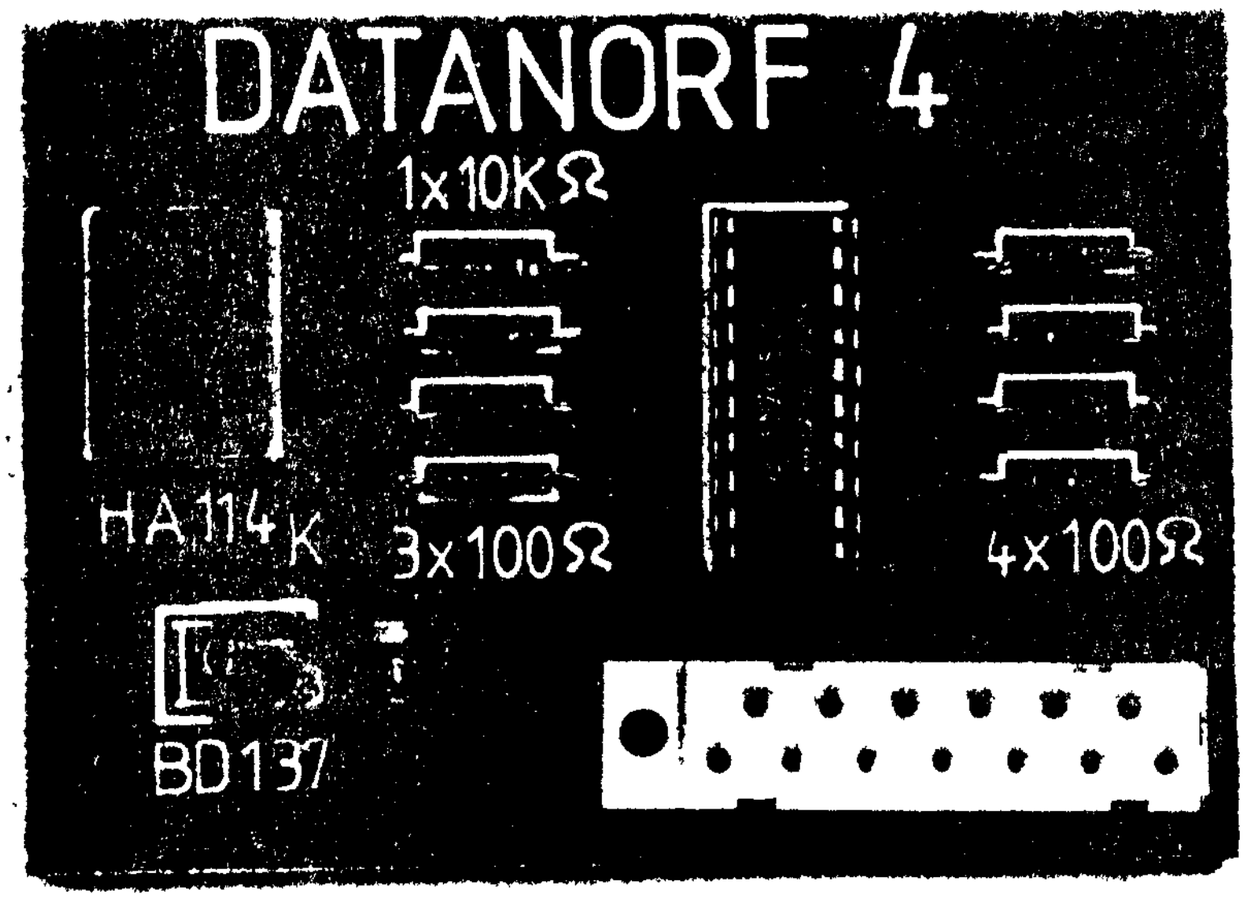


Abb. 48: Platine für die Siebensegmentanzeige

4.6.7. BOHRER

Das Fischertechnik-Bohrrobotermodell hat einen Motor für die Vorlage der Werkstücke sowie einen zweiten für das Heranführen des Bohrers an das Werkstück. Der Bohrvorgang selbst wird simuliert. Für die Erkennung der Endstellungen des Bohrvorschubs werden zwei ~~Schalter~~ abgefragt. Für die Programmierung der Steuerung des Bohrvorschubs reicht die Anwendung eines RS-Flipflops, das über die Schalter S1 und S2 getaktet wird: Der Bohrvorschubmotor wird nach dem Anhalten des Vorlagemotors eingeschaltet und führt die Vorlage solange nach unten, bis der Schalter S1, der ihn abschaltet, erreicht ist. Nach dem Bohren wird der Bohrvorschubmotor so wieder eingeschaltet, daß er den Vorschub nach oben führt. Ist der obere Schalter S2 erreicht, wird der Bohrvorschub gestoppt und das Drehen des Vorlagemotors für die zu bohrenden Werkstücke veranlaßt. Ein dritter Schalter, der durch die Drehung des Vorlagetisches betätigt wird, beendet die Vorlage des zu bohrenden Werkstücks und teilt dem Computer so die Lage des Werkstückes mit. Nach dem Bohren muß der Vorlagemotor bei geschlossenem Schalter S3 gestartet werden. Der Vorlagemotor muß sich weiterdrehen, obwohl der Schalter S3 jetzt geöffnet ist. Erst wenn S3 wieder geschlossen wird, muß der Vorlagemotor wieder angehalten werden.

```
***Bohrroboter***
**Festlegung der Zuordnung im Programm**
* Ausgang A0 : Werkstückvorlage drehen
* Ausgang A1 : Bohrkopf runterfahren
* Ausgang A2 : Bohrkopf hochfahren
* (Achtung: A1 und A2 dürfen nicht gleichzeitig "1" sein !!!)
*
* Eingang E1 : Schalter Vorlage
* Eingang E2 : Schalter unten
* Eingang E3 : Schalter oben
* E/A7 = 0    : Prozedur drehen
* E/A7 = 1    : Prozedur bohren
* In Prozedur bohren : E/A6 = 0 : Prozedur runter
                       E/A6 = 1 : Prozedur hoch
* In Prozedur drehen : E/A6 = 0 : drehen 0
                       E/A6 = 1 : drehen 1
00 INIT                    13 LD   1              24 LD   7
* Prozedur drehen           14 STOC 0              25 AND  6
03 LDC  7                  15 STOC 6              26 OEN  0
04 ANDC 6                  16 STO  7              27 STO  2
05 OEN  0                  *Prozedur runter       28 LD   3
06 STO  0                  17 LD   7              29 STOC 2
07 LD   1                  18 ANDC 6              30 STOC 7
08 STOC 6                  19 OEN  0              31 STOC 6
*Prozedur drehen 1         20 STO  1              32 JMP  x
09 LDC  7                  21 LD   2              * Ende
10 AND  6                  22 STOC 1
11 OEN  0                  23 STO  6
12 STO  0                  *Prozedur hoch
```

4.6.8. PLOTTER

Ein Plotter ist ein Zeichengerät. Wir wollen nun die Schrift "1-bit" auf ein Blatt Papier malen. Der Fischertechnik-Plotter hat zwei Motoren und eine Vorrichtung zum Zeichnen. Beide müssen angesteuert werden, damit der Plotter zeichnen kann. Für die Abfrage der Koordinaten der Stellung des Schlittens für den Zeichenstift werden Zählimpulse über das Interface an den Programmzähler gegeben. Für die digitale Abfrage werden eine Seiltrommel auf die Schneckenwell des Motors für das Drehen des Schreibarms und Winkelstücke auf die Aluminiumschiene zur Führung des Schlittens für den Plotterstift angebracht, die von Schaltern abgefragt werden. (Vergl. Abb. 49). Für das Plotten der Schrift "1-bit" ist ein umfangreiches Programm von 159 Programmschritten nötig. Daher ist die Erklärung des Programms gleich beigefügt.

```
***Plotter***
**Festlegung der Zuordnungen im Programm**
*Start: Links unten, A0: Stift, A1: Links fahren, A2: Rechts
fahren, *A3: Tief fahren, A4: Hoch fahren, A5: 1 = Takt vom
Prozessor, 2 = Takt vom Modell

00 INIT
* "1" Schreiben 1
* Stift runter
* hoch an
03 STO  5
04 STO  0
05 STO  4
06 STOC 5
07 NOP  x /7 mal
* "1" schreiben 2
* hoch aus
* links an
14 STO  5
15 STOC 4
16 STO  1
17 STOC 5
* hoch aus
* Stift hoch
* "i" anfahren 1
18 STO  5
19 STOC 4
20 STOC 0
21 STO  3
22 STOC 5
23 NOP  x /3 mal
* "b" anfahren
* Stift hoch
* rechts an
26 STO  5
27 STO  2
28 STOC 0
29 STOC 5
30 NOP  x /6 mal
* "b" schreiben 1
* Stift runter
* weiter rechts
* "b" schreiben, 4
* links aus
* hoch an
56 STO  5
57 STOC 1
58 STO  4
59 STOC 5
60 NOP  x /7 mal
* "i" anfahren 1
* Stift hoch
* hoch aus
* rechts an
67 STO  5
68 STOC 0
69 STOC 4
70 STO  2
71 STOC 5
72 NOP  x /6 mal
* "i" anfahren 2
* rechts aus
* tief an
78 STO  5
79 STOC 2
80 STO  3
81 STOC 5
* "i" Punkt
* Stift runter
* weiter tief
82 STO  5
83 STO  0
84 STOC 5
* "i" Rest 1
* Stift hoch
* weiter tief
85 STO  5
86 STO  0
* rechts aus
*Stift runter
* hoch an
102 STO  5
103 STOC 2
104 STO  0
105 STO  4
106 STOC 5
107 NOP  x /7 mal
* "t" schreiben 2
* Stift hoch
* hoch aus
* rechts an
114 STO  5
115 STOC 0
116 STOC 4
117 STO  2
118 STOC 5
* "t" schreiben 3
* rechts aus
* tief an
119 STO  5
120 STOC 2
121 STO  3
122 STOC 5
123 STOC 2
124 NOP  x
* "t" schreiben 3
* tief aus
* Stift runter
* links an
125 STO  5
126 STOC 3
127 STO  0
128 STO  1
129 STOC 5
```

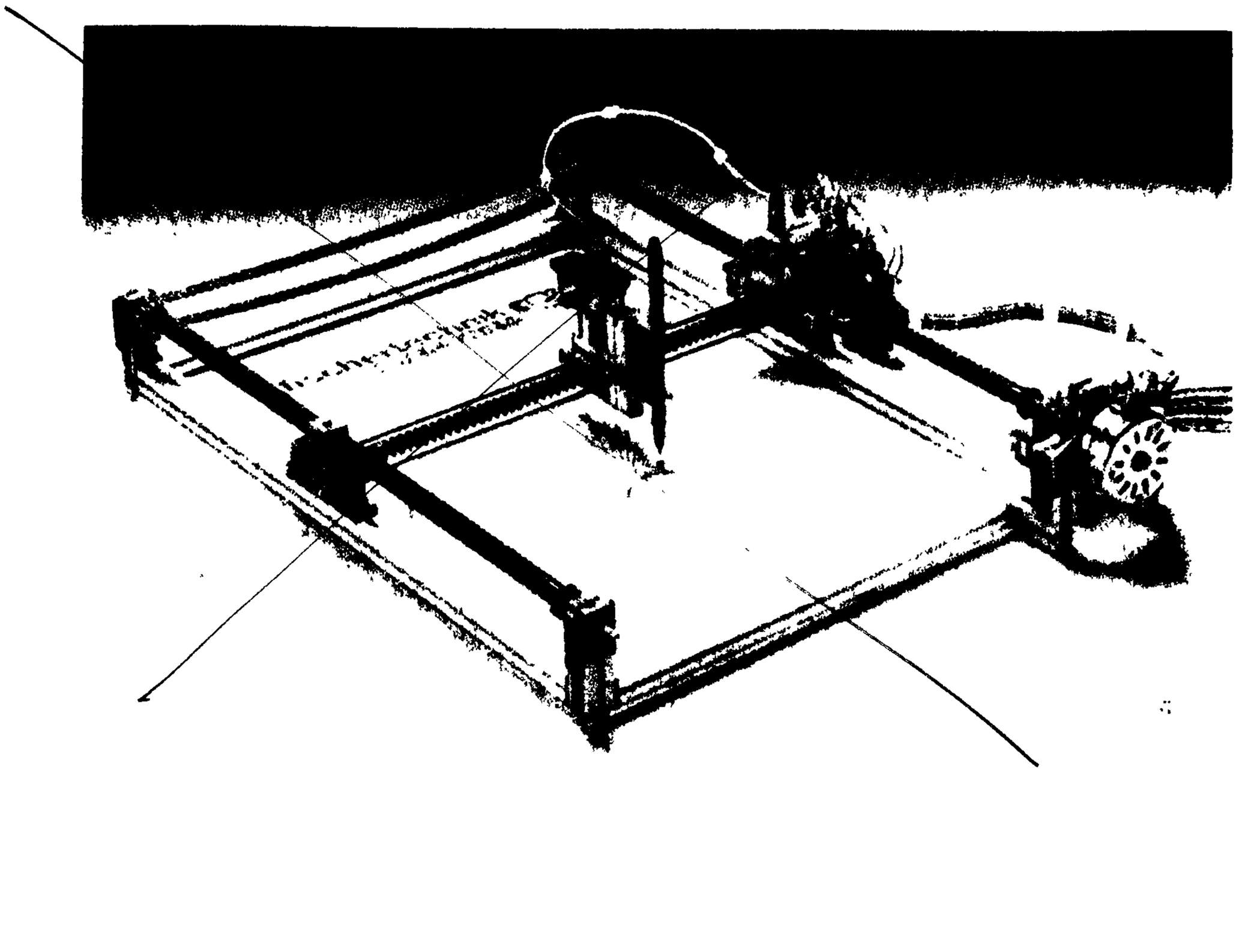
```
36 STO  5
37 STO  0
38 STOC 5
39 NOP  x /3 mal
* "b" schreiben 2
* rechts aus
* tief an
42 STO  5
43 STOC 2
44 STO  3
45 STOC 5
46 NOP  x /3 mal
* "b" schreiben 3
* tief aus
* links an
49 STO  5
50 STOC 3
51 STO  1
52 STOC 5
53 NOP  x /3 mal
```

```
87 STOC 5
88 NOP  x
* "i" Rest 2
* Stift runter
* weiter tief
89 STO  5
90 STO  0
91 STOC 5
92 NOP  x /3 mal
* "t" anfahren
* Stift hoch
* tief aus
* rechts an
95 STO  5
96 STOC 0
97 STOC 3
98 STO  2
99 STOC 5
100 NOP  x /2 mal
* "t" schreiben 1
```

```
130 NOP  x /2 mal
* Anfang anfahren 2
* Stift hoch
* links aus
* tief an
132 STO  5
133 STOC 0
134 STOC 1
135 STO  3
136 STOC 5
137 NOP  x /4 mal
* Anfang anfahren 2
* tief aus
* links an
141 STO  5
142 STOC 3
143 STO  1
144 STOC 5
145 NOP  0 /12 mal
* links aus
157 STO  5
158 STOC 1
159 JMP  x
```

Die im WDR-1-Bit-Computer liegenden Möglichkeiten haben wir bei weitem noch nicht erschöpft. Der interessierte Leser und Computerbauer findet sicher noch viele weitere Beispiele.

Abb. 49: Plotter



Anhang

Übersicht über die Bedienungselemente und LED-Anzeige 120

Internationaler Farbcode für Widerstände 121

Hexadezimal-Dual-Dezimal-Tabelle 122

Dauerhafte Speicherung von Programmen 123

Der Timer NE 555 als Rechteckgenerator 124

Datenblatt Timer NE 555 128

Minimalprojekt "Bit und Byte - Wir bauen einen Computer" 132

Beispiel: Durchführung einer Unterrichtseinheit
im WPF-Bereich (9./10. Schuljahr) in einer Gesamtschule 133

Die Simulation des WDR-1-Bit-Computers
mit dem Schulcomputer 144

Schüler bauen sich Computer 145

Zur Sendereihe "Bit und Byte" 146

Der Computerschein 148

Erweiterungen zum WDR-1-Bit-Computer 149

Literaturhinweise 191

Bezugsquellen 192

Übersicht über die Bedienungselemente und die LED-Anzeige