soll hier nicht erläutert werden (Siehe: Kapitel 4.2.). Für die Abarbeitung eines Programmschrittes wird immer nur eine Taktperiode benötigt.

1. Taktzustand "1"
Der Programmzähler steht auf einer bestimmten Adresse. Der Programmspeicher gibt den zu dieser Adresse gehörenden Programmschritt aus. Der Befehl gelangt zum Mikroprozessor und die Ein-/Ausgabeadresse zum Eingabebaustein und zum Ausgabebaustein.

2. Negative Taktflanke
Der Befehl wird in das Befehlsregister des Mikroprozessors eingelesen.

3. Taktzustand "0"
In der Kontrolleinheit des Mikroprozessors wird der Befehl decodiert.

Eingabebefehl: Die Schreib/Leseleitung enthält eine "0". Somit ist der Eingabebaustein frei und der Ausgabebaustein gesperrt. Im Eingangsbaustein wird eine der acht Eingangsleitungen mit der Datenleitung verbunden. Dabei bestimmt die Ein-/Ausgabeadresse die Nummer der Eingangsleitung. Die Zentrale Logikeinheit des Mikroprozessors erhält die Information der Datenleitung und damit die Information des betreffenden Eingangs. Außerdem erhält die Zentrale Logikeinheit von der Kontrolleinheit die Anweisung, diese Information entweder direkt zu ihrem Ausgang weiterzuleiten oder sie erst zu invertieren und dann weiterzuleiten.

Logikbefehl: In gleicher Weise wie beim Eingabebefehl gelangt hier auch die Information einer Eingangsleitung über die Datenleitung zur Zentralen Logikeinheit. Außerdem gelangt der Inhalt des Ergebnisregisters über den zweiten Dateneingang in die Zentrale Logikeinheit. Je nach Logikbefehl werden die Werte der beiden Eingänge logisch miteinander verknüpft. Das Ergebnis dieser logischen Verknüpfung erscheint am Ausgang der Zentralen Logikeinheit.

Ausgabebefehl: Die Schreib-/Leseleitung enthält eine "1". Somit ist der Ausgabebaustein frei und der Eingabebaustein gesperrt. Der Mikroprozessor schaltet, je nach Ausgabebefehl, entweder den Inhalt des Ergebnisregisters oder dessen Komplement auf die Datenleitung. Die Information der Datenleitung gelangt zum Ausgabebaustein. Dort wird sie auf einen der acht Ausgänge gesetzt. Dabei bestimmt wieder die Eingabe-/Ausgabeadresse die Nummer der Ausgangsleitung. Die Information an diesem Ausgang bleibt solange erhalten, bis sie gelöscht wird oder durch eine andere Information überschrieben wird.

Steuerbefehl: Die Schreib-/Leseleitung enthält eine "0". In gleicher Weise wie beim Eingabebefehl und beim Logikbefehl gelangt auch hier die Information einer Eingangsleitung über die Datenleitung zur Zentralen Logikeinheit. Diese wird jedoch nicht bearbeitet. Je nach Steuerbefehl gelangt an einen der Steuerausgänge des Mikroprozessors eine "1". Da nur der zum Steuerbefehl JMP zugehörige Steuerausgang beschaltet ist, wird

dieser auch hier nur näher behandelt. Dieser Steuerausgang ist mit dem Programmzähler verbunden. Eine "1" auf dieser Leitung setzt die Programmadres.e auf 0.

4. Positive Taktflanke
Der Programmzähler schreitet um eine Zählung fort. Bei einem Eingabebefehl und bei einem Logikbefehl wird die Information vom Ausgang der Zentralen Logikeinheit in das Ergebnisregister gespeichert.
Bei einem Ausgabebefehl geht die Schreib-/Leseleitung wieder auf "0".

5. Taktzustand "1"
Der Eingabebaustein ist gesperrt, da der Takt "1" ist. Der Ausgabebaustein ist gesperrt, da die Schreibleseleitung "0" ist. (Der Programmzähler leitet die neue Programmadresse zum Programmspeicher weiter. Dieser gibt den zu dieser Adresse gehörenden Programmschritt aus.)

6. Negative Taktflanke
War der letzte Befehl ein Steuerbefehl, so wird jetzt das zugehörige Steuersignal zurückgenommen, d.h., die "1" an dem Steuerausgang wird wieder zu "0".
(Der neue Befehl wird in das Befehlsregister des Mikroprozessors eingespeichert.)

3. DER ZUSAMMENBAU

3.1. BAUSATZ UND WERKZEUG (aus (14))

Grundsätzlich ist der WDR-1-Bit-Computer so konzipiert, daß alle Bauteile bis auf die fertig geätzten Platinen im Elektronikfachhandel einzeln erhältlich sind. Da diese Beschaffungsart aber sehr umständlich ist, halten wir es für sinnvoll, einen kompletten Bausatz einschließlich der Platinen bei der Firma raffel-elektronic (vgl. Anhang) zu beziehen. Ersatzteile können jederzeit nachgekauft werden.
Damit die Kosten für den Bausatz niedrig gehalten werden, enthält er entsprechend den vier Funktionseinheiten des Computers vier Einzeltüten mit den jeweiligen unsortierten Bauelementen. Wir werden auf ihren Inhalt näher eingehen, wenn wir die einzelnen Funktionsgruppen aufbauen, die Anzeige-, die Speicher-, die Prozessoreinheit und die Grundplatine. Der Bausatz enthält nicht die Tastatur. Dennoch werden wir ihren Aufbau und ihre Funktion beschreiben, da sie das Programmieren erheblich vereinfacht. Die Tastatur ist ebenfalls als Bausatz erhältlich, jedoch bei der Firma DATANorf (vgl. Anhang).
Nur 5 Dinge braucht der Computerbauer, zumindest in unserem Falle:

-eine Bohrmaschine mit Bohrständer und Bohrern
-einen Lötkolben mit Lötzinn
-einen Seitenschneider
-eine Säge
-ein Meßgerät

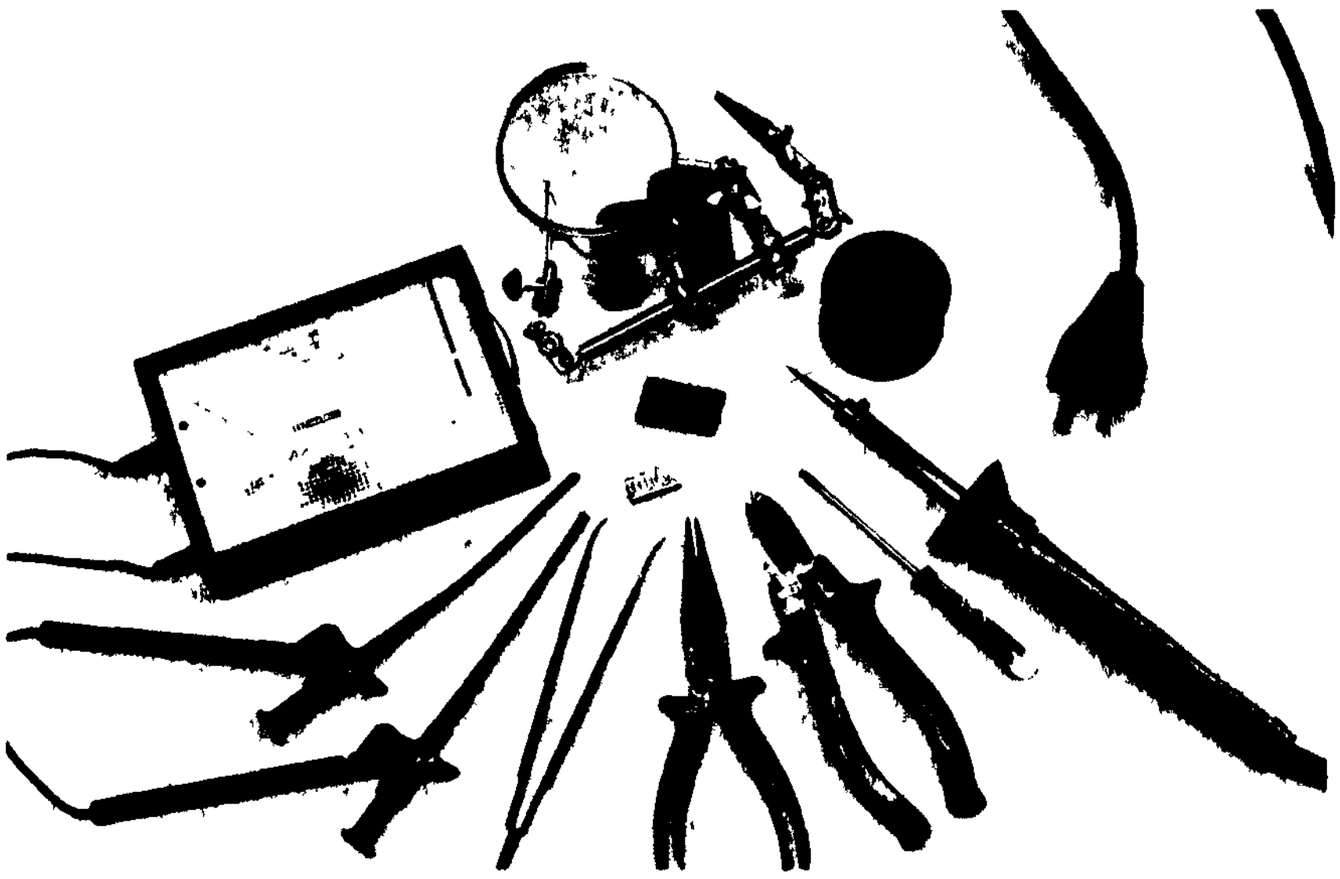


Abb. 26: Dinge, die der Computerbauer braucht

Die Bohrmaschine
Im Prinzip eignet sich jede Bohrmaschine, sofern das Bohrfutter die Bohrer von 1 mm bis 5 mm Durchmesser fassen kann. Bewährt haben sich Kleinbohrmaschinen für Niedervoltbetrieb, weil sie handlicher und sehr viel leiser sind.
Für den Bau des vorliegenden Computers werden die Metallbohrerstärken 1mm, 1,5 mm, 2 mm und 5 mm benötigt.

Der Lötkolben
Bei der Wahl des Lötkolbens ist man eingeschränkter. Hier eignen sich nur Elektroniklötkolben mit feiner Lötspitze und einer Leistung um 20 Watt. Werden andere Lötkolben verwendet, besteht die Gefahr, daß die Platinen zerstört werden. Bei zu groben Lötspitzen entstehen sehr leicht Lötfehler. Bei zu geringer Leistung, unter 15 Watt, fließt das Lötzinn nicht mehr richtig, bei zu hoher Leistung, über 25 Watt, verdampft das im Lötdraht enthaltene Flußmittel (Colophonium) sehr schnell. Dann besteht die Gefahr, daß die Lötstelle schlecht wird. Als Lötdraht eignet sich feiner Elektroniklötdraht von ca. 1 mm Durchmesser.

Lötübungen: auf das Gefühl kommt es an
Für diejenigen, die noch nie mit einem Lötkolben gearbeitet haben, empfehlen wir, aus Teilen nicht mehr verwendeter Elektronikgeräte, wie Radio, Fernseher, Computer o.ä., einzelne Bauteile herauszulöten. Am besten eignen sich eingelötete Bauelemente mit nur zwei Anschlüssen. Dazu wird auf der einen Seite des Bauteileträgers, der Platine, mit einer Pinzette ein Anschluß angefaßt, auf der anderen Seite wird die Lötstelle so lange mit der Lötkolbenspitze erhitzt, bis sich der Anschluß "butterweich" herausziehen läßt.Auf diese Weise erhält man sehr schnell das richtige Gefühl für eine genügend heiße Lötstelle.
Um das fehlerfreie Anlöten zweier Verbindunsstellen zu üben, klemmen wir einen Anschluß eines ausgelöteten Bauteils in einen kleinen Schraubstock und versuchen dann, ein anderes Bauteil anzulöten (siehe Anhang). Wenn dies mehrfach gelungen ist, können wir mit dem Bau unseres Computers beginnen.

Der Seitenschneider
Einen Elektronik-Seitenschneider benutzen wir, um überstehende Anschlüsse der Bauteile fachgerecht abzuschneiden. Nimmt man eine gewöhnliche Kneifzange, riskiert man, die Leiterbahn durch die Hebelwirkung beim Kneifen abzureißen.

Die Säge
Um einzelne Steckerleisten zu kürzen, reicht eine feine Handsäge.

Das Meßgerät
Zur Kontrolle des gelungenen Computeraufbaus brauchen wir ein einfaches Vielfachmeßgerät. Die unterste Preisklasse von ca. 25 DM für ein solches Meßgerät ist vollkommen ausreichend. Es sollte allerdings die Meßbereiche für Volt, Ampere und Ohm umfassen.
Der Gebrauch eines Oszilloskopes ist zwar sehr anschaulich, jedoch in unserem Fall eigentlich überflüssig. Mit diesen 5 Geräten kommen wir aus. Erleichtern können wir uns die Arbeit allerdings, wenn wir zusätzlich noch einige andere Geräte benutzen :

- Ein Biegewerkzeug, um die Anschlußdrähte einzelner Bauteile so zu biegen, daß sie problemlos in die entsprechenden Löcher der Platine passen.
- Eine Pinzette mit stumpfer, geriffelter Spitze, um kleine Anschlüsse zurechtzubiegen.
- Einen Kleinschraubstock, um die abzusägenden Bauteile festzuklemmen.

3.2. DIE ANZEIGEPLATINE

Der Bausatz enthält unter anderem eine Tüte mit den Teilen für die Anzeigeeinheit. Sie sind in der Abb. 27 dargestellt und aufgelistet.

Abb. 27: Bauteile für die Anzeigeeinheit: Platine, Widerstände, ICs, Fassungen für die ICs, Leuchtdioden, Stecker-leiste.

Der erste Arbeitsschritt besteht darin, diese Platine zu bohren. Dazu legen wir sie mit der Lötseite nach oben (vgl. Abb. 28) unter den Bohrer im Bohrständer.

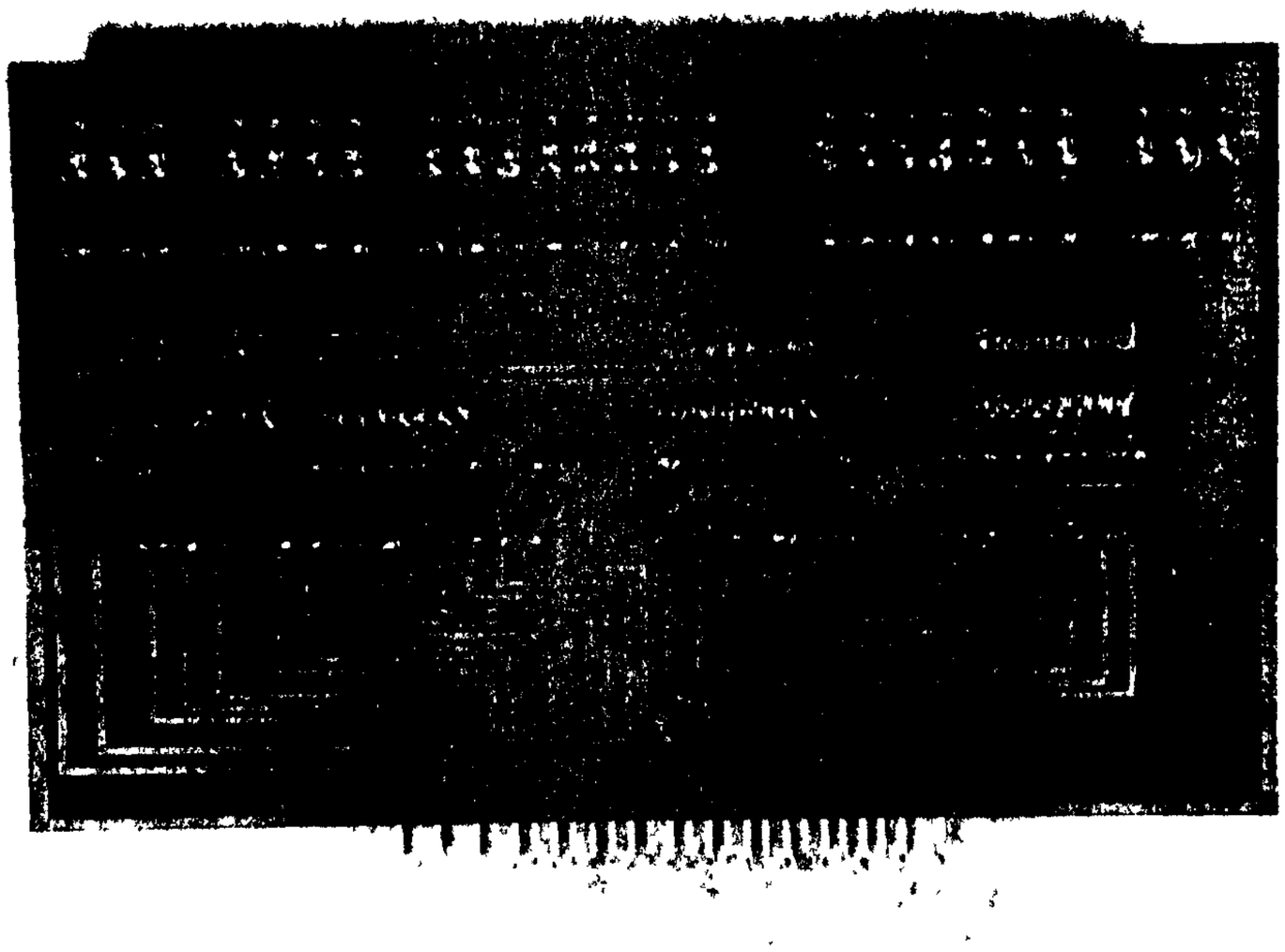


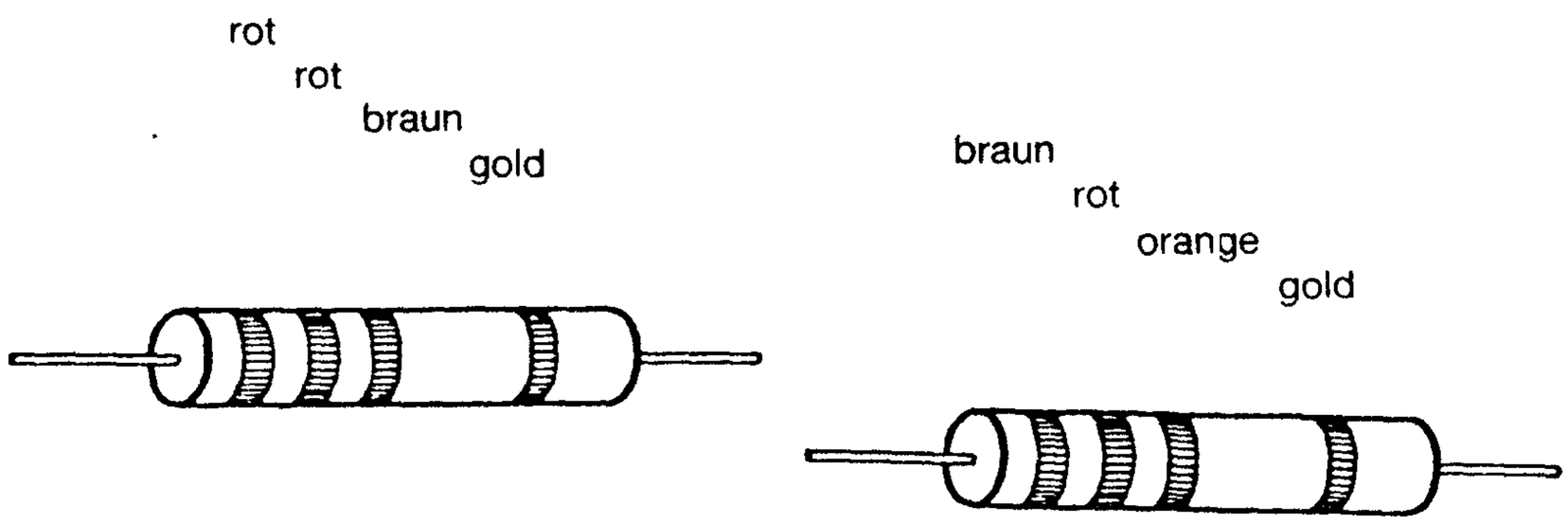
Abb. 28: Lötseite der Anzeigeplatine

Auf der Lötseite erkennen wir deutlich die Lötaugen mit den Bohrpunkten. Beim Bohren mit dem 1 mm Bohrer müssen wir diese Bohrpunkte genautreffen, dann hat der Bohrer automatisch die richtige Führung durch das Loch im Lötauge. Er rutscht dann nämlich nicht mehr unkontrolliert auf dem Lötauge herum, sondern bleibt in dem kleinen kupferfreien Punkt im Zentrum des Lötauges.

Die Bauteile werden eingesetzt

Wenn sämtliche Lötaugen durchbohrt sind, setzen wir die Bauteile ein. Der Aufdruck der Bestückungsseite gibt uns an, welche Bausteine wohin gehören.

Fangen wir mit den Widerständen an. Es gibt hier zwei verschiedene Werte, zum einen 220 Ohm, zum anderen 12 Kiloohm, die jeweils bestimmt codiert sind (vgl. Abb.29). Die Bedeutung der einzelenen Ringe in Abhängigkeit ihrer Farbe und Position ist im Anhang genau aufgeführt.



220 Ohm Widerstand

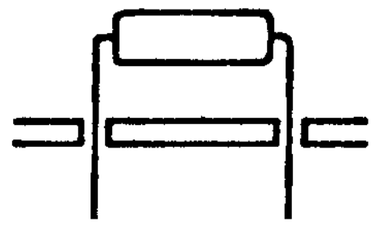
12 Kiloohm Widerstand

Abb. 29: Widerstände mit ihrer Codierung

Biegen wir zunächst die Anschlüsse der 220-Ohm-Widerstände, die den Farbcode "rot-rot-braun-gold" tragen, auf die richtige Länge, so daß sie mühelos in die Bohrlöcher passen. Dies geht besonders schnell mit dem Biegewerkzeug.

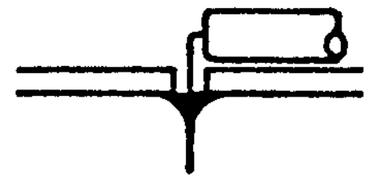
Wir empfehlen, die Widerstände in gleicher Ausrichtung ihrer Farben in die Platine einzusetzen. So erleichtern wir eine eventuelle Fehlerüberprüfung. Da wir die Platine zum Löten nun umdrehen müssen, winkeln wir vorher die herausstehenden Anschlußdrähte leicht ab. Dann fallen die Widerstände nicht heraus. Außerdem liegen die Bauteile eng an der Platine an. Dadurch wird verhindert, daß bei mechanischer Belastung die Bauteile durchgedrückt werden und so Leiterbahnen zerreißen.

Ist dies getan, löten wir einen Widerstand nach dem anderen an. Zuerst plazieren wir die heiße Lötspitze so, daß das entsprechende Lötauge und der Anschlußdraht im Lötbereich gleichzeitig erhitzt werden. Dann erst führen wir ein wenig Lötzinn zu. Wenn das Lötzinn auf dem Lötauge gut verlaufen ist, lassen wir die Lötstelle abkühlen. Ist eine Lötstelle nicht heiß genug geworden, eine sogenannte "kalte" Lötstelle, kann der elektrische Strom nicht gut fließen. Sieht sie so aus, wie ganz rechts in Abb. 30, dann müssen wir noch einmal nachlöten. Die Abb. 30 faßt die geschilderten Vorgänge noch einmal zusammen.

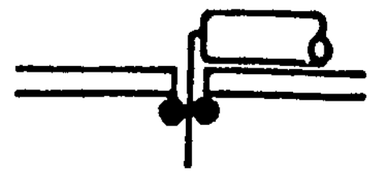


Falscher Aufbau
Widerstande mussen auf der P atine aufliegen, Bei Halbleitern sollte der Abstand zur Plat'ne mind. 5 mm betragen.

Richtiger Aufbau



So s eht eine gute Lotung aus, guter e ektrischer und mechanischer Kontakt.



Schlechte Lotung, kein oder nur mangelhafter elektrischer Kontakt.

Abb. 30: Einsetzen und Einlöten von Bauelementen

Sind alle 25 Widerstände richtig eingelötet, werden ihre überstehenden Enden mit dem Seitenschneider abgekniffen. Die abgekniffenen Drahtenden sollten wir aufbewahren. Wir benötigen sie noch an späterer Stelle (für anzufertigende Durchkontaktierungen).
Den gleichen Vorgang wiederholen wir mit den 12-Kiloohm-Widerständen. Diese 25 Widerstände haben den Farbcode "braun-rot-orange-gold".
Als nächstes setzen wir die Leuchtdioden (LEDs) ein. Dabei müssen wir auf die Polung (vgl. Abb.31) und die Farben der LEDs achten. Der kurze Anschluß ist die Kathode, die mit dem Minuspol verbunden werden muß. Die LEDs werden so in die Platine eingesetzt, daß die kurzen Anschlüsse an die 220-Ohm-Widerstände gebracht werden. Also stecken wir die langen Anschlüsse in die obere Lochreihe, die leitend miteinander verbunden ist. Die gelbe LED findet ihren polrichtigen Platz in den Löchern bei "Daten", sie ist also die dritte von links. Ganz links, bei "Takt", wird eine grüne LED eingesetzt. Die restlichen drei grünen LEDs sind die "Adressen"-LEDs, sie befinden sich ganz rechts. In den restlichen Löchern werden die roten LEDs plaziert.
Auch diesmal werden die Anschlüsse leicht abgewinkelt und anschließend wie folgt angelötet: Zuerst werden alle oberen Anschlüsse angelötet und dann erst die unteren. Wir erreichen dadurch, daß die Wärmeaufnahme der LEDs beim Einlöten in Grenzen gehalten wird, denn sie sind sehr hitzeempfindlich.

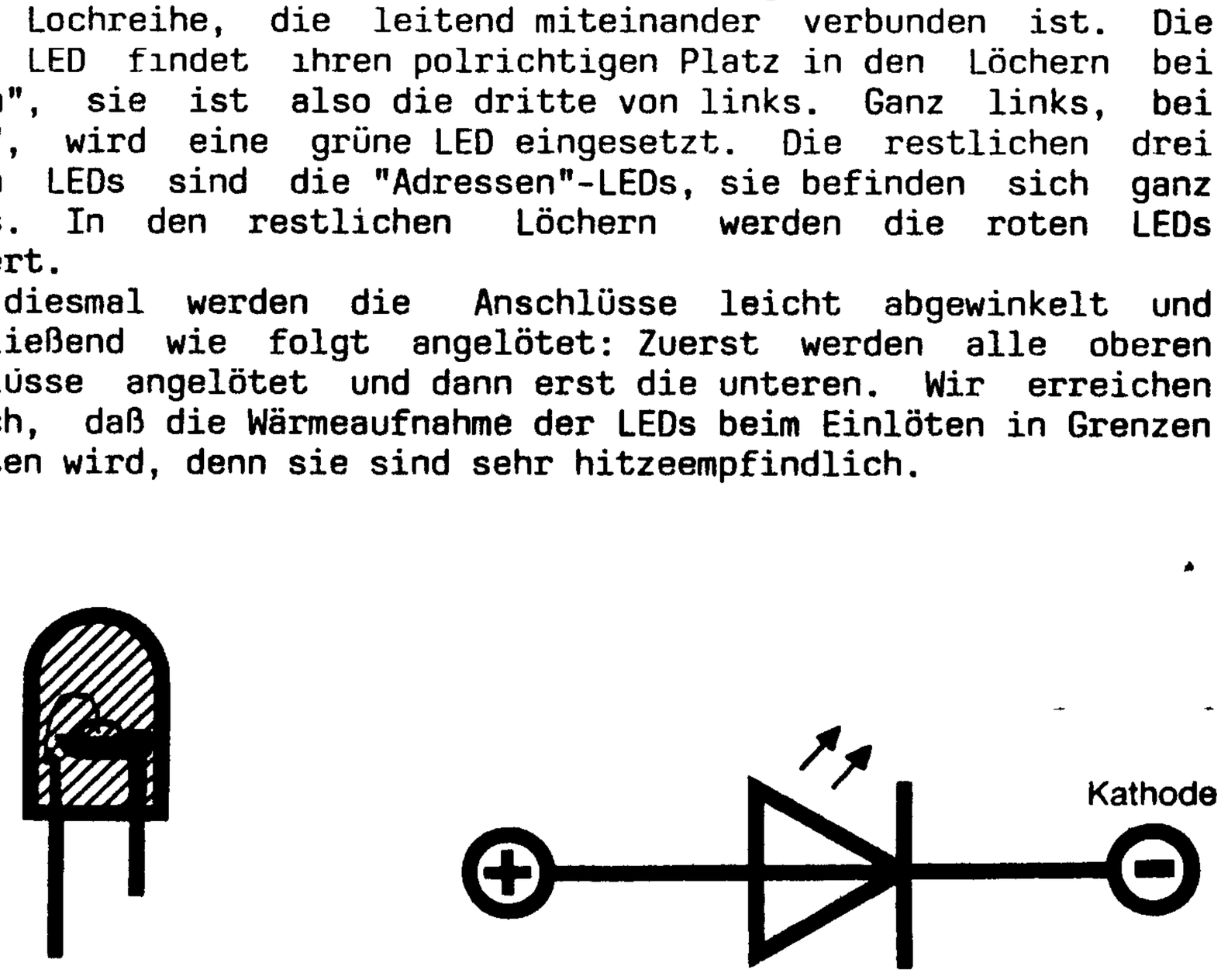


Abb. 31: Gehäuse, Anschlüsse und Polung einer LED

Nun sind weniger wärmeempfindliche Bauteile an der Reihe, die Fassungen für die Integrierten Schaltkreise (ICs) und die Steckerleiste. Beim Einsetzen der IC-Fassungen ist darauf zu achten, daß ihre Markierungszeichen (ein Halbkreis) mit denen des Aufdrucks auf der Platine übereinstimmen, sich also links befinden. Dadurch wird einer späteren Verpolung der ICs vorgebeugt. Vor dem Einlöten vergewissern wir uns, daß alle (!) Anschlüsse durchgesteckt sind. Andernfalls muß der gesamte Sockel wieder ausgelötet werden, und dies ist sehr aufwendig. Dann biegen wir wenigstens zwei diagonal weit auseinanderliegende Anschlüsse jedes Sockels so um, daß der Sockel eng an der Platine anliegt. So verhindern wir ein Durchdrücken der Anschlüsse beim Einsetzen der ICs und ein damit verbundenes Abreißen von Leiterbahnen.

Wenn wir die IC-Sockel-Anschlüsse einlöten, kann es trotz vorhandenem Lötstopplack passieren, daß zwei benachbarte Anschlüsse durch Lötzinn miteinander verbunden werden. Dann erwärmen wir die Lötstelle noch einmal und trennen diese ungewollte Verbindung gleichzeitig mit Hilfe einer angespitzten (!) Bleistiftmine. (Didaktisch-methodische Anmerkung: Ein Bleistift ist ein in der Schule vorhandener üblicher Gegenstand, auch wenn er in den seltensten Fällen angespitzt ist. Es ist nicht zu befürchten, daß die Graphitschicht, die beim Auftrennen der ungewollten Lötbrücke möglicherweise entsteht, beim WDR-1-Bit-Computer im Schulbetrieb zu Störungen führt.) Die ICs werden jetzt noch nicht eingesetzt. Übrig bleibt noch die Steckerleiste. Abb. 32 zeigt den Lötanschluß der Steckerleiste von der Seite.

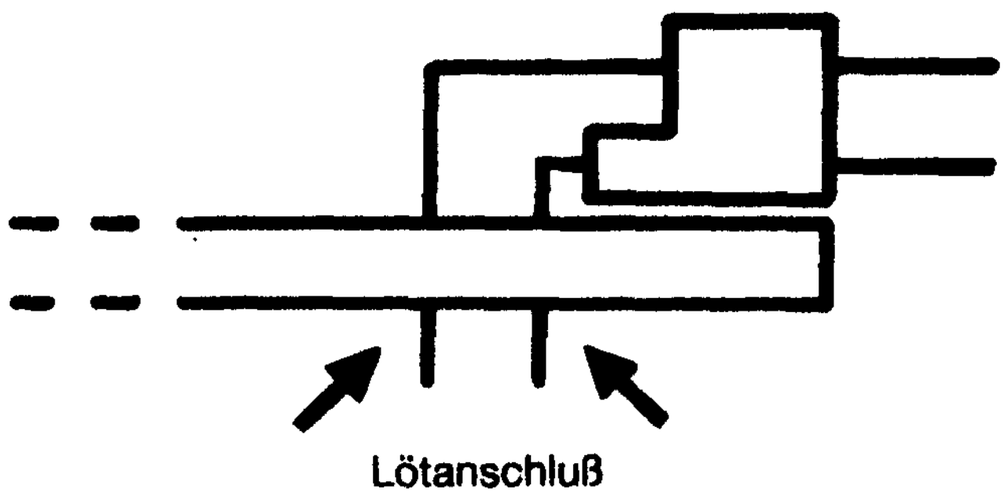


Abb. 32: Lötanschluß der Steckerleiste von der Seite

Nach dem Durchstecken der Lötanschlüsse kontrollieren wir, ob auch wirklich alle Anschlüsse auf der Lötseite zu sehen sind. Das Anlöten der Steckerleiste sollte uns nun leichtfallen. Abb. 33 zeigt die fertige Anzeigeplatine.

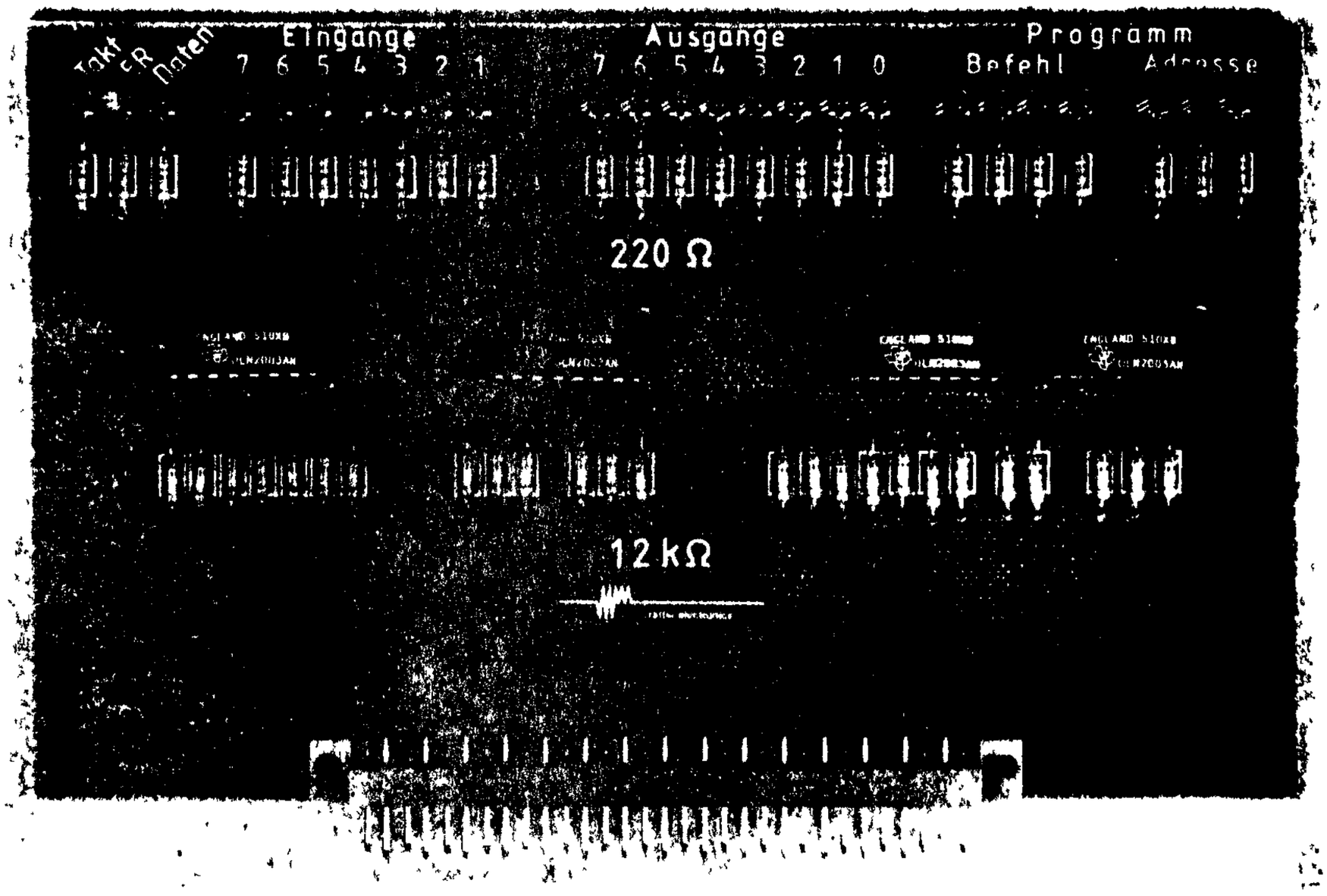


Abb. 33: Fertiggestellte Anzeigeplatine

Fehlerprüfung nach Fertigstellung

Haben wir fehlerfrei gearbeitet? Um dies zu überprüfen, testen wir die fertiggestellte Anzeigeplatine. Dazu benötigen wir eine Spannungsquelle mit 5 Volt Gleichspannung. Eine 4,5-Volt-Flachbatterie kann allerdings genauso verwendet werden. Beim Testen lernen wir gleichzeitig das Schaltbild lesen. Zuerst überprüfen wir die Leuchtdioden: Wir führen den Pluspol unserer Spannungsquelle mit Hilfe eines roten Kabels an einen oberen Anschluß einer beliebigen LED. Den Minuspol der Spannungsquelle dürfen wir auf keinen Fall an den anderen Anschluß der LED legen. Die hier verwendeten LEDs vertragen höchstens einen Strom von je 20 mA. Um sie nicht zu zerstören, müssen wir beim Testen den dazugehörenden Vorwiderstand von 220 Ohm als Strombegrenzer einbeziehen. Wir legen also den Minuspol an den Widerstand an.Um eine Polverwechslung möglichst auszuschließen, verwenden wir hierfür ein schwarzes Kabel. Diese LED muß nun leuchten. Indem wir den Minuspol an sämtlichen Widerständen vorbeiführen, überprüfen wir die Funktionsfähigkeit aller LEDs und Widerstände. Wenn eine LED nicht aufleuchtet, kann dies folgende Ursachen haben:

- Es liegt eine kalte Lötstelle vor. Wir löten nach.
- Die LED ist verpolt. Nach dem Umpolen der Testanordnung müßte sie leuchten. Ist dies der Fall, wird die Leuchtdiode umgelötet.
- Der Widerstand ist falsch. Wir kontrollieren zunächst die Farbringe. Stimmen diese, so messen wir den Widerstand mit dem Vielfachmeßgerät im Ohmbereich. Weicht dieser stark vom erwarteten Wert ab, so wird ein neuer Widerstand mit dem Wert 220 Ohm eingesetzt.
- Die LED ist defekt. Wir wechseln die LED aus.

Wenn alle LEDs ansprechen, suchen wir sie und ihre Vorwiderstände im Schaltbild und markieren sie farbig, z.B. gelb.

Nun können wir die noch fehlenden vier ICs, Treiber-ICs polrichtig einsetzen, d.h., ihre Markierungen müssen mit denen der Fassungen übereinstimmen. Sie haben die Aufgabe, den

geringen Steuerstrom für die LEDs aus den einzelnen Funktionseinheiten des Computers zu verstärken (die LEDs zu treiben). Daher nennt man solche ICs auch Treiber-ICs.
Wir überprüfen sie zusammen mit ihren Vorwiderständen, indem wir die Spannung diesmal an die Steckerleiste führen. Diese Vorwiderstände haben die Aufgabe, den für die Verstärker zu hohen Ansteuerstrom zu begrenzen. Pin 1, das ist der ganz rechte auf der Steckerleiste, ist der Pluspol. Wir markieren ihn am Stecker rot. Pin 2, also der Pin daneben ist der Minuspol. Wir markieren den Stecker an dieser Stelle schwarz. Mit Hilfe der beiden Kabel stellen wir eine Verbindung zwischen Pin 1 sowie Pin 2 und der Spannungsquelle her. Ein zweites rotes Kabel wird mit dem Pluspol verbunden und nacheinander an die restlichen Stifte der Steckerleiste geführt. Nun müssen alle LEDs der Reihe nach aufleuchten. Gibt es eine LED die nicht aufleuchtet? Folgende Fehler können vorliegen:

- Ein IC ist beschädigt. Wir tauschen es mit einem Nachbar-IC aus. Ist der Fehler an dieser Stelle behoben, so muß er nun an einer anderen Stelle auftauchen.
- Der Vorwiderstand ist defekt, hat einen falschen Wert, oder ist fehlerhaft eingelötet.
- Die dazugehörende Leiterbahn ist unterbrochen. Dies überprüfen wir mit dem Ohmmeter.

Leuchten beim Testen eines Treibers zwei oder mehrere LEDs gleichzeitig auf, so suchen wir unter dem IC nach einer Lötbrücke und entfernen sie.

Die Anzeigeplatine ist nun betriebsbereit. Anschließend suchen wir noch die vier ICs ULN 2003 und die dazugehörenden Vorwiderstände im Schaltbild und markieren sie mit der gelben Farbe für die Elemente der Anzeigeplatine.

3.3. DIE GRUNDPLATINE

Der erste Arbeitsschritt für die Fertigstellung der Grundplatine besteht wieder darin, die Löcher von der Lötseite her, dies ist die Seite ohne Bestückungsaufdruck, mit einem 1-mm-Bohrer zu bohren. Die fünf großen Bohrpunkte werden mit einem 5-mm-Bohrer bearbeitet. Für die mittelgroßen Bohrpunkte (neben den Bezeichnungen "HT", "LT" und "ST" sowie "1M") nehmen wir einen 1,5-mm-Bohrer. Die Grundplatine hat auf beiden Seiten Leiterbahnen, die zum Teil elektrisch verbunden werden müssen. Wir setzen hierfür an den mit weißen Kreisen markierten Stellen Durchkontaktierungen ein. Dazu benutzen wir z.B. die beim Aufbau der Anzeigeplatine abgekniffenen Drahtenden der Widerstände. Das Schema einer Durchkontaktierung zeigt Abb. 34.

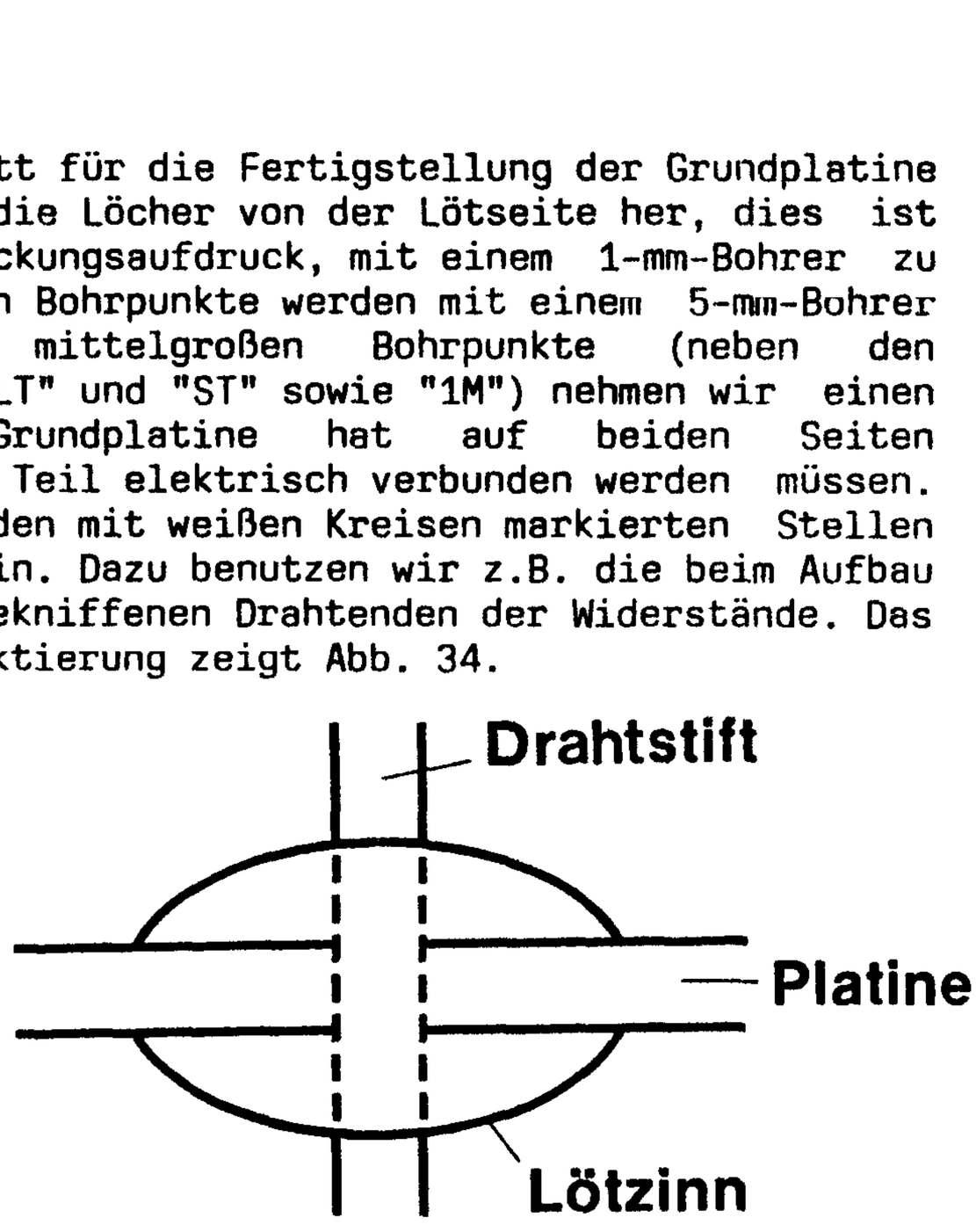


Abb. 34: Schema einer Durchkontaktierung

Damit die Durchkontaktierungen auf der Platinenunterseite genügend lang herausragt, legen wir unter die Platine einige Geldstücke. Wir löten so alle Drahtstifte auf der Bestückungsseite fest, biegen sie leicht um und wenden die Platine. Jetzt können wir die Durchkontaktierungen auch auf der Lötseite zu Ende führen, ohne daß die Stifte herausfallen oder herausgezogen werden. Überstehende Drahtenden der Durchkontaktierungen werden danach abgeschnitten. Einige der Buchsenleisten müssen vor dem Einlöten an ihren Enden gekürzt werden: eine 13-polige und die beiden 31-poligen Buchsenleisten sowie die 21-polige Steckerleiste. Eine Pinzette hilft, die vielen Anschlüsse einer Buchsenleiste in die vielen Löcher zu führen, manchmal stehen sie je nach vorheriger Lagerung etwas seitlich ab. Beim Einlöten der Steckerleiste sowie einiger anderer Bauteile müssen wir darauf achten, daß einige ~~Anschlüsse~~ zusätzlich auch auf der Bestückungsseite angelötet werden müssen. Dies erkennen wir an den kupfernen Lötaugen. Jetzt werden die 3,3-Kiloohm-Widerstände (Farbcodierung: "orange-orange-rot-gold") und der 100-Kiloohm-Widerstand (Farbcodierung: "braun-schwarz-gelb-rot") eingesetzt und verlötet. Es folgen die Kondensatoren. Sie müssen so eingesetzt werden, daß ihre aufgedruckten Polmarkierungen dem Bestückungsaufdruck der Platine entsprechen. Wenn wir die Sockel, Schalter und Taster einsetzen, achten wir wieder darauf, daß auch diese Bauteile wieder eng an der Platine anliegen. Jeder einzelne Taster muß so eingesetzt werden, daß der "Drehpunkt" des Tasters hinten liegt, der Taster sich also damit vorne absenken läßt. Auch den Spannungskonstanter 7805 und die Diode müssen wir polrichtig einlöten (Siehe Abb. 35).

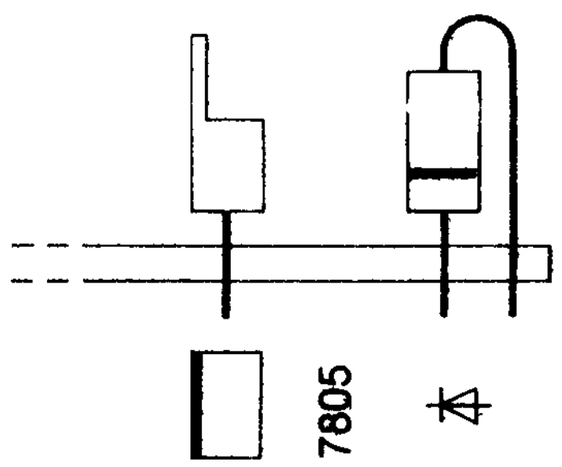


Abb. 35: Spannungskonstanter und Diode

Diese Abbildung zeigt auch, daß die Diode stehend eingesetzt werden sollte. Es fehlen noch das Einlöten des 1-Megaohm-Potentiometers und das Einsetzen der 5 Gummifüsse in die 5-mm-Löcher. Die Platine ist fertig aufgebaut, wenn kein kupfernes Lötauge auf beiden Seiten der Platine mehr zu sehen ist.
Abb. 36 zeigt die fertige Grundplatine.

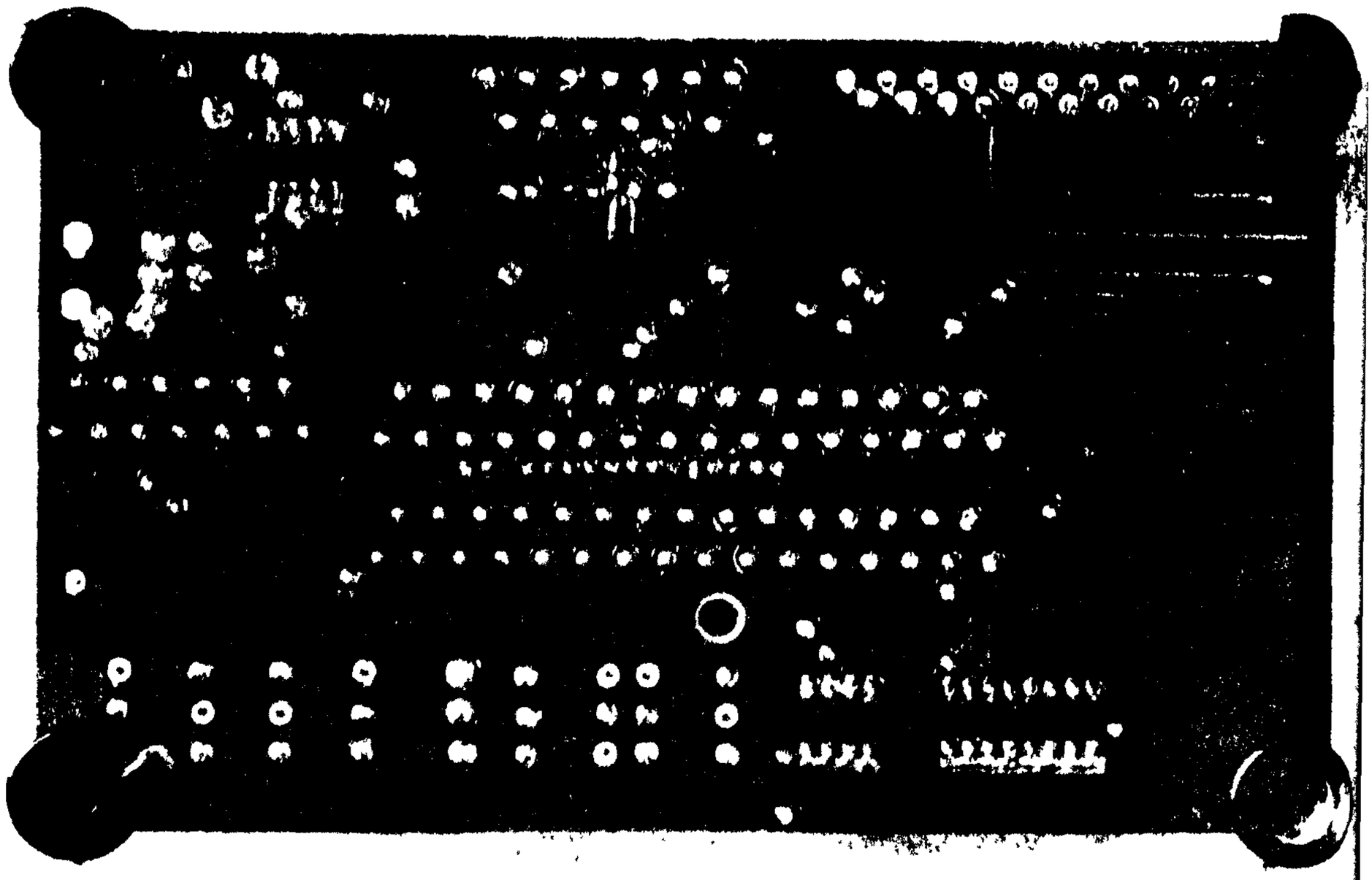


Abb. 36: Fertiggestellte Grundplatine von der Rückseite

Haben wir auch diesmal fehlerfrei gearbeitet? Um dies herauszufinden, legen wir an den Eingang des Spannungs konstanters (+ und - Aufdruck der Platine in der Nähe der Diode) eine Wechselspannung von etwa 9 Volt. An der Kathode der Diode liegen jetzt ca. 12 Volt Gleichspannung an (Siehe Kap. 2.2.9.). Dies überprufen wir mit unserem Vielfachmeßgerät. Messen wir nur eine Spannung von etwa 4 Volt, so haben wir vergessen, den 1000-Mikrofarad-Kondensator auf der Bestückungsseite anzulöten. Ist jedoch alles in Ordnung, so können wir statt der Wechselspannung auch eine Gleichspannung von etwa 9 Volt an den Eingang der Stromversorgung anlegen. In diesem Falle arbeitet die Diode als Verpolungsschutz. Jedes handelsübliche Steckernetzgerät für die Versorgung von Transistorgeräten mit einem max. Strom von mindestens 350 mA und einer Spannung von ca. 9 Volt eignet sich für unseren Computer.
Nun überprüfen wir die Ausgangsspannung der Stromversorgung an den Buchsenleisten: An Buchse 1 jeder Buchsenleiste liegt jeweils die 5-Volt-Spannung, an Buchse 2 aller 31poligen und der alleinstehenden 13-poligen Buchsenleisten liegt die Masse. Dabei ist zu beachten, daß die alleinstehende 13polige Buchsenleiste umgekehrt eingelötet worden ist.
Folgende Fehler können vorliegen, wenn wir nicht 5 Volt an diesen Meßpunkten nachweisen können:
- Der Spannungskonstanter arbeitet nicht, weil er beispielsweise defekt ist oder falsch eingelötet wurde.
- Der 1-µF-Kondensator ist verpolt worden.

Im Schaltbild suchen wir die Elemente der Stromversorgung (links unten) und markieren sie rosa. Nun können wir unbesorgt den Timer NE 555 (das Taktgeber-IC) in seine Fassung setzen und seine Wirkung im Verbund mit dem 1-µF-Kondensator, dem 22-Kiloohm-Widerstand und dem 1-Megaohm-Potentiometer testen. Dazu greifen wir am Ausgang Q des Taktgeber-ICs, dies ist sein

Anschluß 3 mit der Verbindung zu dem 100-Kiloohm-Widerstand, die Ausgangsspannung ab. Je nach Stellung des Potentiometers kann der Zeiger dem Takt folgen.

Der Taktgeber arbeitet nicht?
- Der Kondensator ist verpolt.
- Das IC ist falsch eingesetzt.
  Am Anschluß 4 oder am Anschluß 8 des IC´s liegt keine Betriebsspannung.

Die Elemente des Timers markieren wir im Schaltbild grün. Auf die fertige Grundplatine können wir nun die Anzeigeplatine stecken. Dabei zeigt sich, ob diese beiden Platinen schon zusammenspielen. Wir können die Eingangsleuchtdioden 1 bis 4 ein- bzw. ausschalten, indem wir die vier Eingangsschalter, die in dem 8poligen Sockel stecken, betätigen.

3.4. DIE SPEICHERPLATINE

Die zweiseitig geätzte Speicherplatine wird wieder von der Lötseite her mit einem 1-mm-Bohrer bearbeitet. Als erstes werden die Durchkontaktierungen gesetzt und verlötet. Die überstehenden Drahtenden kneifen wir wieder ab, außer dem Stift, der sich rechts unter dem Testpunkt "T" befindet. Jetzt setzen wir die 22-Kiloohm-Widerstände ("rot-rot-orange-gold") ein und verlöten sie. Es folgen die Sockel für die ICs und die Steckerleiste. Die ICs setzen wir erst ein, wenn wir die Platine auf Kurzschlüsse hin überprüft haben. Dazu stecken wir sie hinten auf die Grundplatine (vgl. Abb.37).

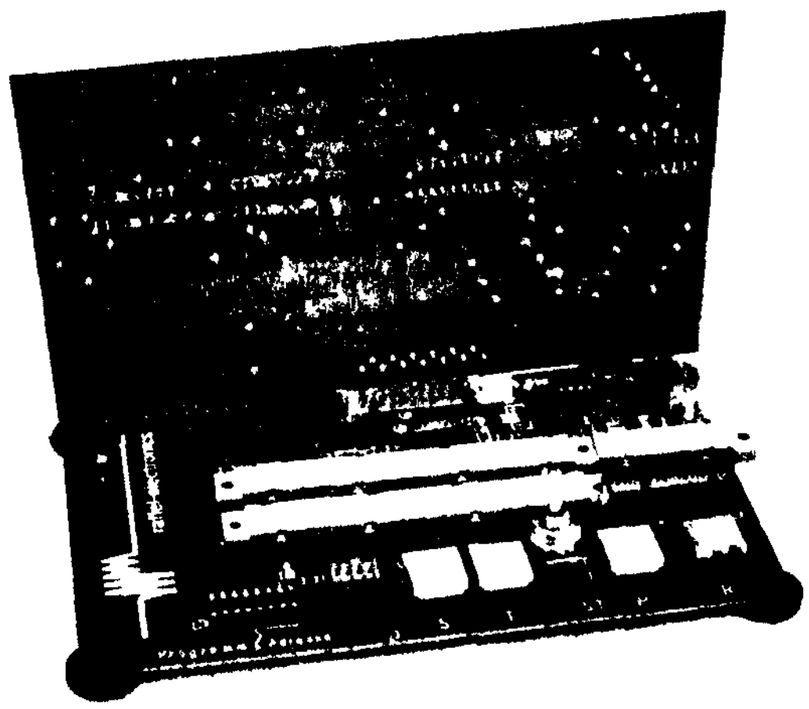


Abb. 37: Grundplatine mit aufgesteckter Speicherplatine

Vorne stecken wir die Anzeigeplatine dazu. Leuchten die vier Eingangs-LEDs wieder, so hat die Speicherplatine keinen Kurzschluß zwischen der 5 V-Leitung und der Masse. Liegt jedoch

ein Kurzschluß vor, so hilft das Vielfachmeßgerät als Ohmmeter bei der Kurzschlußsuche. Wir nehmen die Speicherplatine wieder heraus und setzen die beiden Speicher-ICs, die 2112, gemäß dem Bestückungsaufdruck in ihre Fassungen. Nach dem erneuten Einsetzen in die Grundplatine zeigen die 7 Programm-LEDs der Anzeigeplatine nun zufällige Werte an.
Um einen bestimmten Programminhalt anzuzeigen, legen wir die Information "0000 000" an die 7 Speichereingänge. Dies geschieht mit Hilfe von 7 Schaltern im 16poligen Sockel der Grundplatine. Der vierte Schalter von rechts hat keine Bedeutung in unserem System.
Im Schaltbild kennzeichnen wir die beiden Speicher-ICs und die 7 Schalter (vier Befehls- und drei Adreßschalter) mit einem roten Stift. Aus dem Schaltbild geht hervor, daß an den Eingängen der Speicher, das sind die Anschlüsse 9 bis 11 bzw. 9 bis 12, 0 Volt anliegen, wenn die Schalter geschlossen sind. Wir drücken nun auf den Schreibtaster "S" der Grundplatine. Dadurch wird die eingestellte Information "0000 000" in den Speicher übernommen. Die 7 Programm-LEDs auf der Anzeigeplatine müssen nun erloschen sein. Ist dies nicht der Fall, so sind folgende Fehler denkbar:
- Die Schalter wurden nicht geschlossen. Das kann mit dem Ohmmeter überprüft werden.
- Die Speicher ICs sind verpolt.
- Die Speicher-ICs sind defekt.
- Der Schreibtaster arbeitet nicht. Auch hier hilft das Ohmmeter bei der Fehlersuche.

Wir überprüfen, ob unsere beiden Speicher auch die Information "1111 111" behalten können. Dazu öffnen wir die 7 Schalter und betätigen wieder den Schreibtaster. Jetzt müssen alle 7 Programm-LEDs leuchten. Ist dies nicht der Fall, so sind folgende Fehler möglich:
- Ein Speicher-IC ist defekt. Ein Tausch der Speicher-ICs bestätigt die Vermutung.
- Die Zuleitung von den Speichern zu der Anzeigeplatine ist unterbrochen.
- Der Widerstand am entsprechenden Speichereingang ist nicht wirksam. Wir überprüfen die Lötstelle bzw. seinen Wert. Das Aufsuchen dieses 22-Kiloohm-Widerstandes im Schaltbild verdeutlicht, daß dieser den Speichereingang immer dann auf "1" setzt, wenn der dazugehörige Schalter geöffnet ist.

Unter den Speicher-ICs mit den dazugehörigen Widerständen erkennen wir im Schaltbild die beiden Zähler-ICs CD 4029. Die Anschlüsse 15 sind zusammengeschaltet. Dies sind die Takteingänge für die Zähler. Wir markieren im Schaltbild die beiden ICs ebenfalls rot. Diese beiden ICs setzen wir in die Fassungen der Speicherplatine ein und testen diese nun zu Ende. Wir verbinden den Ausgang des Timers (Anschluß 3) auf der Grundplatine mit dem Testpunkt "T" der Speicherplatine. Er ist der genannte Takteingang für die Zähler. Je nach Taktgeschwindigkeit des Taktgebers können wir das wechselnde Aufleuchten der Programmleuchtdioden verfolgen. Ist dies nicht der Fall, so sind folgende Fehler möglich:
- Das Taktgeberpotentiometer ist zu schnell eingestellt.
- Die Zähler-ICs sind verpolt.
  Die Zähler-ICs sind defekt.
- Die Stromversorgung an den Sockeln ist unvollständig. Hier kann man nachmessen.
- Der Speicherinhalt ist immer der gleiche. Diesen können wir inzwischen verändern.

Die Speicherplatine ist damit fertig gebaut.

3.5. DIE PROZESSORPLATINE

Abbildung 38 zeigt die Bestückungsseite der Prozessorplatine.

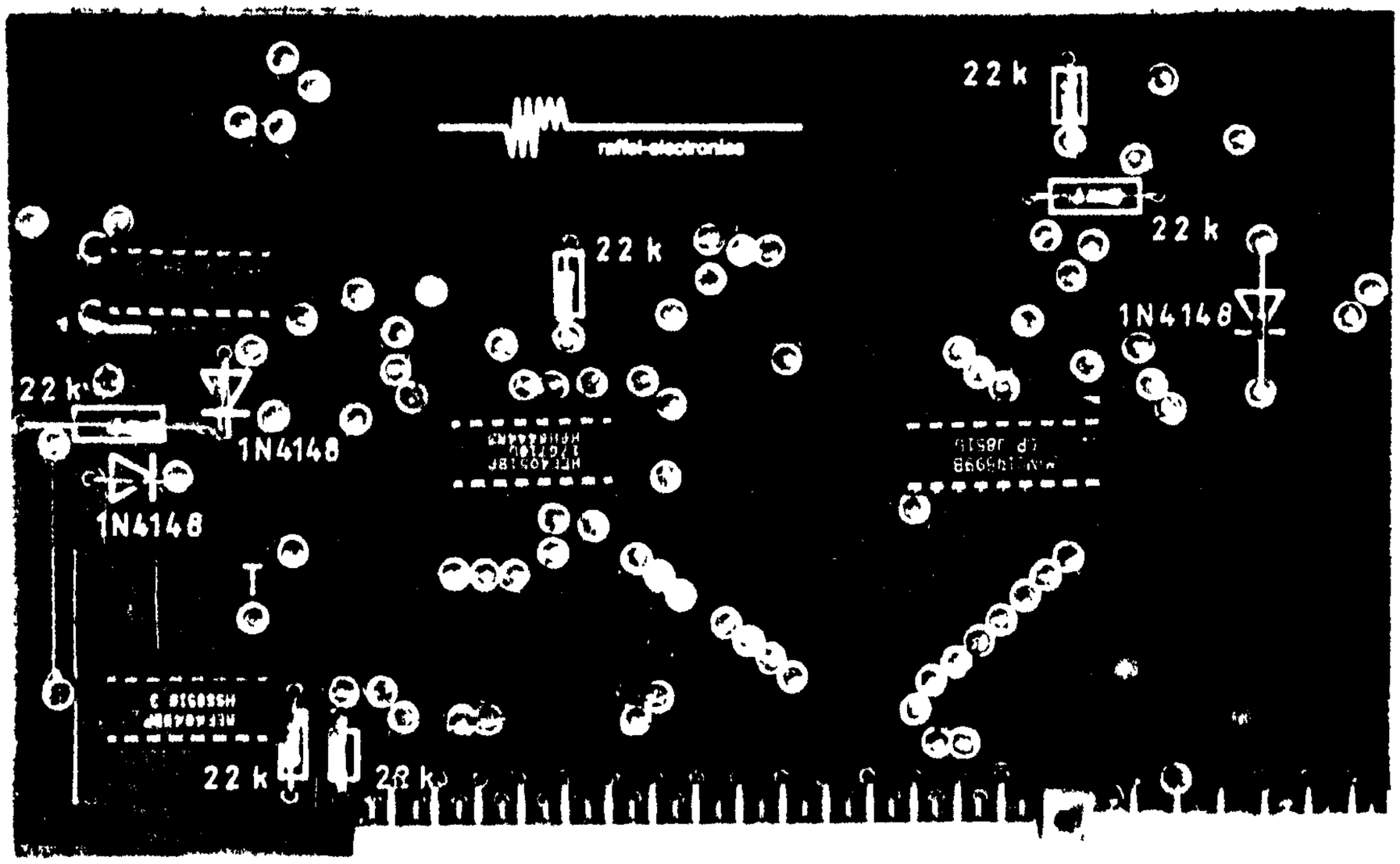


Abb. 38: Bestückungsseite der Prozessorplatine

Auch sie ist wieder von beiden Seiten geätzt. Daraus folgt, daß wir sie wieder durchkontaktieren müssen. Danach löten wir die 22-Kiloohm-Widerstände ein. Bei den drei Dioden müssen wir vor dem Einlöten wieder auf das polrichtige Einsetzen achten. Wir schlagen vor, die IC-Sockel in einer bestimmten Reihenfolge einzusetzen: zuerst den 18poligen und dann die 16poligen .So verhindern wir, daß z.B. ein 16poliger Sockel an den Platz des 18poligen versehentlich gerät.
Die Steckerleisten müssen vor dem Einsetzen auf das richtige Maß gekürzt werden. Die fertig aufgebaute Prozessorplatine stecken wir zunächst noch ohne ICs zusammen mit den anderen beiden Platinen in unsere Grundplatine. Verändert sich die Helligkeit der Eingangs-LEDs nicht, so liegt kein Kurzschluß vor. Die ICs setzen wir in einer ganz bestimmten Reihenfolge ein, und zwar das IC CD 4049 als erstes. Es enthält 6 Signalumkehrer, sogenannte Inverter, von denen allerdings nur zwei benötigt werden. Das Betätigen eines Tasters kann viele Impulse verursachen, wenn der Taster prellt. Wir benötigen für einen Taktschritt jedoch nur einen Einzelimpuls. Die mit Hilfe der beiden Inverter aufgebaute Schaltung entprellt den Takttaster "T" auf der Grundplatine, der beim Betätigen so nur einen definierten Impuls erzeugt. Im Schaltbild markieren wir die beiden Inverter (sie liegen neben der Stromversorgung) blau. Die Entprellschaltung funktioniert so: Die Tasterstellung im Schaltbild legt an den Eingang (Pin 3) des Inverters eine "1". Das invertierte Signal erzeugt am Ausgang (Pin 2) eine "0". Somit liegt über dem 22-Kiloohm-Widerstand am Eingang des anderen Inverters (Pin 14) eine "0". Diese erscheint invertiert

als "1" am Ausgang dieses Inverters (Pin 13) und somit als Einzelimpuls. Betätigen wir den Taster, so kehren sich sämtliche Zustände um. Um zu erkennen, ob die Handtaktschaltung richtig arbeitet, verbinden wir den Testpunkt "T" der Speicherplatine mit dem Testpunkt "T" der Prozessorplatine. Dann überprüfen wir den Taster mehrmals. Dabei leuchtet die Takt-LED im Rhythmus des Taktes, den wir mit der Hand erzeugen: die LED ist ~~aus~~, wenn der Taster gedrückt ist, sie leuchtet, wenn wir den Taster loslassen. Geschieht dies nicht?

- Der Taster arbeitet nicht, weil er schlechte Kontakte hat. Hier hilft das Ohmmeter.
- Das IC ist defekt!
- Die Leiterbahn zwischen dem Testpunkt "T" der Prozessorplatine und dem Anschluß 31 der Buchsenleiste für die Anzeigeplatine ist unterbrochen.

Wir können nun die Leitung zwischen den beiden Testpunkten entfernen. Danach setzen wir das IC MC 14500, ein und testen seinen internen Taktgenerator (Siehe Abb. 5). Gleichzeitig überprüfen wir dabei die Funktion des Taktwahlschalters. Die Takt-LED leuchtet, wenn der Taktwahlschalter auf Handtakt "HT" gestellt wird. Betätigen wir den Takttaster, erlischt sie. Wir bringen jetzt den Taktwahlschalter in die mittlere Stellung, auf Langsamtakt "LT". Je nach Stellung des Potentiometers für den NE 555 erkennen wir das Aufleuchten der Takt-LED. In der Stellung Schnelltakt "ST" wirkt der prozessorinterne Taktgenerator. Dann leuchtet die LED ca. 1 Million mal in der Sekunde auf. Für das menschliche Auge hat dies den Anschein, als ob die LED dauernd leuchte. Die Takt-LED leuchtet allerdings nicht mehr so hell.
Ersetzen wir den 1oo Kiloohm-Widerstand an Pin 3 des NE 555 durch einen mit dem Wert von 560 Kiloohm, so verringern wir den Einfluß des Langsamtaktes auf den Schnelltakt.
Wenden wir uns dem Schaltbild zu. Dort markieren wir das IC 14500 blau. Es gibt dort nur noch zwei weiße Flecke. Es ist der noch fehlende Eingabebaustein und der Ausgabebaustein, die ICs 4051 und 14599. Wir markieren beide ICs im Schaltbild blau und setzen sie in ihre Sockel auf der Prozessorplatine ein. Ob sie, und damit unser nun fertig aufgebauter Computer, funktionieren, können wir ohne Programm nicht mehr testen. Dies wird in Kap. 4 geschehen.

3.6. DIE TASTATUR

Die Tastatur ist kein Luxus. Wir können unseren Computer auch ohne sie aus methodischen Gründen mit Hilfe der Programmschalter programmieren. Dies ist für den Anfang erwünscht, damit Lernende den jeweiligen Programmschritt, sein zugeordnetes Bitmuster bei den Eingabeschaltern und die dazugehörende Leuchtdiodenanzeige interpretieren können. Später, wenn dies beherrscht wird, ist diese Eingabe jedoch sehr umständlich. Abb. 39 zeigt die Tastaturplatine.

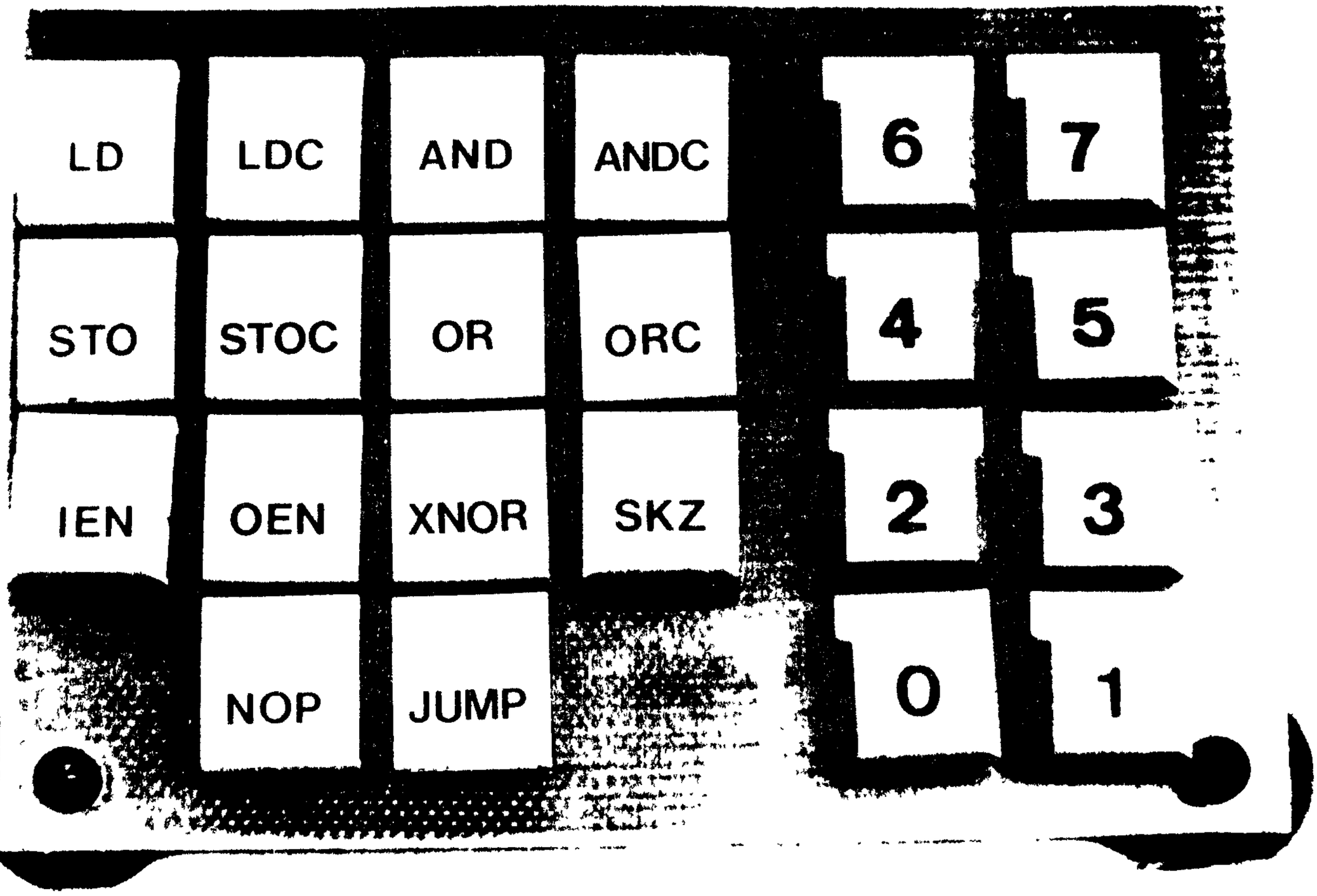


Abb. 39: Die Tastatur

Zum Bohren benötigen wir für die Füße den 5-mm Bohrer, für alle anderen Teile den 1-mm-Bohrer. Der Bestückungsaufdruck zeigt (aus Platzgründen) die Diodenschaltzeichen nur unvollständig. Die kurzen Querstriche entsprechen den Kathoden. Nach dem Einlöten der Dioden werden die beiden Drahtbrücken gesetzt. Bevor wir die Taster einsetzen, müssen wir ihre zwei Plastikstifte auf der Unterseite entfernen. Wir setzen die Taster so ein, daß ihre "Drehpunkte" in Richtung Dioden zeigen. Den Anschluß der fertig aufgebauten Tastatur an den WDR-1-Bit-Computer zeigt die Abb. 40.

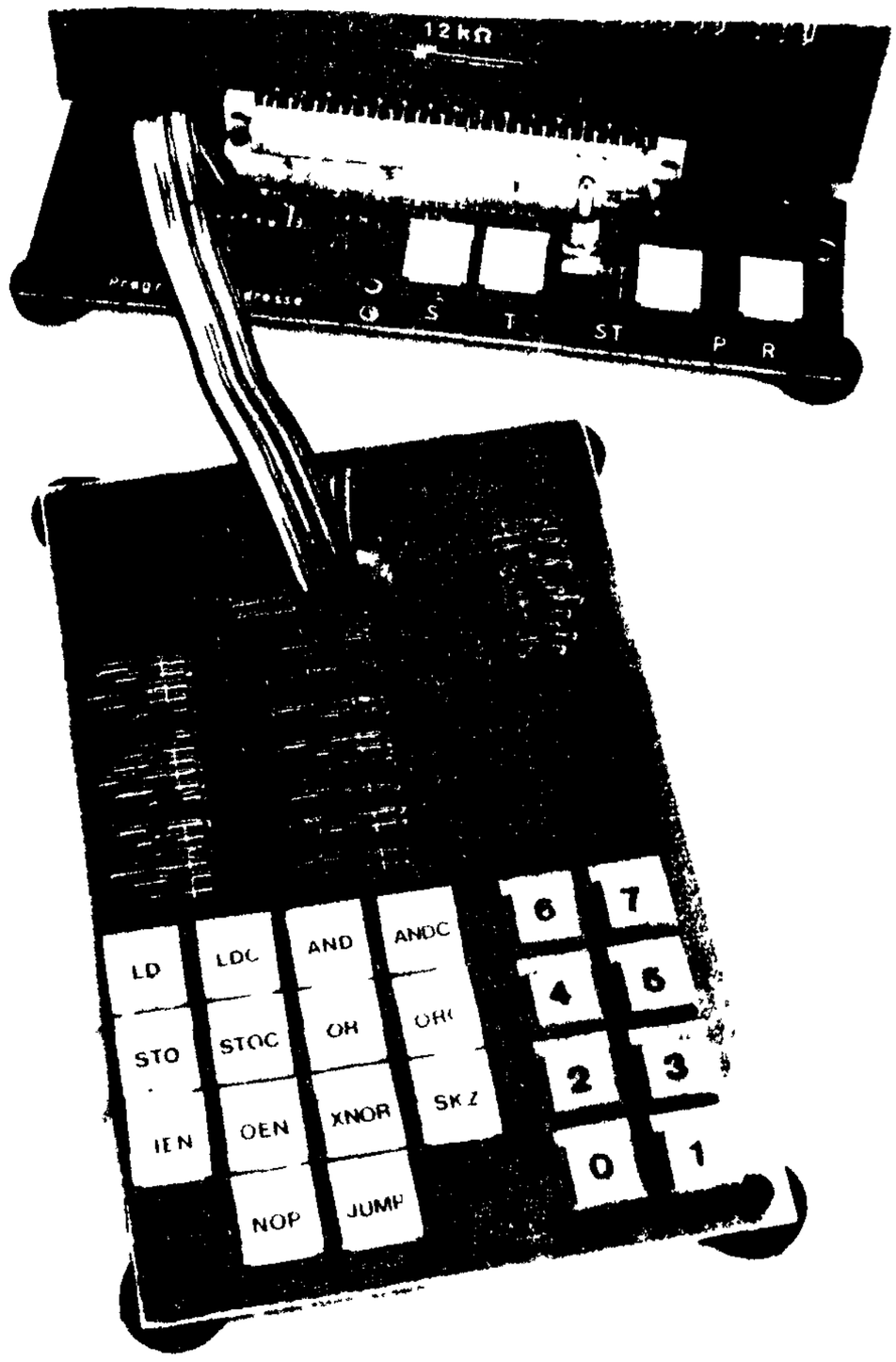


Abb. 40: Anschluß der DATANorf-Tastatur an den WDR-1-Bit-Computer.

Vor dem Einlöten des Flachbandkabels in die Tastatur bzw. den mitgelieferten IC-Sockel müssen ihre Enden kurz (!) abisoliert werden. Zuerst löten wir das Kabel in die Tastatur ein. Die Reihenfolge der Farbe beim Flachbandkabel spielt keine Rolle. Die linke Ader des Kabels stellt nach dem Einlöten die Masseleitung dar, die anderen Adern sind die vier Programm- und drei Adreßleitungen in der Reihenfolge von links nach rechts. An das andere Ende des Flachbandkabels löten wir einen IC-Sockel nach folgendem Gesichtspunkt ein: Lötpunkte sind die Fassungspunkte oben auf dem Sockel, also nicht die unteren Sockelanschlüsse. Der Masseleitung des Kabels gehört in die untere Reihe, am besten nach links unten. Die anderen Adern löten wir in der oben angegebenen Reihenfolge links oben beginnend an. Wenn wir das Kabel auf der Tastatur noch zusätzlich befestigen, kann es sich nicht mehr so leicht lösen und an den Lötstellen abreißen.

4. DIE PROGRAMMIERUNG DES WDR-1-BIT-COMPUTERS: DIE SOFTWARE

Nun ist es soweit, nachdem wir unseren Computer - hoffentlich erfolgreich - aufgebaut und getestet haben, werden wir ihn jetzt programmieren. Die einfacheren Programme simulieren logische Verknüpfungen und deren Anwendung, z.B. Flipflops. Andere Programme führen von einer Motorsteuerung bis hin zu komplexen Robotersteuerungen. Dabei zeigt sich der WDR-1-Bit-Computer von seiner starken Seiten: Seine Programme sind einfach und überschaubar, weil sein Befehlssatz überblickbar bleibt. Er demonstriert das Wesen der industriellen Revolution durch Mikroprozessoren. In Maschinensprache geschriebene Programme zeigen Lernenden in elementarer Form, wie Roboter und andere Maschinen steuerbar sind.

In diesem Abschnitt wollen wir die Programmierung des WDR-1-Bit-Computers behandeln. Der englische Fachausdruck dafür ist "Software". Zunächst muß aber der Befehlssatz der CPU näher kennengelernt werden. Dies geschieht in Kapitel 4.1.

4.1. DER BEFEHLSSATZ DES MIKROPROZESSORS

Befehlscode	Name	CPU-Tätigkeit
0000	NOP	No OPeration - Keine Operation Die Kontrolleinheit gibt nur einen 5 V-Impuls auf Pin 10 ab. Sonst geschieht nichts. Da Pin 10 nicht beschaltet ist, dient der Befehl nur dazu, in das Programm Lücken bzw. Pausen einzubauen. Dieser Befehl heißt eigentlich NOPO. Da aber der Befehl NOPF in diesem System nicht existiert, soll der Befehl NOPO zur Vereinfachung NOP genannt werden.
0001	LD	LoaD - Laden Die CPU holt eine Information über die Datenleitung und speichert sie, wenn die Eingabe freigegeben ist, im Ergebnisregister. Ist die Eingabe nicht freigegeben, so wird dort eine "0" gespeichert.
0010	LDC	LoaD Complement - Komplement laden Die CPU holt eine Information über die Datenleitung und speichert das Komplement, wenn die Eingabe freigegeben ist, im Ergebnisregister. Ist die Eingabe nicht freigegeben, so wird dort eine "1" gespeichert.
0011	AND	UND-Funktion Es wird die UND-Funktion gebildet aus dem Inhalt des Ergebnisregisters und der Information auf der Datenleitung. Das Ergebnis steht dann im Ergebnisregister. Hier steht nur dann eine "1", wenn vorher im Ergebnisregister und auf der Datenleitung eine "1" gestanden hat und die Eingabe freigegeben ist.
0100	ANDC	UND-Funktion mit dem Komplement der Datenleitung Es wird die UND-Funktion gebildet aus dem Inhalt des Ergebnisregisters und dem Gegenteil der Information auf der Datenleitung. Ins Ergebnisregister kommt nur dann eine "1", wenn dort vorher eine "1" und auf der Datenleitung eine "0" gestanden hat, oder die Eingabe gesperrt ist.

0101	OR	ODER-Funktion Es wird die ODER-Funktion gebildet aus dem Inhalt des Ergebnisregisters und der Information auf der Datenleitung. Ins Ergebnisregister kommt nur dann eine "0", wenn dort vorher eine "0" und auf der Datenleitung eine "0" gestanden hat oder die Eingabe gesperrt ist.
0110	ORC	ODER-Funktion mit dem Komplement der Datenleitung Es wird die ODER-Funktion gebildet aus dem Inhalt des Ergebnisregisters und dem Gegenteil der Information auf der Datenleitung. Ins Ergebnisregister kommt nur dann eine "0", wenn dort vorher eine "0" und auf der Datenleitung eine "1" gestanden hat und die Eingabe freigegeben ist.
0111	XNOR	eXclusive NOR - Äquivalenzfunktion Es wird die Äquivalenzfunktion gebildet aus dem Inhalt des Ergebnisregisters und der Information auf der Datenleitung. Ins Ergebnisregister kommt nur dann eine "1", wenn dort vorher das Gleiche gestanden hat wie auf der Datenleitung, oder wenn im Ergebnisregister eine "0" stand und die Eingabe gesperrt ist.
1000	STO	STOre - Speichern Die CPU gibt den Inhalt des Ergebnisregisters über die Datenleitung aus, wenn die Ausgabe freigegeben ist. Ist die Ausgabe nicht freigegeben, so wird nichts ausgegeben. In keinem Fall ändert sich der Inhalt der Ergebnisregisters.
1001	STOC	STOre Complement - Komplement speichern Die CPU gibt das Gegenteil des Inhaltes des Ergebnisregisters über die Datenleitung aus, wenn die Ausgabe freigegeben ist. Auch hier bleibt der Inhalt des Ergebnisregisters erhalten.
1010	IEN	Input ENable - Eingabe freimachen Die CPU holt eine Information über die Datenleitung und speichert sie im IEN-Register. (Siehe: Kapitel 2.2.1) Der Inhalt des Ergebnisregisters ändert sich nicht.
1011	OEN	Output ENable - Ausgabe freimachen Die CPU holt eine Information über die Datenleitung und speichert sie im OEN-Register. (Siehe: Kapitel 2.2.1) Der Inhalt des Ergebnisregisters bleibt dabei erhalten.
1100	JMP	JuMP - Springen Die Kontrolleinheit der CPU gibt einen 5 V-Impuls auf Pin 12 ab. Dieser Impuls setzt den Programmzähler auf die Programmadresse 0, denn Pin 12 ist über eine Diode mit Pin 1 des Programmzählers verbunden. Die Diode verhindert, daß beim Betätigen des Programmanfangs-Tasters der JMP-Ausgang der CPU zerstört wird.
1101	RTN	Dieser Befehl wird hier nicht benutzt. Wäre er vorgesehen, würde die Kontrolleinheit der CPU einen 5V-Impuls auf den hier nicht beschalteten PIN 11 abgeben. Außerdem würde der nächste Befehl übersprungen werden zum übernächsten.

1110	SKZ	SKip if Zero - Überspringen, wenn Null. Falls im Ergebnisregister eine "1" steht, führt die CPU den nächsten Befehl normal aus. Falls dort eine "0" steht, überspringt die CPU den nächsten Befehl und geht zum übernächsten weiter.
1111	NOPF	No OPeration Function - keine Operationsfunktion Dieser Befehl ist hier auch nicht vorgesehen. Er würde einen 5V-Impuls von der CPU an den hier nicht beschalteten PIN 9 hervorrufen.

Besonderheiten:
Die Steuerbefehle NOP, JMP und der Programmverzweigungsbefehl SKZ benötigen keine Ein-/Ausgabeadresse, da sie keine Information der Datenleitung brauchen und auch keine Information über sie ausgeben. Anstatt einer bestimmten Ein-/Ausgabeadresse wird in diesem Abschnitt in Programmen hinter diesen Befehl ein x geschrieben. Dieses x kann also 0, 1, 2, 3, 4, 5, 6, oder 7 heißen. Außerdem bleibt auch bei diesen Befehl der Inhalt des Ergebnisregisters erhalten.Die Eingangsleitung Nr. 0 (E0) ist mit dem Ergebnisregister verbunden (Siehe: Abb. 2). Die Ein-/Ausgabeadresse 0 stellt in Zusammenhang mit den Eingabebefehlen und den Rechenbefehlen eine weitere Besonderheit dar. Der Programmschritt LD 0 ist ein überflüssiger Befehl. Er hat folgende Wirkung: Der Inhalt des Ergebnisregisters wird in das Ergebnisregister gebracht. Der Programmschritt LDC 0 invertiert immer den Inhalt des Ergebnisregisters, d.h., er bildet das Komplement. Bei den Rechenbefehlen wird der Inhalt des Ergebnisregisters mit sich selbst verknüpft. Bei den Programmschritten ORC 0 und XNOR 0 gelangt immer eine "1" in das Ergebnisregister. Beim Programmschritt ANDC 0 gelangt immer eine "0" in das Ergebnisregister. Bei den Programmschritten AND 0 und OR 0 wird der Inhalt des Ergebnisregisters nicht verändert. Voraussetzung hierbei ist, daß die Eingabe freigegeben ist.

4.2. DAS EINGEBEN VON PROGRAMMEN ÜBER DIL-SCHALTER

Wir geben jeden einzelnen Programmschritt ein, indem wir die vier Befehlsschalter und die drei Adressenschalter betätigen, entsprechend dem Aufdruck auf der Grundplatine links unten. Diese Schalter sind Teil des 8-fach DIL-Schalters, der sich auf dem 16-poligen Sockel der Grundplatine befindet. Die Schalter 1-4 dienen der Programmeingabe, die Schalter 6-8 der Adresseingabe. Der Schalter 5 wird also nicht benutzt.
Das mit den Schaltern eingestellte Bitmuster erscheint, nachdem wir den Schreibtaster betätigt haben, auf den 7 Programmleuchtdioden der Arzeigeplatine. Bevor der nächste Programmschritt eingegeben werden kann, betätigen wir noch den Takttaster, dadurch wird der Zählerstand um 1 erhöht. Damit ergibt sich das folgende Muster, um einzelne Programmschritte in den WDR-1-Bit-Computer einzuschreiben:

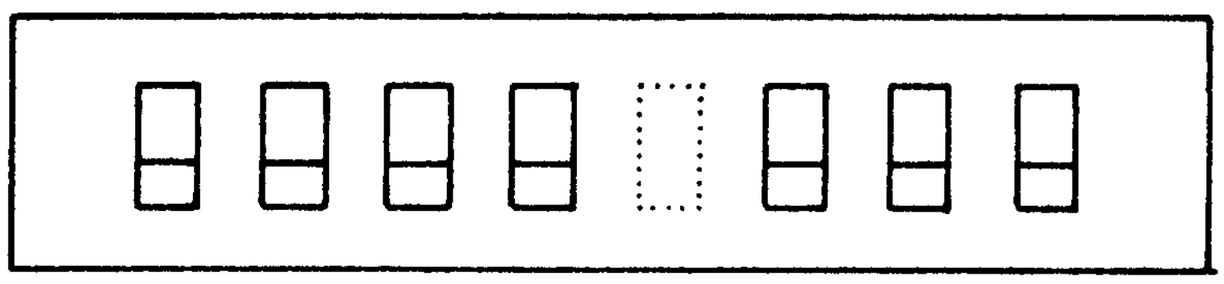


Abb. 41: Schema der Eingabe eines Programmschritts

Wenn wir den Computer eingeschaltet haben, stehen in seinem Speicher irgendwelche zufälligen Informationen, die uns stören können wenn wir programmieren. Wir setzen daher mit dem Programmschritt NOP 0 alle Speicherzellen auf Null:

1. Taktwahlschalter auf Schnelltakt "ST" stellen.
2. NOP 0, also das Bitmuster 0000 000, am 8-fach DIL-Schalter einstellen.
3. Den Schreibtaster "S" betätigen, wodurch alle Programmspeicher auf 0 gesetzt werden.
4. Den Taktwahlschalter auf Handtakt "HT" stellen.
5. Den Programmanfangstaster "P" betätigen. Dadurch wird der Programmzähler auf 0 gesetzt.
6. Den Rücksetztaster "R" drücken, wodurch alle Informationen in den Registern des Prozessors und des Ausgabebausteins gelöscht werden.

Der 5. Schritt sorgt also dafür, daß jedes Programm bei der Programmadresse 0 bzw. binär "0000 0000" beginnt. Aus Gründen der Vereinfachung werden wir bei unseren Programmen den Stand des Programmzählers nicht binär sondern dezimal angeben, da unser Computer den Wert des Zählers nicht anzeigt.
Nach den ersten sechs Vorbereitungsschritten, die der besseren Übersicht beim Programmieren dienen, folgen hieraus drei zusätzliche Schritte, um den Computer für weitere Eingaben durch den Programmierer vorzubereiten.

FREIGABE FÜR AUS- UND EINGABE

Die Ein- und die Ausgabe von Daten in die Logikeinheit (LU = logical unit) bzw. aus der LU muß erst mit Hilfe zweier Register, des IEN- und des OEN-Registers, freigegeben werden.
Das IEN-Register wird durch den IEN-Befehl (Eingangsfreigabe-Befehl) Input-ENable-Befehl geladen.
Ist im IEN-Register einen "0" gespeichert, so können keine Daten in die CPU eingelesen werden, denn am Ausgang des IEN-Registers liegt dann immer eine "0", unabhängig von ihren tatsächlichen Wert. Ist im IEN-Register hingegen eine "1" gespeichert, so werden alle Daten korrekt eingelesen.
Um zu gewährleisten, daß Eingangssignale auch tatsächlich in die Logik-Einheit gelangen, muß dafür gesorgt werden, daß im IEN-Register eine "1" steht.
Wir müssen also dafür sorgen, daß einer der Eingänge einen von uns erwünschten logischen Zustand hat, eine "1" oder eine "0". Betrachten wir den Eingang E0 . Er ist direkt mit dem Ausgang des ER verbunden. Wir können daher mit dem Befehl "IEN 0" den Pegel des FR in das IEN-Register laden. Bleibt noch zu erreichen, daß im ER eine 1 steht. Dies erreichen wir mit dem Befehl ORC 0.

Ist die Eingabe gesperrt, so liest die CPU das mit dem Befehl ORC 0 eine "0" vom Ausgang des UND-Gatters ein. (Siehe Kapitel 2.2.1.-Eingangsschaltung der CPU). Die "0" wird in der Logikeinheit invertiert, also zur "1" und dann mit dem Inhalt des ER ODER-verknüpft. Das Ergebnis dieser Verknüpfung ist immer eine "1", es wird ins ER übernommen und kann dann über Pin 15 der CPU über die Eingabeeinheit und über die Datenleitung ins IEN-Register geladen werden. Danach ist die Eingabe von Daten freigegeben.

Mit der Befehlsfolge ORC 0, IEN 0 können wir also erreichen, daß Signale über die Datenleitung in die CPU eingelesen werden. Wir müssen nun noch dafür sorgen, daß unser Computer Daten auch ausgeben kann.
Hierfür ist das OEN-Register in Verbindung mit dem OEN-Befehl zuständig (OEN steht für Output ENable, also für Ausgangsfreigabe.
Das OEN-Register wird in gleicher Weise wie das IEN-Register über die Datenleitung geladen. Steht im OEN-Register eine "1", so können Daten mit Hilfe des STO- oder STOC-Befehls über die Datenleitung an die Ausgabeeinheit gelangen. (Siehe Kapitel 2.2.1.-Ausgangsschaltung der CPU).
Es ist also auch notwendig, das OEN-Register mit einer "1" zu laden, um den Mikroprozessor und damit unseren Computer zur Programmbearbeitung vorzubereiten. Dieses Vorbereiten nennt man Initialisieren.
Zusammenfassend lautet unser Initialisierungsprogramm INIT also wie folgt:

INIT

Programmadresse	Programmschritt Befehl	Adresse
001	ORC	0
002	IEN	0
003	OEN	0

Dieses Initialisierungsprogramm besteht aus drei Programmschritten. Es steht immer am Anfang eines jeden Programms und wird in Zukunft mit INIT abgekürzt:

000 INIT

Und so werden diese drei Programmschritte in den Computer eingegeben:
7. ORC0, also das Bitmuster 0110 000, am 8-fach DIL-Schalter einstellen und den Schreibtaster betätigen.
8. Danach wird der Takt-Taster einmal betätigt. Hierdurch springt der Programmzähler auf die Programmadresse 1. Gleichzeitig wird der Programmschritt ORC 0 von der Programmadresse 0 ausgeführt.
9. In gleicher Weise wie ORC 0 werden auch IEN 0, binär: 1010 000, und und OEN 0, binär: 1011 000, eingeben.

(In Programmen, in denen keine Daten eingegeben werden brauchen, kann der Programmschritt IEN 0 fehlen. Genauso kann OEN 0 fehlen, wenn keine Daten ausgegeben werden sollen.)
Von großer Bedeutung ist, daß der ER-Ausgang mit dem Eingang E0 verbunden ist und daß der Inhalt des ER durch die Schritte 2 und 3 nicht verändert wird.
Ausgang und Eingang sind durch diese Initialisierung nun freigegeben. Erst jezt werden alle weiteren Programmschritte korrekt ausgeführt.

4.3 EINFACHE PROGRAMME

4.3.1. PROGRAMME ZUR EINGABE VON DATEN

In den WDR-1-Bit-Computer können Informationen mit Hilfe des 4-fach DIL-Schalter oder über die Anschlüsse 3 bis 5 und 14 des Expansionssteckers mit den Eingabebefaehlen LD oder LDC eingegeben werden. Diese Anschlüsse sind, wie das Schaltbild zeigt, miteinander

verbunden. Schließen wir die 4 Eingangsschalter oder legen wir eine positive Spannung von 5 Volt an die genannten Anschlüsse des Expansionssteckers, so leuchten die entsprechenden 4 Eingangsleuchtdioden.
Das folgende Programm wird eingegeben, es zeigt uns, wie der logische Pegel des Eingangs gelesen und im Ergebnisregister abgespeichert wird:

```
**Eingang E1 einlesen**

00 INIT     INIT=Initialisierung, also die Programmschritte
            *
            00 ORC 0 , binär: 0110 000
            01 IEN 0 , binär: 1010 000
            02 OEN 0 , binär: 1011 000
            *
03 LD       Lade den Pegel des Eingangs 1, binär: 0001 001
```

Wenn wir das Programm starten, indem wir den Zähler mit dem Programmvorwahltaster "P" auf den Programmanfang setzen und dann z.B. den Takttaster viermal betätigen, zeigt uns die "ER"-Leuchtdiode über den Inhalt des Ergebnisregisters an, welcher Eingangspegel am Eingang 1 gerade anliegt.
Wenn wir nun weitertakten, durchläuft der Computer unnötigerweise sämtliche restlichen 252 Speicherstellen. Um dies zu verhindern, lautet der nächste Befehl: Springe an den Anfang zurück, also Jump (JMP):

```
04 JMP x    binär 1100 xxx Das x bedeutet eine beliebige Adresse.
            Eine Adressenangabe erübrigt sich bei diesem Befehl, da
            weder die Eingabe- noch die Ausgabeeinheit
            angesprochen werden.
```

Soll der einzulesende Eingangswert umgekehrt werden, setzen wir anstelle des Befehls LD (Laden) den Befehl LDC (Laden Komplement). Der Programmschritt 03 lautet dann:
03 LDC 1 , binär: 0010 001
Dieser Befehl erspart zusätzliche Hardware (in diesem Falle den Einbau eines Inverters), wenn aus programmtechnischen Gründen invertierte Signale benötigen.
Bisher haben wir nur den Pegel des Eingangs E1 eingelesen. Wollen wir die Pegel der anderen drei Eingänge (E2-E4) in den Computereingeben, ändern wir die Adresse: LD 3 (binär 0001 011) lädt z.B. den Pegel des Eingangs E3.

4.3.2 PROGRAMME ZUR AUSGABE VON DATEN

Um die im Computer verarbeiteten Eingangssignale auszugeben, wenden wir die Ausgabebefehle STO bzw. STOC an. Dadurch wird der sich im Ergebnisregister befindende Pegel bzw. dessen Komplement auf die Ausgänge A0 bis A7, das sind die Anschlüsse 6 - 13 des Expansionssteckers, gelegt und dort gespeichert. Daher der Befehlsname Store.
Auch hier gilt wieder, daß die Ausgangsadresse mit dem Befehl eingegeben werden muß, will man einen bestimmten Ausgang anwählen.
Ist z.B. der Wert des Ergebnisregisters "1", und soll der Inhalt des Ergebnisregisters auf den Ausgang 4 gelesen werden, so lautet der entsprechende Befehl STO 4.
Wie bei der Eingabe kann auch bei der Ausgabe das Komplement gebildet werden. STOC 4 gibt also das Gegenteil des Wertes des Ergebnisregisters auf A4.
Das folgende Programm schaltet den Ausgang A4 immer wieder ein und aus. Es handelt sich um einen Rechteckgenerator oder astabilen Multivibrator.

```
**Rechteckgenerator**

00 INIT
03 STO  4   binär: 1000 100
04 STOC 4     "    1001 100
05 JMP  x     "    1100 xxx
```

Soll das Programm automatisch ablaufen, und wollen wir es dabei beobachten, stellen wir den Taktwahlschalter auf Langsamtakt "LT". Durch Variieren des Potentiometers kann die Frequenz der Rechteckpulse geändert werden. Dabei beobachten wir, daß das Tastverhältnis zwischen hell und dunkel der Ausgangsleuchtdiode unsymmetrisch ist. Soll unser Rechteckgenerator dazu benutzt werden, gleich lange Impulse zu liefern, müssen wir es ändern. Die Binärwerte übertragen wir einfach aus dem obigen Programm: ** Rechtecksgenerator **

```
**Rechteckgenerator mit symmetrischem Tastverhältnis**

00 INIT
03 STO  4
04 NOP  x
05 NOP  x
06 NOP  x
07 NOP  x
08 STOC 4
0  JMP  x
```

Da die Ausgänge beliebig beschaltet werden können, ist es möglich, ein Lauflicht in den verschiedensten Variationen zu programmieren. Das folgende Programm zeigt ein solches Lauflicht:

```
**Lauflicht**

00 INIT
03 STO  0
04 STO  1
05 STO  2
06 STO  3
07 STO  4
```

```
08 STO  5
09 STO  6
13 STO  7
11 STOC 7
12 STOC 6
13 STOC 5
14 STOC 4
15 STOC 3
16 STOC 2
17 STOC 1
18 STOC 0
19 JMP  x
```

Natürlich können wir unser Programm um weitere 237 Programmschritte erweitern. Dann allerdings empfehlen wir die Benutzung der Tastatur.

4.4. ES WIRD BEQUEMER: DIE TASTATUR

Die Eingabe von Programmen über den DIL-Schalter erweist sich zu Beginn als sehr lehrreich, jedoch auch als sehr unkomfortabel. Zudem muß immer vor der Eingabe eines Programmschrittes das Bitmuster dieses Programmschritts ermittelt werden. Daher benutzen wir in Zukunft die Tastatur, um Programme einzugeben.
Zunächst entfernen wir den 8-poligen DIL-Schalter und setzen an seine Stelle den Tastaturstecker so ein, daß die 7 Anschlüsse in der hinteren Reihe stecken, der eine Anschluß also vorne liegt. Die Tastatur ist damit schon gebrauchsfähig.
Um einen Programmschritt einzugeben, drücken wir den gewünschten Befehlstaste und den Adressentaster gleichzeitig (!). Zusätzlich betätigen wir mit der anderen Hand den Schreibtaster. Damit ist der Programmschritt schon gespeichert. Nach der Betätigung des Takttasters können wir auf dieselbe Art den nächsten Programmschritt eingeben.

4.5. WEITERE PROGRAMME

4.5.1. LOGISCHE VERKNÜPFUNGEN

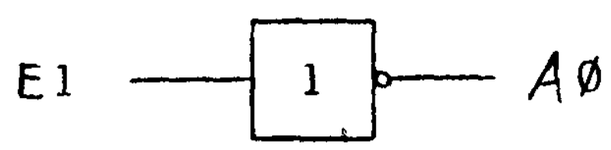
In Kapitel 2.2.1. haben wir die wichtigsten logischen Verknüpfungen kennengelernt. In diesem Kapitel sollen logische Verknüpfungen mittels eines Programmes mit dem Computer verwirklicht werden.
Gleichzeitig sollen die Rechenbefehle AND, ANDC, OR, ORC und XNOR kennengelernt werden.
Die Eingänge einer logischen Schaltung sind E1, E2, E3 bis E4. Das Ergebnis der logischen Verknüpfung wird auf A0 angezeigt. Ist ein Programm eingegeben, so wird der Taktwahlschalter auf Stellung "Schnell-Takt" gestellt. Betätigt man die Schalter für die Eingänge, so erscheint an A0 sofort das Ergebnis der logischen Verknüpfung.

1. NICHT-Glied (Inverter)

Wahrheitstabelle:

E1	A0
0	1
1	0

Schaltzeichen:



Programm:

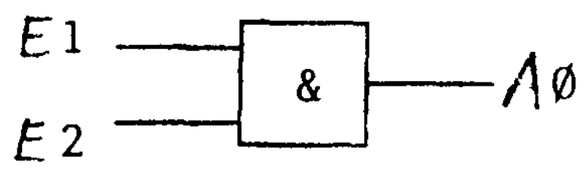
```
00 INIT
03 LD   1
04 STOC 0
05 JMP  x
```

2. UND-Glied mit zwei Eingängen

Wahrheits_abelle:

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

Schaltung:



Programm:

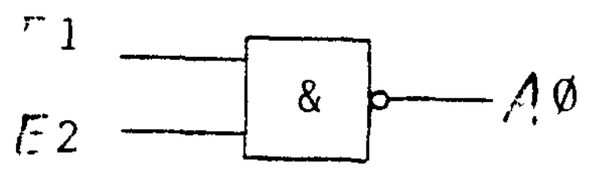
```
00 INIT
03 LD   1
04 AND  2
05 STO  0
06 JMP  x
```

3. NAND-Glied mit zwei Eingängen

Wahrheitstabelle:

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

Schaltung:



Programm:

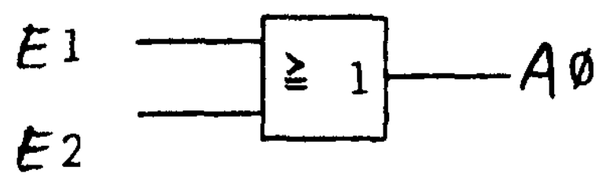
```
00 INIT
03 LD   1
04 AND  2
05 STOC 0
06 JMP  x
```

4. ODER-Glied mit zwei Eingängen

Wahrheitstabelle:

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

Schaltung:



Programm:

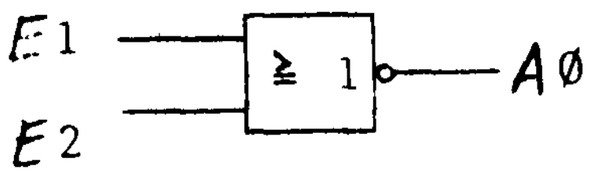
```
00 INIT
03 LD   1
04 OR   2
05 STO  0
06 JMP  x
```

5. NOR-Glied mit zwei Eingängen

Wahrheitstabelle:

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

Schaltung:

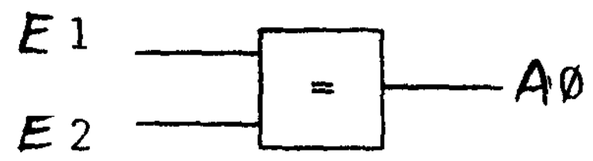


Programm:

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

6. Äquivalenz-Glied (Exklusiv-NOR-Glied) mit zwei Eingängen

Schaltung:



Wahrheitstabelle:

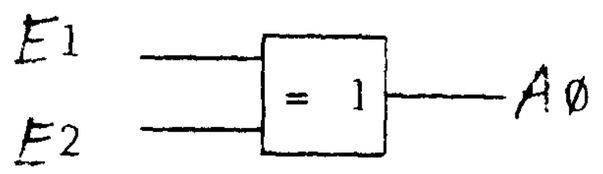
E2	E1	AO
0	0	1
0	1	0
1	0	0
1	1	1

Programm:

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

7. Antivalenz-Glied (Exklusiv-ODER-Glied) mit zwei Eingängen

Schaltung:



Wahrheitstabelle:

E2	E1	AO
0	0	0
0	1	1
1	0	1
1	1	0

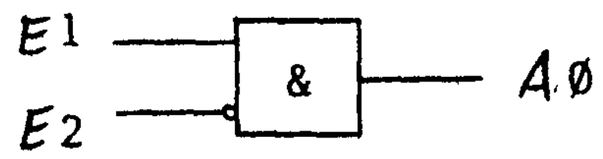
Programm:

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

8. Inhibitations-Glied

Die Inhibitation ist eine besondere Art der UND-Verknüpfung.
Ein Eingangszustand wird vor der UND-Verknüpfung negiert.

Schaltung:



Wahrheitstabelle:

E2	E1	AO
0	0	0
0	1	1
1	0	0
1	1	0

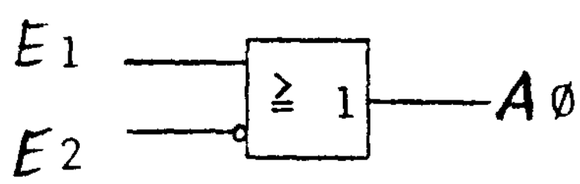
Programm:

```
00 INIT
03 LD   1
04 ANDC 2
05 STO  0
06 JMP  x
```

9. Implikations-Glied

Die Implikation ist eine besondere Art der ODER-Verknüpfung.
Ein Eingangszustand wird vor der ODER-Verknüpfung negiert.

Schaltung:



Wahrheitstabelle:

E2	E1	AO
0	0	1
0	1	1
1	0	0
1	1	1

Programm:

```
00 INIT
03 LD   1
04 ORC  2
05 STO  0
06 JMP  x
```

9. Vergleich zweier logischer Schaltungen

Als Beispiel nehmen wir das 1. de Morgansche Gesetz, dessen allgemeine Gültigkeit unser Computer überprüfen soll.
Es lautet: $\overline{E1 \vee E2} = \overline{E1} \wedge \overline{E2}$
Das Programm:

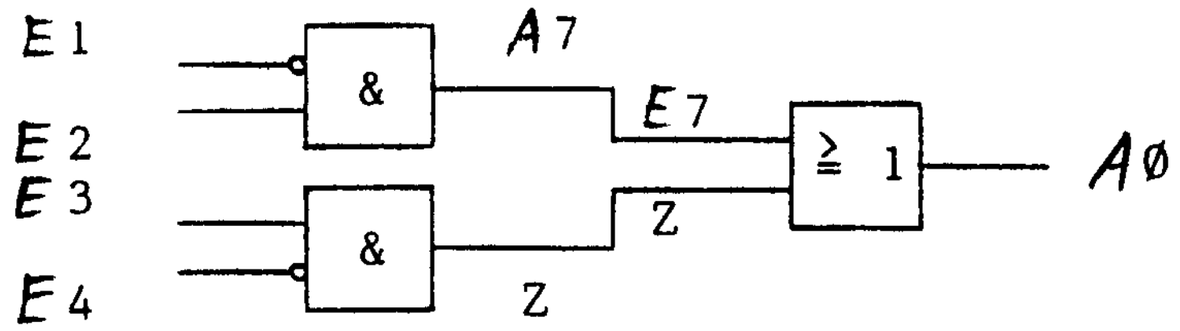
```
** 1. de Morgansches Gesetz **
*
00 INIT
03 LD   1
04 OR   2
05 STOC 7
06 LDC  1
07 ANDC 2
08 XNOR 7
09 STO  0
10 JMP  x
```

Im 5. Programmschritt wird das Komplement der Verknüpfung E1 ∨ E2 zwischengespeichert. Als Zwischenspeicher stehen die drei Datenspeicher A5, A6 und A7 zur Verfügung. Im 8. Schritt werden dann die beiden Seiten der Gleichung miteinander verglichen. Um das Gesetz zu überprüfen, betätigen wir die Eingangsschalter E1 und E2. Das Ergebnis dieses Vergleichs liegt am Ausgang A0 an. Leuchtet die Ausgangsleuchtdiode, dann liegt Gleichheit vor.

10. Logische Schaltung mit mehreren Gliedern

Möchte man mehrere Glieder miteinander verbinden, so muß beachtet werden, daß das Ergebnis jeder parallelen logischen Verknüpfung, bis auf das Ergebnis der letzten parallelen logischen Verknüpfung, zwischengespeichert werden muß.
Im folgenden Beispiel wird der Datenspeicher A7 benutzt. Dort erscheint immer das Ergebnis der ersten logischen Verknüpfung. Das Ergebnis der zweiten logischen Verknüpfung wird willkürlich "Z" genannt.

Schaltung:



Wahrheitstabelle:

E4	E3	E2	E1	A7/E7	Z	A0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	1
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	1
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	1	0	1
1	1	1	1	0	0	0

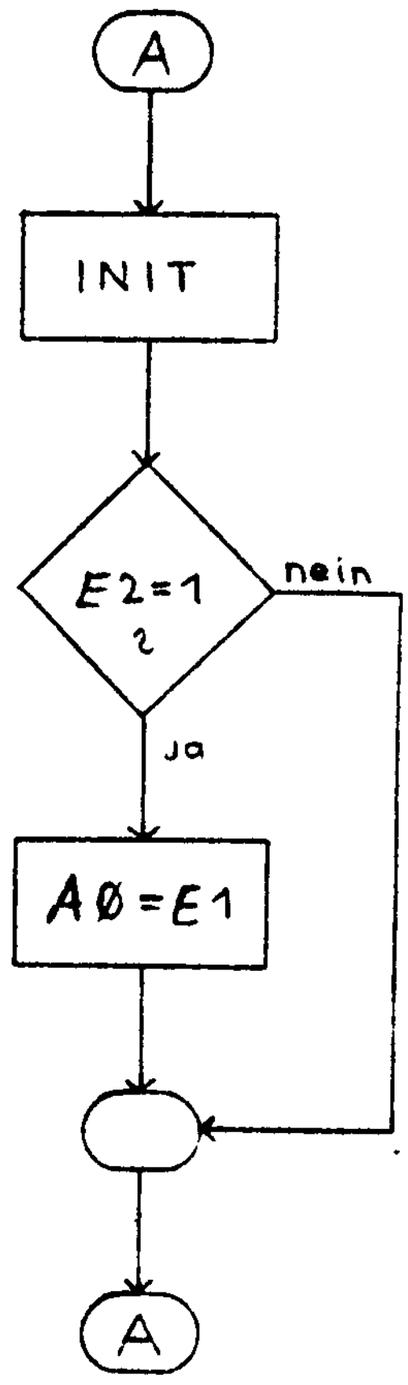
```
Programm:       00 INIT
                03 LDC  1
                04 AND  2
                05 STO  7
                06 LD   3
                07 ANDC 4
                08 OR   7
                09 STO  0
                10 JMP  x
```

4.5.2. PROGRAMMVERZWEIGUNGEN

Mit Hilfe des IEN-Befehls, des OEN-Befehls und des SKZ-Befehls ist es möglich, innerhalb eines Programmes nur bestimmte Programmblöcke zu bearbeiten, andere Programmabschnitte werden dagegen ignoriert. Dem IEN-Befehl kommt dabei nur eine geringe Bedeutung zu.
Hier sollen also nur der OEN-Befehl und der SKZ-Befehl behandelt werden.
Um die Programmverzweigung deutlich zu machen, werden die Programme mit Hilfe eines Flußdiagrammes dargestellt. Ein Flußdiagramm gibt den strukturellen Verlauf eines Programmes wieder. Es besteht aus Marken, durch Kreise oder Ellipsen gekennzeichnet, aus Abfragen, durch Karos gekennzeichnet und aus Zuordnungen, Durch Rechtecke gekennzeichnet. Marken stehen immer am Programmanfang (Abkürzung: A), am Programmende und dort, wo Programmzweige wieder zusammenlaufen. Abfragen stehen immer dort, wo Entscheidungen nötig sind. Deswegen verzweigt sich hier das Programm in zwei Programmzweige. Die Abfrage kann nur mit "ja" oder "nein" beantwortet werden. Je nachdem wird der eine oder der andere Programmzweig bearbeitet. In den Zuordnungen stehen arithmetische Anweisungen.

1. Die WENN-DANN-Programmierung

Die Benutzung der Bezeichnung WENN-DANN sagt aus: WENN ein Zustand eingetreten ist, DANN soll ein bestimmter Programmabschnitt bearbeitet werden. Ist der Zustand dagegen nicht erreicht, so wird der bestimmte Programmabschnitt ignoriert. Die WENN-DANN-Beziehung soll an einem Beispiel verdeutlicht werden:
WENN die Ausgabe durch den Eingang E2 freigegeben ist, DANN soll der logische Pegel des Eingangs E1 am Ausgang A0 erscheinen. Ist die Ausgabe nicht freigegeben, so erscheint der logische Pegel des Eingangs E1 nicht.
Nebenstehend das Flußdiagramm!

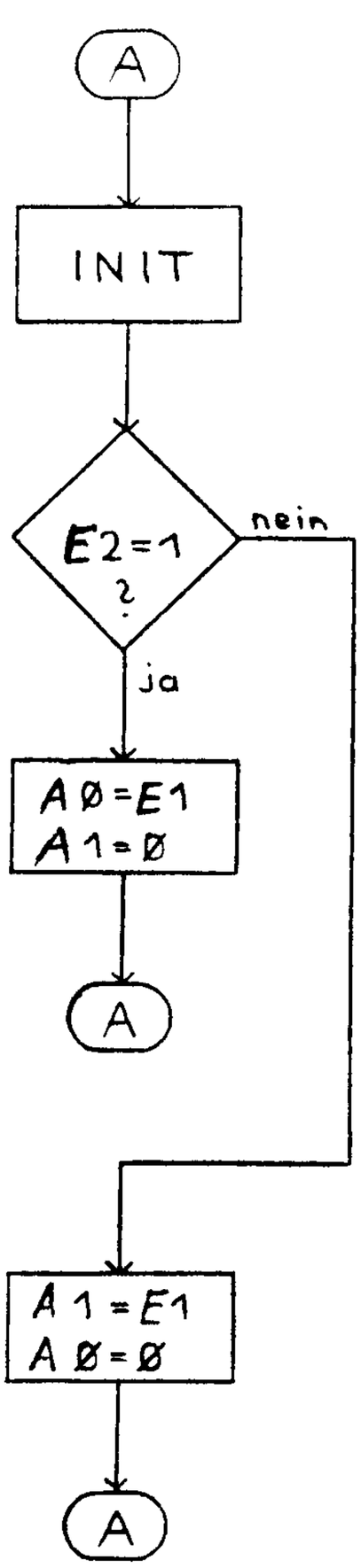



```
Programm:       00 INIT
                03 OEN  2
                04 LD   1
                05 STO  0
                06 ANDC 0
                07 STO  0
                08 JMP  x
Stellung:       Schnell-Takt
```

Die WENN-DANN-Beziehung wird mit Hilfe des OEN-Befehls möglich gemacht. OEN 2 bewirkt folgendes: Die Information des Eingangs E2 wird in das OEN-Register geladen. Steht dann dort eine "1", so wird die Information des Eingangs E1 auf den Ausgang A0 gesetzt. Steht dort jedoch eine "0", so ist die Ausgabe nicht freigegeben und STO 0 wird ignoriert. Dieses hat die gleiche Wirkung wie ein Überspringen dieses Programmschrittes. (ANDC 0 und STO 0 bewirken ein Löschen der Information des Ausgangs A0, damit nicht der Eindruck entsteht, daß eine Information ausgegeben wird, wenn die Ausgabe gesperrt ist.)
Ist dieses Programm Teil eines anderen Programmes, d.h., soll am Ende dieses Programmes nicht zurückgesprungen werden, sondern sollen weitere Programmteile bearbeitet werden, so muß die Ausgabe wieder freigemacht werden.

2. Die WENN-DANN-SONST-Programmierung

Die Benutzung der Bezeichnung WENN DANN SONST sagt aus: WENN ein Zustand eingetreten ist, DANN soll ein bestimmter Programmabschnitt bearbeitet werden; SONST soll ein anderer Programmabschnitt bearbeitet werden.
DIE WENN-DANN-SONST-Beziehung soll auch an einem Beispiel erläutert werden:
WENN die Ausgabe durch den Eingang E2 freigegeben ist, DANN soll der logische Pegel des Eingangs E1 am Ausgang A0 erscheinen; SONST, d.h., wenn die Ausgabe nicht freigegeben ist, soll er am Ausgang A1 erscheinen.
Flußdiagramm nebenstehend!



Programm:

```
00 INIT
03 LD   1
04 STO  5
05 OEN  2
06 STO  0
07 ANDC 0
08 STO  1
09 LDC  2
10 OEN  0
11 LD   5
12 STO  1
13 ANDC 0
14 STO  0
15 JMP  x
```

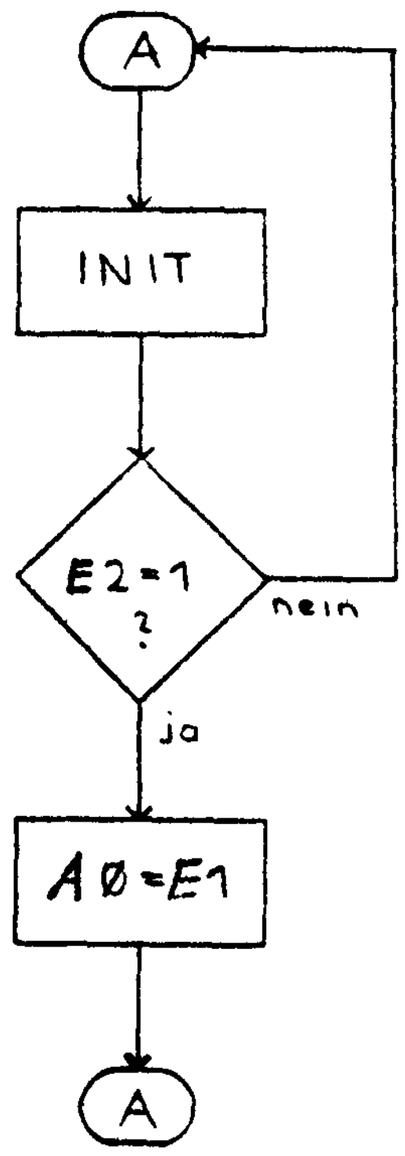
Stellung: Schnell-Takt

Die WENN-DANN-SONST-Beziehung wird mit Hilfe zweier OEN-Befehle möglich gemacht. OEN 2 bewirkt das Gleiche wie bei der WENN-DANN-Beziehung. Er entscheidet über die Freigabe des ersten Programmblocks (Programmadressen 06...08). LDC 2 und OEN 0 stellen für sich auch eine WENN-DANN-Beziehung dar. Sie

entscheiden über die Freigabe des zweiten Programmblocks (Programmadressen 11...14). Da immer nur einer der beiden Programmblöcke bearbeitet wird, ergeben beide zusammen eine WENN-DANN-SONST-Struktur.

3. Die SOLANGE-Programmierung

Die Benutzung der Bezeichnung SOLANGE sagt aus: SOLANGE ein bestimmter Zustand nicht eingetreten ist, soll das Programm in einer Warteschleife laufen. Ist der Zustand eingetreten, so wird das restliche Programm durchlaufen. Die SOLANGE-Beziehung soll ebenfalls an einem Beispiel erklärt werden:
SOLANGE die Ausgabe durch den Eingang E2 nicht freigegeben ist, soll der logische Pegel des Eingangs E1 am Ausgang A0 nicht erscheinen.



Flußdiagramm nebenstehend!

Programm:

```
00 INIT
03 LDC  2
04 SKZ  x
05 JMP  x
06 LD   1
07 STO  0
08 ANDC 0
09 STO  0
10 JMP  x
```

Stellung: Schnell-Takt

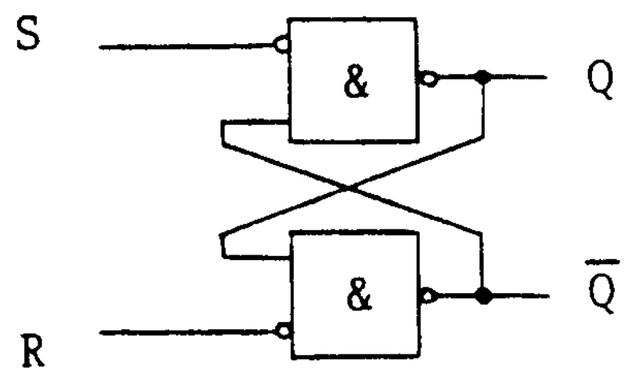
Die SOLANGE-Beziehung wird mit Hilfe des SKZ-Befehls und des JMP-Befehls möglich gemacht. Steht nach LDC 2 eine "1" im Ergebnisregister, so wird JMP x (Programmadresse 05) ausgeführt und das Programm startet wieder von neuem. Dies geschieht solange, bis mit LDC 2 eine "0" ins Ergebnisregister gelangt. Dann wird JMP x ignoriert und das restliche Programm bearbeitet.
Beim Eingeben des Programmes ist darauf zu achten, daß vor dem Einschreiben von SKZ x eine "0" im Ergebnisregister steht, da sonst nach JMP x der Programmzähler auf der Programmadresse 00 steht.

4.5.3. Programme zur Darstellung von Flipflops

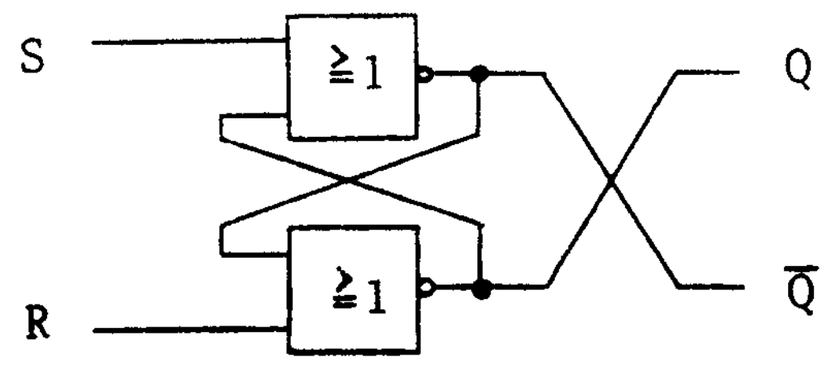
Drei Flipflop-Typen, das RS-Flipflop, das D-Flipflop und das JK-Flipflop sollen mit Hilfe von Programmen dargestellt werden. Alle Programme laufen in Stellung "Schnell-Takt" ab.

1. Ungetaktetes RS-Flipflop

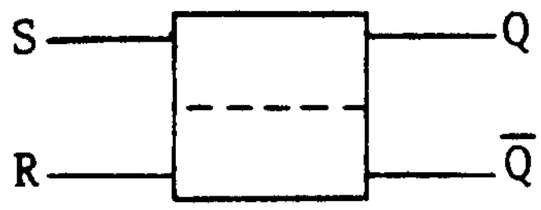
Schaltung (aufgebaut aus NAND-Gliedern):



Schaltung (aufgebaut aus NOR-Gliedern):



Schaltzeichen:



Wahrheitstabelle:

R	S	Q	$\overline{Q}$
0	0	Q-	$\overline{Q}$-
0	1	1	0
1	0	0	1
1	1	-	-

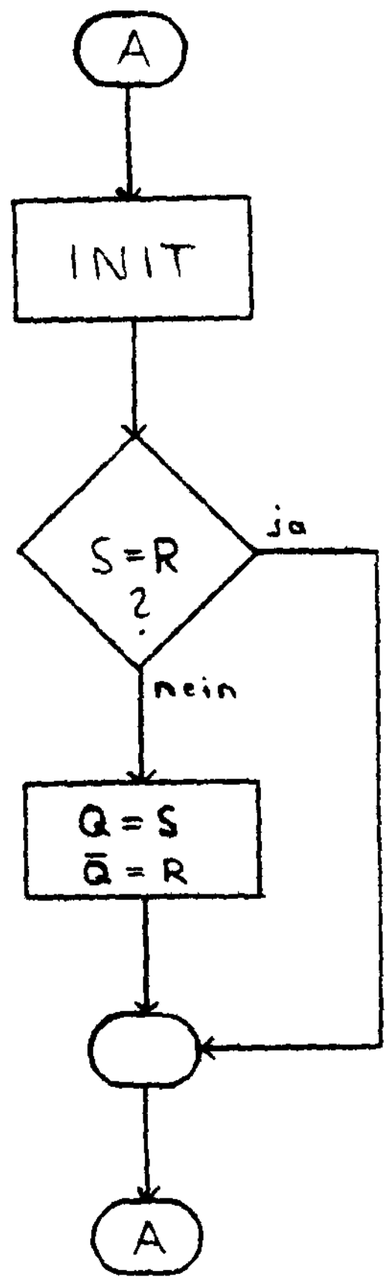
Ein RS-Flipflop kann aus NAND-Gliedern und aus NOR-Gliedern aufgebaut werden. Beide Flipflops haben einen nicht zulässigen Fall: R=S=1. Weiterhin haben sie einen Speicherfall: R=S=0. Hierbei bleibt der alte Ausgangszustand erhalten. Alte Zustände

werden in diesem Kapitel durch ein Minuszeichen gekennzeichnet. Alte Ausgangszustände heißen also Q- und $\overline{Q}$-.
Springt man vom nicht zulässigen Fall zum Speicherfall, so ist der Ausgangszustand ein Zufallsergebnis. Er ist davon abhängig, welches der beiden Glieder schneller umschaltet. Daher sollte dieser Fall vermieden werden. Im Programm wird dieser Fall durch den Speicherfall ersetzt.

Neue Wahrheitstabelle:

R	S	Q	$\overline{Q}$
0	0	Q-	$\overline{Q}$-
0	1	1	0
1	0	0	1
1	1	Q-	$\overline{Q}$-

Flußdiagramm nebenstehend!
Grundsätzlich liegt dem Flipflop eine WENN DANN Beziehung zugrunde. Um das Programm für ein Flipflop zu schreiben, müssen wir die in der Wahrheitstabelle enthaltenen Ein- und Ausgänge auf unseren Computer übertragen. Festlegung der Zuordnungen im Programm:
R = E2, S = E1, Q = A0, $\overline{Q}$ = A1.

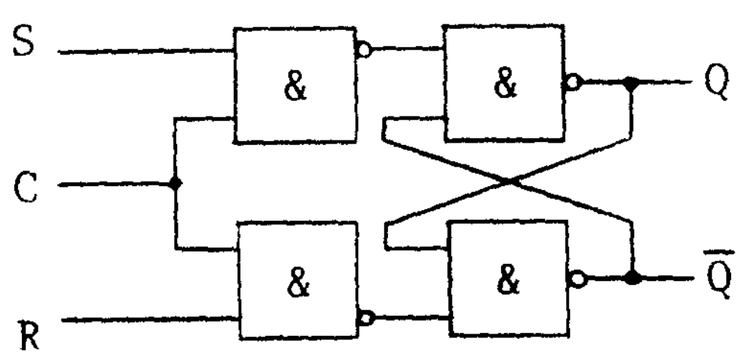



```
00 INIT                          ** Flipflop **
03 LD   1
04 STO  5
05 LDC  2
06 XNOR 5
07 OEN  0
08 LD   5
09 STO  0
10 STOC 1
11 JMP  x
```

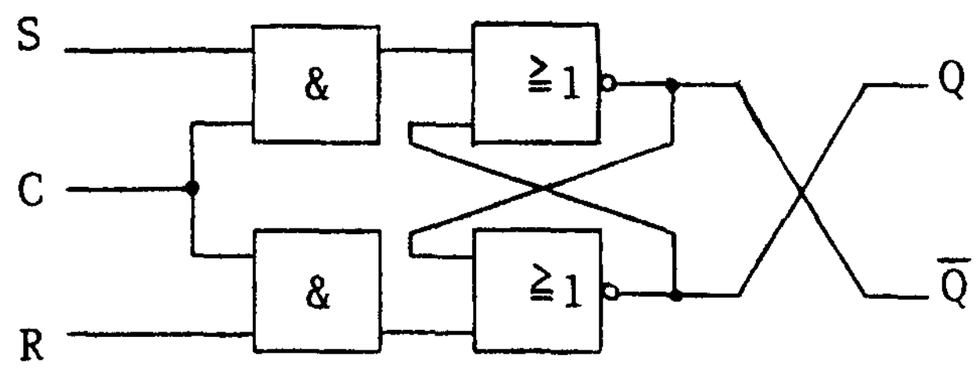
Durch LD 1 und STO 5 wird der Pegel des Eingangs S = E1 eingelesen und auf dem Ausgang A5 zwischengespeicher . Mit LDC 2 wird der invertierte Pegel des Eingangs R = E2 in das Ergebnisregister geladen. XNOR 5 vergleicht nun den Inhalt des Ergebnisregisters mit dem Ausgang A5. Falls R = "0" und S = "1" oder R = "1" und S = "0" ist, so ist das Ergebnis dieser logischen Verknüpfung eine "1". Ansonsten ist das Ergebnis eine "0". Das Ergebnis wird in das OEN-Register geladen. Im ersten Fall ist die Ausgabe freigegeben und mit LD 5, STO 0 und STOC 1 gelangt der zwischengespeicherte Wert von S auf Q = A0 und dessen Komplement auf $\overline{Q}$ = A1. Im zweiten Fall geschieht an den Ausgängen nichts, da die Ausgabe gesperrt ist.

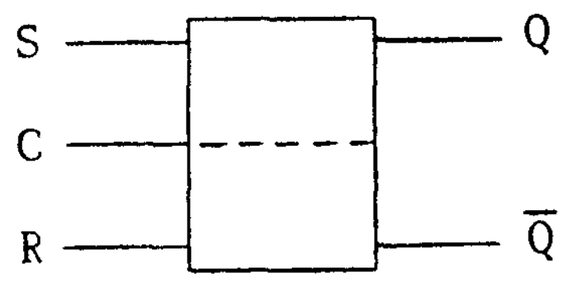
2. Taktzustandsgesteuertes RS-Flipflop

Schaltung (aufgebaut aus NAND-Gliedern):



Schaltung (aufgebaut aus NOR-Gliedern mit UND-Gliedern im Eingang)





Schaltzeichen nebenstehend!

Wahrheitstabelle:

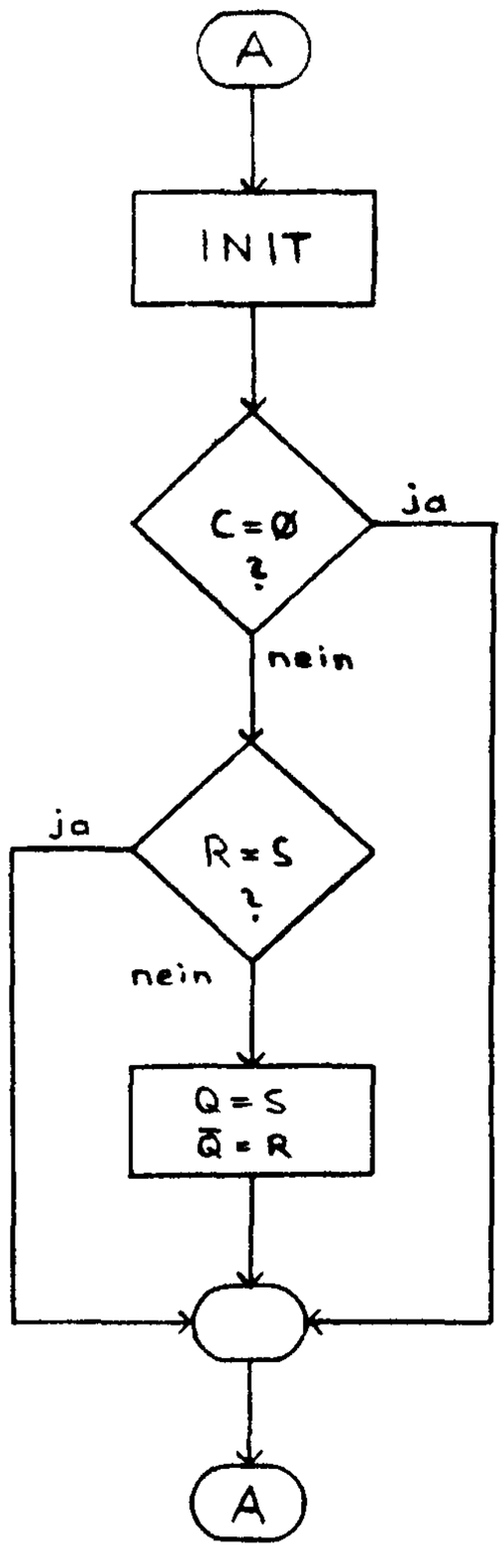
C	R	S	Q	$\overline{Q}$
0	0	0	Q-	$\overline{Q}$-
0	0	1	Q-	$\overline{Q}$-
0	1	0	Q-	$\overline{Q}$-
0	1	1	Q-	$\overline{Q}$-
1	0	0	Q-	$\overline{Q}$-
1	0	1	1	0
1	1	0	0	1
1	1	1	Q-	$\overline{Q}$-

Beim taktzustandsgesteuerten RS-Flipflop gibt es als dritten Eingang einen Takt (engl. Clock - C). Ist der Takt "0", so gilt der Speicherfall. Ist der Takt "1", so gilt die Wahrheitstabelle für das ungetaktete RS-Flipflop. Es gibt also auch hier einen nicht zulässigen Fall: C = R = S = 1. Auch hier wird im Programm dieser Fall durch den Speicherfall ersetzt, was in dér Wahrheitstabelle schon berücksichtigt wurde.

Flußdiagramm nebenstehend!

Festlegung der Zuordnungen im Programm:

C = E3, R = E2, S = E1, Q = A0, $\overline{Q}$ = A1



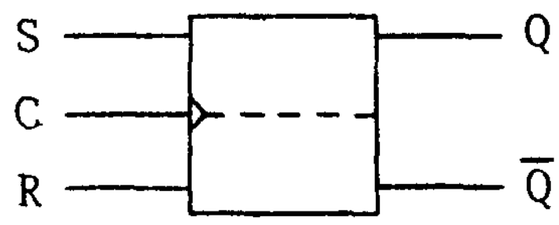
Programm:

```
00 INIT
03 LD   1
04 STO  5
05 LDC  2
06 XNOR 5
07 AND  3
08 OEN  0
09 LD   5
10 STO  0
11 STOC 1
12 JMP  x
```

3. Taktflankengesteuertes RS-Flipflop

Beim taktzustandsgesteuerten RS-Flipflop wurde beim Takt zwischen "0" und "1" unterschieden. Diese Unterscheidung wird beim taktflankengesteuerten RS-Flipflop nicht mehr gemacht. Hier sind die positive Taktflanke, d.h., Übergang von "0" auf "1" (↑), und die negative Taktflanke, d.h., Übergang von "1" auf "0" (↓), von Bedeutung.

RS-Flipflop, das bei positiver Taktflanke schaltet:
Schaltzeichen:

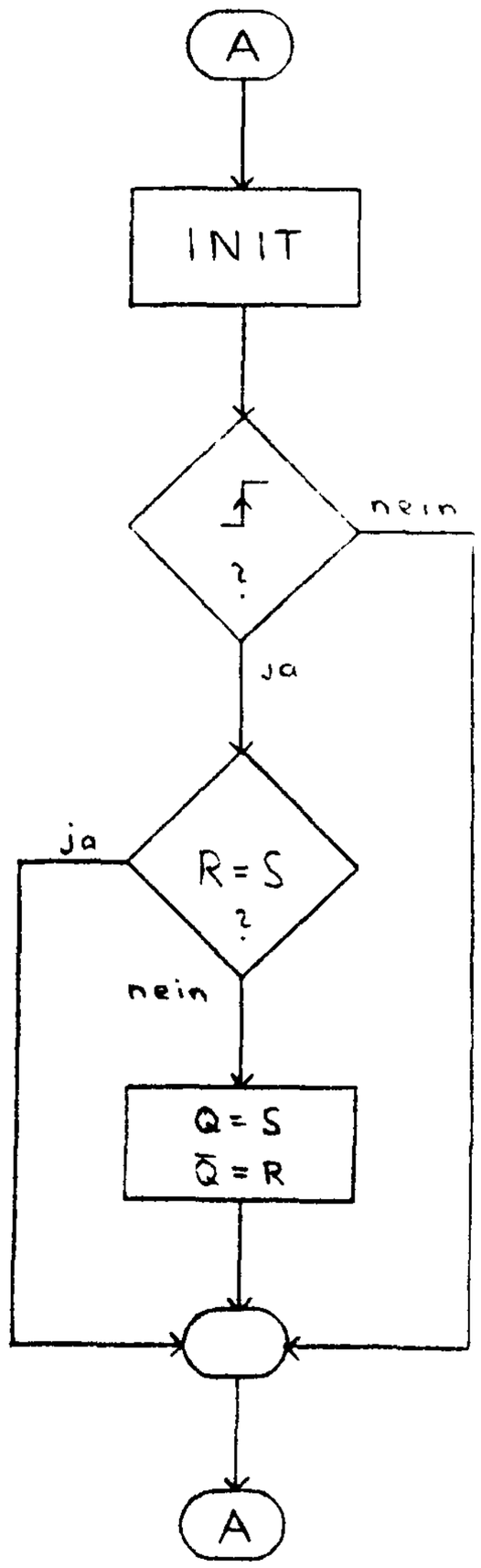


Wahrheitstabelle:
Bei der positiven Taktflanke gilt die Wahrheitstabelle für das ungetaktete RS-Flipflop. Ansonsten gilt der Speicherfall.

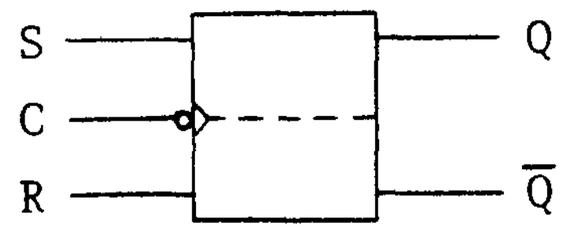
Flußdiagramm nebenstehend!
Festlegung der Zuordnungen im Programm:
C- = E5, C = E3, R = E2, S = E1, Q = A0, $\overline{Q}$ = A1

Programm:

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



RS-Flipflop, das bei negativer Taktflanke schaltet:
Schaltzeichen:



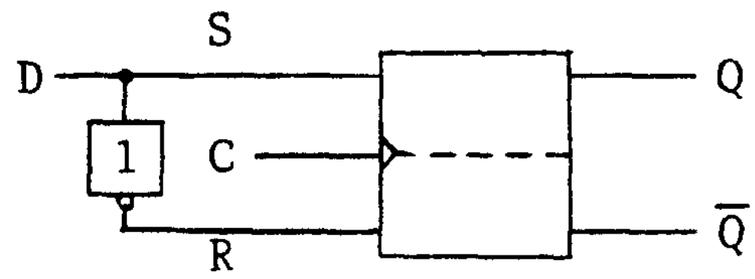
Wahrheitstabelle:
Bei der negativen Taktflanke gilt die Wahrheitstabelle für das ungetaktete RS-Flipflop. Ansonsten gilt der Speicherfall.
Gegenüber dem RS-Flipflop, das bei positiver Taktflanke schaltet, brauchen hier nur wenige Änderungen vorgenommen zu werden:
Flußdiagramm: Aus ⌐ wird ⌐̲ .

Programm:

```
07 ORC  6
08 STOC 6
```

4. Taktflankengesteuertes D-Flipflop

Bildung eines D-Flipflops aus einem RS-Flipflop:



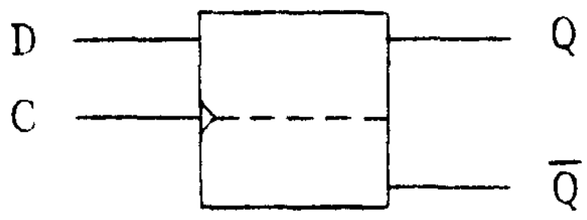
Der Zustand des Eingangs D wird mit der positiven oder mit der negativen Taktflanke des Eingangs C eingelesen und auf den Ausgang Q gesetzt.

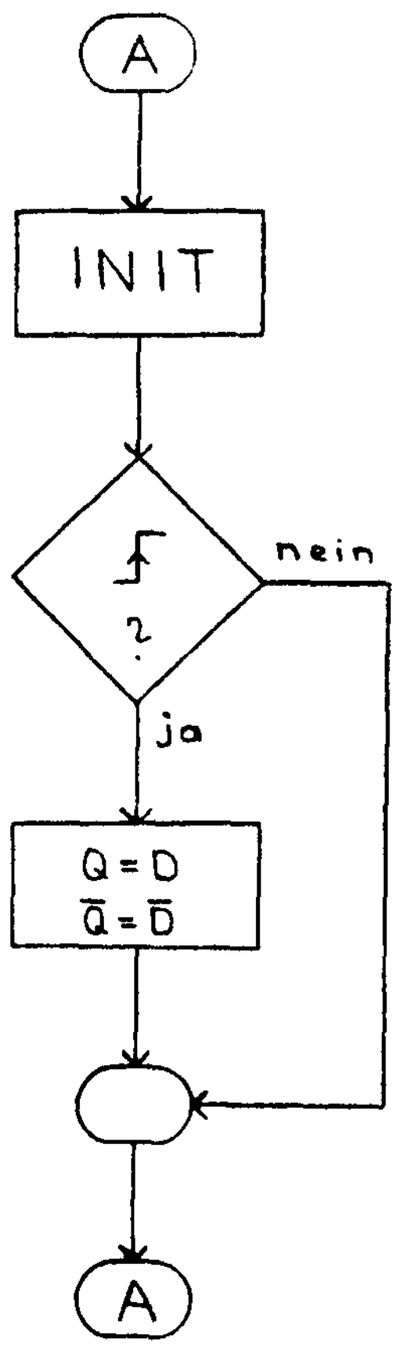
Wahrheitstabelle:

D	Q	$\overline{Q}$
0	0	1
1	1	0

D-Flipflop, das bei positiver Taktflanke schaltet:

Schaltzeichen:





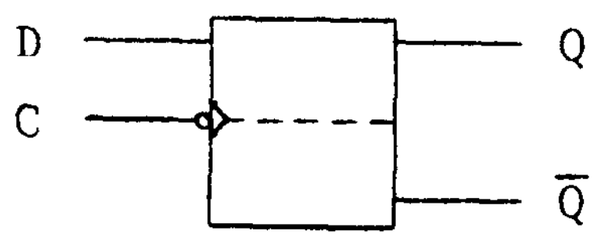
Flußdiagramm nebenstehend!
Festlegung der Zuordnungen im Programm:
C- = E5, C = E3, D = E1, Q = A0, $\overline{Q}$ = A1

Programm:

```
00 INIT
03 LD   5      C (alt) wird umgeladen
04 STO  6      und zwischengespeichert.
05 LD   3      C (neu) wird geladen
06 STO  5      und zwischengespeichert.
07 ANDC 6      Feststellen, ob der Takt von "0" nach
                "1" geht,
08 OEN  0      nur wenn dies der Fall ist, werden die
                nachfolgenden Ausgabebefehle durch-
                geführt.
09 LD   1      D wird geladen.
10 STO  0      Der Wert von D wird in Q ausgegeben.
11 STOC 1      Der invertierte Wert von Q wird in Q̄
                ausgegeben.
12 JMP  x
```

D-Flipflop, das bei negativer Taktflanke schaltet:

Schaltzeichen:



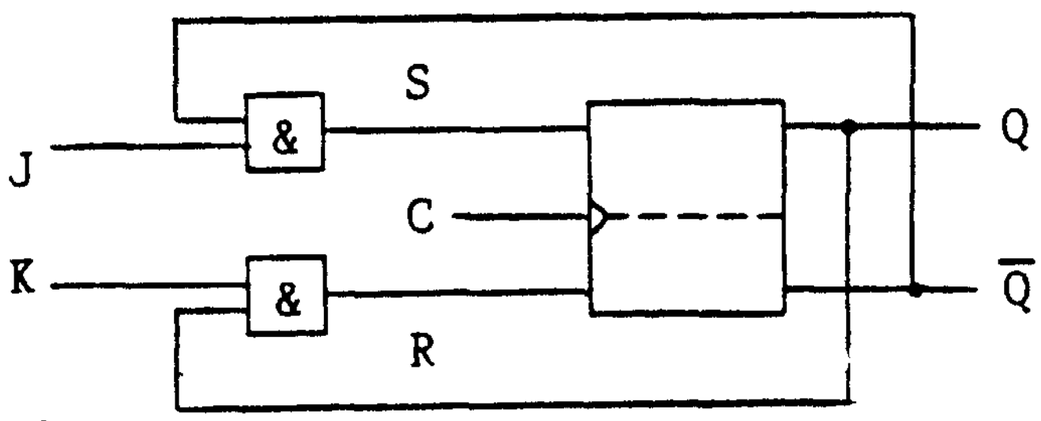
Änderungen gegenüber dem D-Flipflop, das bei positiver Taktflanke schaltet:
Flußdiagramm: Aus ⊓ wird ⊓̲.

```
Programm        07 ORC  6
                08 LDC  0
                09 OEN  0
                    .
                    .
                    .
```

Die restlichen Programmschritte rutschen gegenüber dem vorherigen Programm um eine Programmadresse weiter.

5. Taktflankengesteuertes JK-Flipflop

Bildung eines JK-Flipflops aus einem RS-Flipflop:



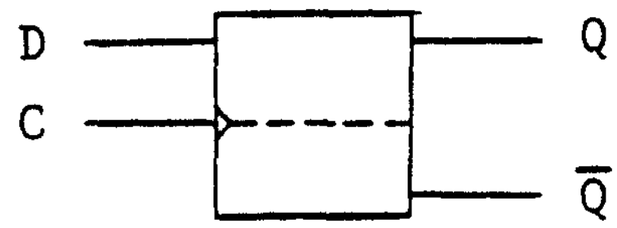
Wahrheitstabelle:

J	K	Q	$\overline{Q}$
0	0	Q-	$\overline{Q}$-
0	1	0	1
1	0	1	0
1	1	$\overline{Q}$-	Q-

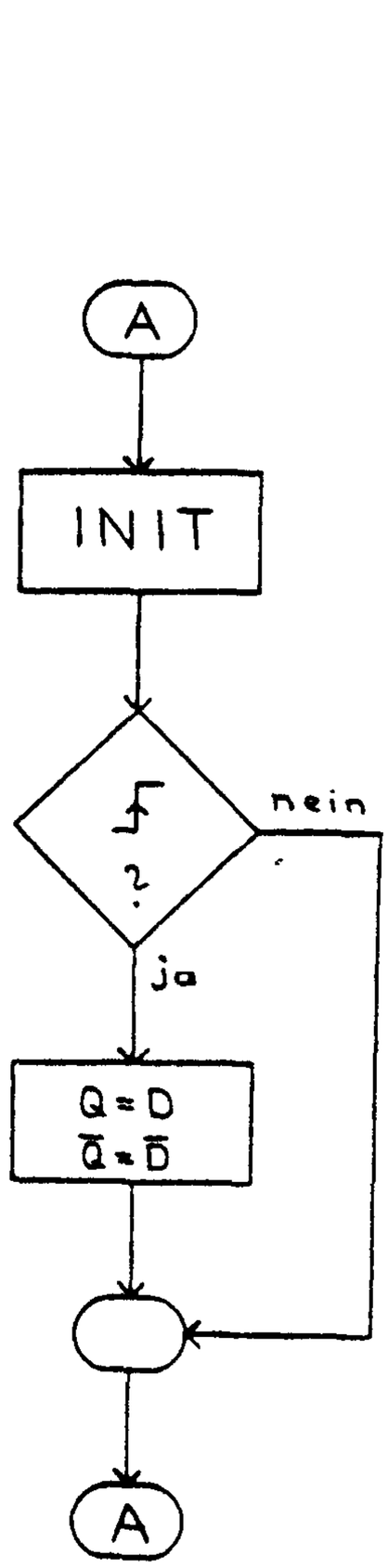
Das Verhalten eines JK-Flipflops läßt sich am besten durch eine logische Gleichung ausdrücken:
$Q = J*\overline{Q}- + K*\overline{Q}-$

JK-Flipflop, das bei positiver Taktflanke schaltet:

Schaltzeichen:



Flußdiagramm nebenstehend!
Festlegung der Zuordnungen im Programm:
Q- = E7, C- = E5, C = E3, K = E2, J = E1, Q = A0, $\overline{Q}$ = A1



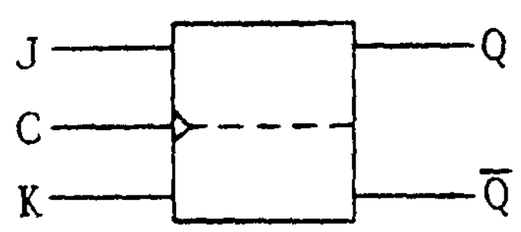
Programm:

```
00 INIT
03 LD   5
04 STO  6
05 LD   3
06 STO  5
```

```
07 ANDC 6
08 OEN  0
09 LD   1       J wird geladen
10 ANDC 7       und mit Q (alt) AND-verknüpft
11 STO  6       Das Ergebnis der Operation wird
                zwischengespeichert.
12 LDC  2       Das invertierte Signal von K wird
                geladen
13 AND  7       und mit Q (alt) AND-verknüpft.
14 OR   6       Das Ergebnis der letzten Operation
                wird mit dem zwischengespeicherten
                Ergebnis OR-verknüpft. Das Ergebnis
                ist Q.
15 STO  7       Q wird neu zwischengespeichert.
16 STO  0       Q wird auf Ausgang A0 ausgegeben.
17 STOC 1       Q̄ wird auf Ausgang A1 ausgegeben.
18 JMP  x
```

JK-Flipflop, das bei negativer Taktflanke schaltet:

Schaltzeichen:



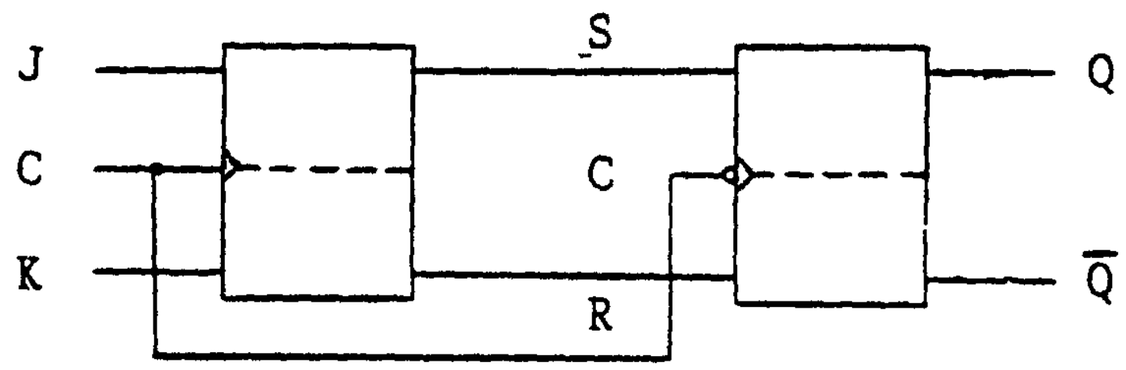
Änderungen gegenuber dem JK-Flipflop, das bei positiver Taktflanke schaltet:
Flußdiagramm: Aus ⌐ wird ⌙ .

```
Programm:       07 ORC  6
                08 LDC  0
                09 OEN  0
```

Die restlichen Programmschritte rutschen gegenüber dem vorherigen Programm um eine Programmadresse weiter.

6. JK-Master-Slave-Flipflop

Bildung eines JK-Master-Slave-Flipflops aus einem JK-Flipflop und einem RS-Flipflop:

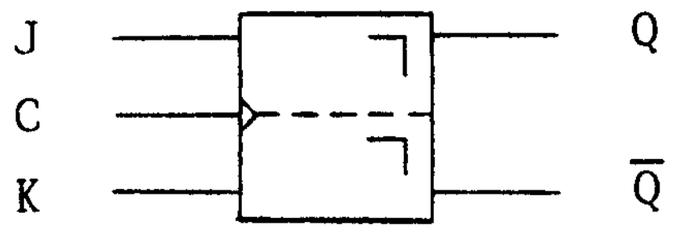


JK-Master-Slave-Flipflops nehmen bei der positiven Taktflanke das Eingangssignal auf. Dieses wird zwischengespeichert und erst bei der negativen Taktflanke zum Ausgang durchgeschaltet. Selbstverständlich gibt es auch JK-Master-Slave-Flipflops, die mit der negativen Taktflanke das erste Flipflop (Master-Flipflop) schalten. Dann schaltet die positive Taktflanke das zweite Flipflop (Slave-Flipflop).

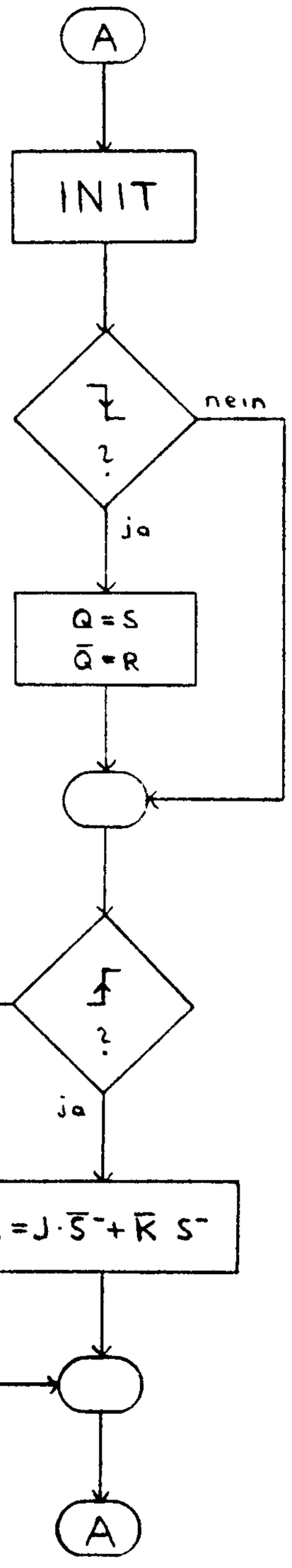
Wahrheitstabelle:

Es gilt die Wahrheitstabelle für das einfache taktflankengesteuerte JK-Flipflop.
JK-Master-Slave-Flipflop, das bei positiver Taktflanke das Master-Flipflop schaltet:

Schaltzeichen:



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