

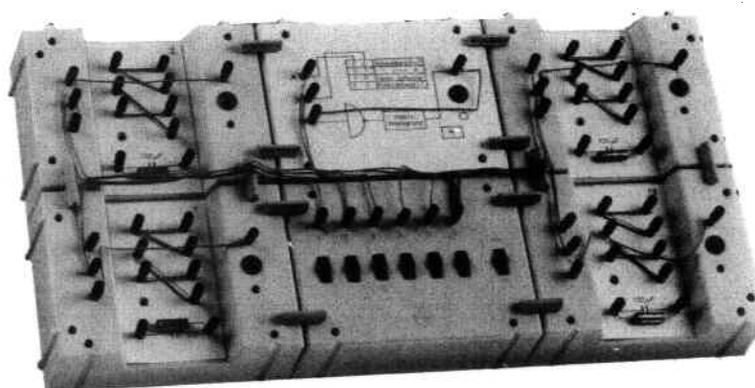
PHILIPS



Computer

Lehrbaukasten

CL 1601



2011

A n l e i t u n g s b u c h

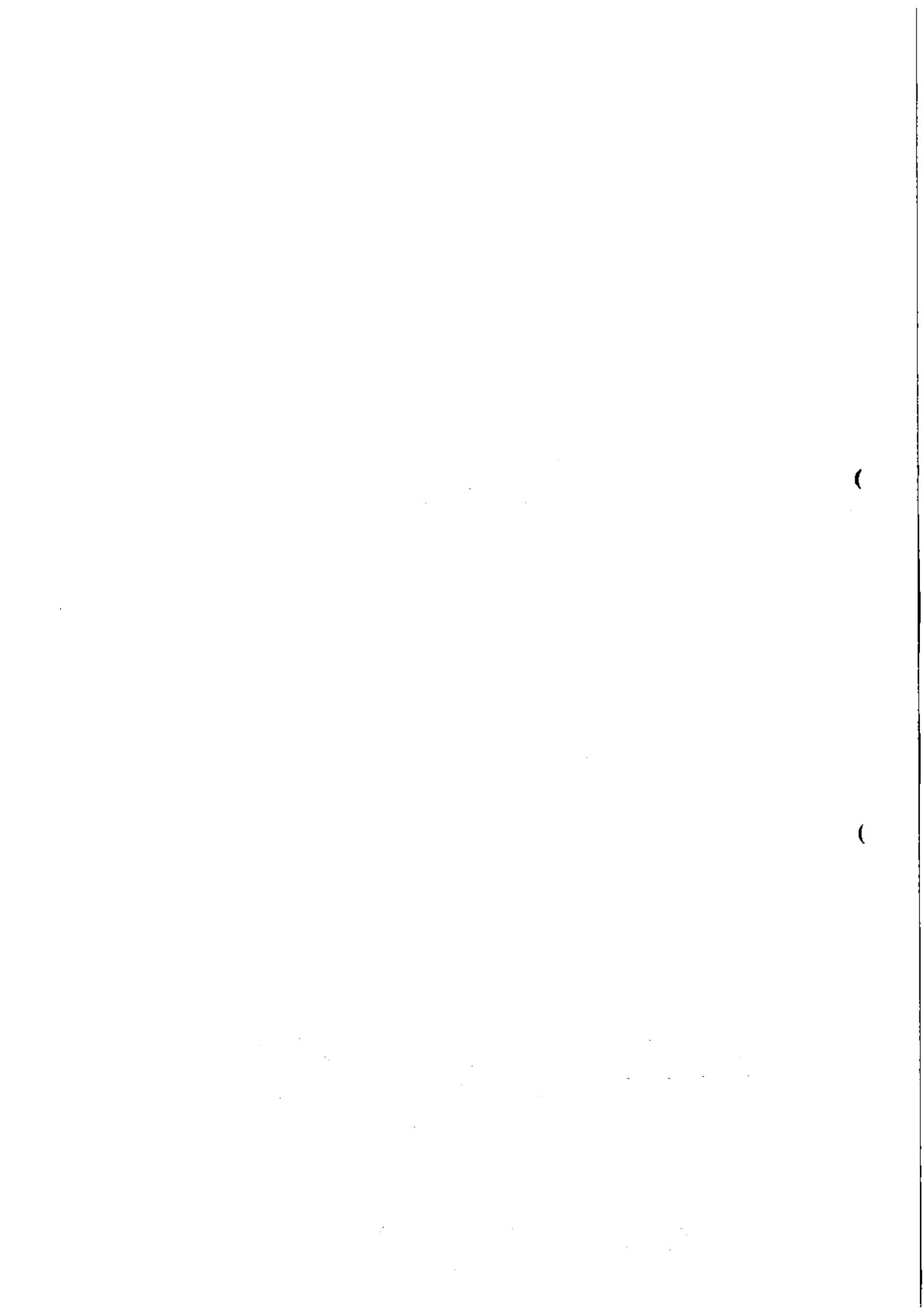
Computer-Lehrbaukasten
CL 1601

Alle Rechte vorbehalten. Nachdruck und fotomechanische Wiedergabe - auch auszugsweise - nicht gestattet. Wir übernehmen keine Gewähr, daß die in diesem Buch enthaltenen Angaben frei von Schutzrechten sind.

Technische Änderungen vorbehalten.

© Herausgegeben von der Deutschen Philips GmbH
Abt. Technische Spielwaren, 2 Hamburg 1, Mönckebergstraße 7

1972



V O R W O R T

Computer in der Welt von heute -

und der PHILIPS Computer-Lehrbaukasten CL 1601

Seit Jahrtausenden ist der Mensch bestrebt, sich körperliche Arbeit durch Technik zu erleichtern. Der große Durchbruch geschah zu Beginn des vorigen Jahrhunderts: Die industrielle Revolution - möglich durch die Erfindung der Dampfmaschine einige Jahrzehnte zuvor. Heute erleben wir die zweite industrielle Revolution: Computer erweitern unsere geistige Leistungsfähigkeit in einem Ausmaß, das man vor gut 30 Jahren, als der erste programmgesteuerte Computer der Welt seine Arbeit aufnahm, noch für utopische Phantasterei gehalten hätte.

Computer führen in wenigen Sekunden wissenschaftliche Berechnungen durch, für die ein ganzes Team von Wissenschaftlern Monate oder Jahre brauchen würde -

Computer steuern und regeln vollautomatisch komplizierte Fabrikationsprozesse -

Computer untersuchen und überwachen Patienten -

Computer geben Unterricht -

Computer berechnen Satellitenbahnen -

Computer steuern Raketen -

Computer regeln Atomreaktoren -

Computer konstruieren Flugzeuge -

Auch im Alltag des Wirtschaftslebens spielen Computer von Tag zu Tag eine wichtigere Rolle. Von der Lohnabrechnung bis zur Produktionsplanung, von der Einwohnerregistrierung bis zur Steuerberechnung -

Computer rationalisieren die Arbeit und entlasten den Menschen.

Computer - Elektronengehirne - Elektronische Datenverarbeitungsanlagen - oder wie auch immer diese Wunderwerke von Präzision und Schnelligkeit genannt werden, sind den meisten Menschen fremd, unverständlich, ja sogar unheimlich.

Dieser Computer-Lehrbaukasten soll Ihnen helfen, in diese komplizierte Materie einzudringen.

I n h a l t s v e r z e i c h n i s

1. Allgemeine Bauanleitung
 - 1.1. Eingabeeinheit
 - 1.2. Logik-Baustein
 - 1.3. Elektrolytkondensator
 - 1.4. Stromversorgungsstecker
 - 1.5. Stecker und Kontaktbügel
 - 1.6. Verdrahtungspläne
 - 1.7. Papiersablonen
 - 1.8. Kontrolle des Logik-Bausteins

2. Begriffserklärungen

3. Logische Grundfunktionen
 - 3.1. Darf ich nach der Party Auto fahren - Eine UND-Funktion
 - 3.2. Internationales Ferienlager
 - 3.3. Wann ist es hell im Zimmer - Eine ODER-Funktion
 - 3.4. Wann muß ich an einer Ampel halten
 - 3.5. Wie komme ich nach München - Eine Exklusiv-ODER-Funktion
 - 3.6. Sommerurlaub
 - 3.7. Wann ist die Waage im Gleichgewicht - Eine Äquivalenz-Funktion
 - 3.8. Bleibt die Erde trocken - Identität und Negation
 - 3.9. Welches ist der richtige Weg
 - 3.10. Eine Badewanne wird gefüllt
 - 3.11. Ein Lagerfeuer wird entzündet

4. Aufbau eines Computers
 - 4.1. Ist der Mensch ein Computer
 - 4.2. Ein großer Computer wird vorgestellt
 - 4.3. Das Programm
 - 4.4. Der Philips Computer-Lehrbaukasten CL 1601

5. Logische Funktionen mit zwei Eingängen
 - 5.1. Das Mantel-Problem

6. Logische Funktionen mit 3 Eingängen
 - 6.1. UND-Funktion
 - 6.2. Einschließende ODER-Funktion
 - 6.3. Exklusiv-ODER-Funktion
 - 6.4. Äquivalenz-Funktion
 - 6.5. NAND-Funktion
 - 6.6. NOR-Funktion

7. Die Programmierung der Logik-Bausteine

8. Der Aufbau der Logik-Bausteine

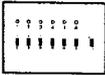
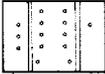
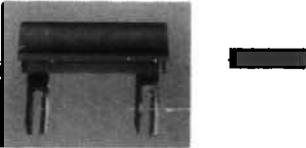
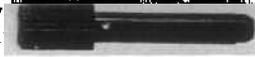
9. Blinkschaltungen

10. Demonstration der Baustein-Laufzeiten

11. Flipflops
 - 11.1. Speicher-Flipflop
 - 11.2. Auffang-Flipflop
 - 11.3. Binärzähler

12. Das Dualsystem
 - 12.1. Einführung
 - 12.2. Zahlen raten
 - 12.3. Rechnen mit Dualzahlen
 - 12.4. Addition
 - 12.5. Subtraktion
 - 12.6. Sequentielle Addier- und Subtrahierschaltung

13. Anwendungsmöglichkeiten des CL 1601
 - 13.1. Ein Party-Problem
 - 13.2. Wettervorhersage
 - 13.3. Kreditprüfung
 - 13.4. Platzreservierung
 - 13.5. Ein Tresor wird geknackt
 - 13.6. Knobeln
 - 13.7. Würfelspiel
 - 13.8. Bauer, Ziege, Wolf, Kohlkopf
 - 13.9. Reaktionstest
 - 13.10. Mensch gegen Computer

Teil und Symbol	Nr.	Bezeichnung	Inhalt CL 1601
	349.6502	Eingabeeinheit	1
	349.6501	Logik-Baustein	5
	349.1006	Elektrolyt-Kondensator 220 μF	1
	349.6019	Stromversorgungs- stecker	12
	349.6020	Stecker	60
	349.6021	Kontaktbügel für Stecker	60
	349.1017	Isolierter Draht	5 m

1. Allgemeine Bauanleitung

1.1. Eingabeeinheit

Die Eingabeeinheit nach Abbildung 1 übernimmt zwei wichtige Funktionen:
Alle Logik-Bausteine werden durch das eingebaute Stromversorgungs-
teil mit der notwendigen Betriebsspannung versorgt.

Über 6 Schalter lassen sich bei entsprechender Verdrahtung
Informationen auf die Eingänge der Logik-Bausteine geben.

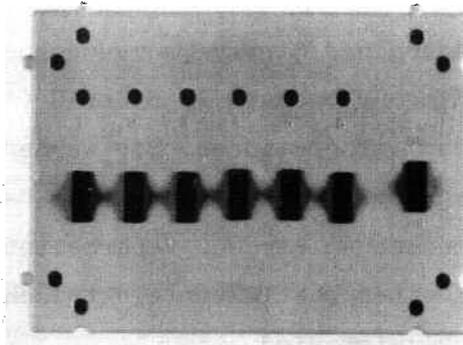


Abb. 1

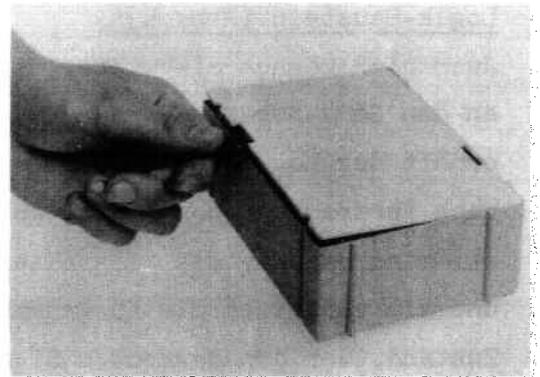


Abb. 2

Damit die Eingabeeinheit diese beiden Aufgaben erfüllen kann, müssen Sie
zunächst den Batteriehalter im Gehäuseinneren mit 4 Stück 1,5 Volt Trok-
kenbatterien (z.B. Philips Babyzellen, Typ R 14 TR) versehen. Entfernen
Sie deshalb zunächst mit einer Münze die Bodenplatte vom Gehäuse (Abb.2)
und setzen Sie dann gemäß Abb. 3 die Batterien in den Halter, wobei Sie
unbedingt auf die im Halter angegebene Polung achten müssen. Danach ver-
schließen Sie die Eingabeeinheit wieder mit der Bodenplatte.

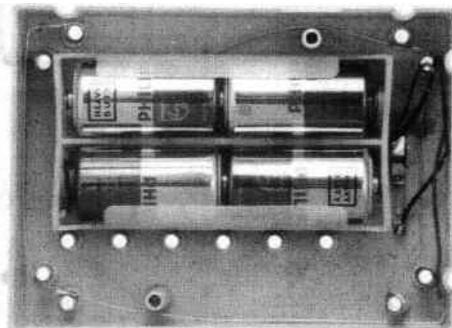


Abb. 3

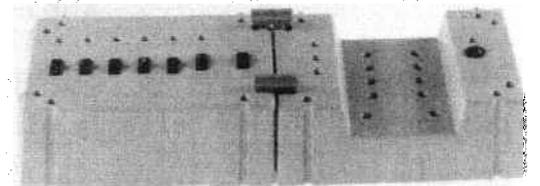


Abb. 4

Auf der Oberseite befindet sich ganz rechts der EIN-AUS-Schalter. In der
EIN-Stellung verbindet er den Batteriehalter mit den Stromversorgungs-
buchsen an der Gehäusekante. Bei Verwendung der Stromversorgungsstecker

als Verbindungselemente zwischen Eingabeeinheit und den Logik-Bausteinen ist durch die spezielle Anordnung der Buchsen, Nuten und Federn (Abb. 4) eine falsche Polung der Versorgungsspannung ausgeschlossen.

Neben dem EIN-AUS-Schalter befinden sich 6 weitere schwarze Schieber, die Eingabeschalter genannt werden. Sie sind den mit 1 bis 6 bezifferten Buchsen zugeordnet. Je nach Schalterstellung können über Drahtleitungen elektrische Signale (Informationen) in die Logik-Bausteine eingegeben werden.

1.2. Logik-Baustein (Abb. 5)

Auch hier können Sie wieder die Nuten, Federn und Stromversorgungsbuchsen an den Gehäusekanten erkennen. Über die Eingangsbuchsen A, B und C (Abb.6) erhält der Logik-Baustein alle elektrischen Informationen. Sie werden je nach Verdrahtung des tiefer gelegenen Programmierfeldes nach einem bestimmten Schema, das Sie später noch kennenlernen werden, verarbeitet, und sie versetzen die Ausgangsbuchse F in einen bestimmten elektrischen Zustand, der von der roten Signallampe angezeigt wird.

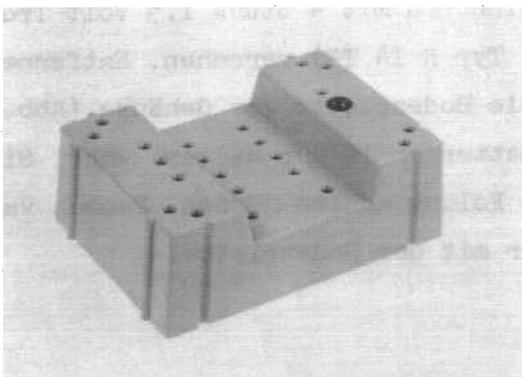


Abb. 5

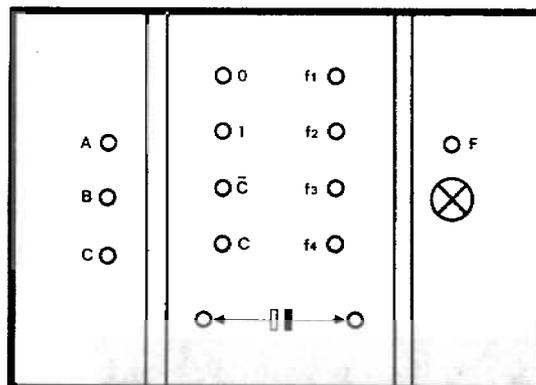


Abb. 6

Die linken 4 Buchsen im Programmierfeld (sie werden mit 0, 1, \bar{C} und C bezeichnet) können beliebig mit den gegenüberliegenden 4 rechten Programmieringängen f1, f2, f3 und f4 verbunden werden.

Bei einigen Anwendungsbeispielen ist es notwendig, die auf die Eingänge A, B und C gegebene Information gegenüber dem Ausgangssignal F zeitlich zu verzögern. Das kann durch einen Elektrolytkondensator erreicht werden, der in diesem Fall an die beiden unteren Buchsen im Programmierfeld anzuschließen ist. Bitte, achten Sie auf die in allen Schaltbildern und an allen Logik-Bausteinen angegebene Polung.

1.3. Elektrolytkondensator (Abb. 7)

Mit diesem Bauelement läßt sich eine Verzögerung vom Eingangs- zum Ausgangssignal erreichen. Allerdings muß der Kondensator in der richtigen Polarität an die betreffenden Buchsen im Programmierfeld des Logik-Bausteins angeschlossen werden (Abb. 8). Der Pluspol ist auf dem Kondensator deutlich mit dem Zeichen + versehen. Dieses Zeichen finden Sie in allen Schaltbildern und im Programmierfeld der Logik-Bausteine wieder.



Abb. 7



Abb. 9

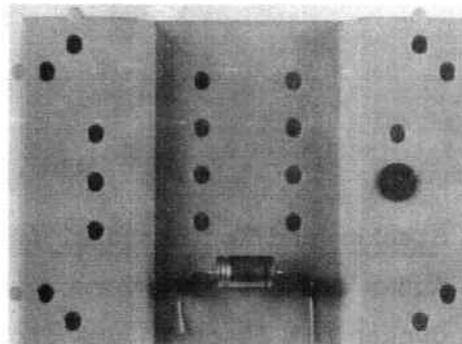


Abb. 8

1.4. Stromversorgungsstecker (Abb. 9)

Sie dienen als Verbindungselement zwischen dem Stromversorgungsteil der Eingabeeinheit und den Stromversorgungsbuchsen der Logik-Bausteine. Schieben Sie Nut und Feder zweier Bausteine wie in Abb. 10 angegeben zusammen, und drücken Sie nach Abb. 11 die Stromversorgungsstecker in die Buchsen. Falls ein Stecker zu leicht in die Stromversorgungsbuchsen gleitet, müssen Sie die Kontakte etwas aufbiegen. In den Verdrahtungsplänen sind die benötigten Stromversorgungsstecker durch einen breiten schwarzen Balken angedeutet. Schließen Sie nur die für das Spiel benötigte Anzahl von Logik-Bausteinen an die Eingabeeinheit an (Abb. 12).

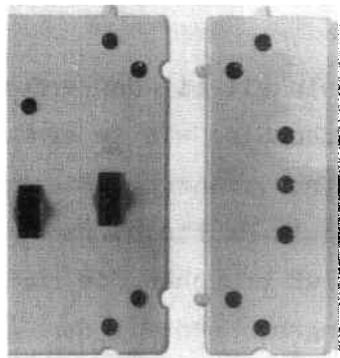


Abb. 10

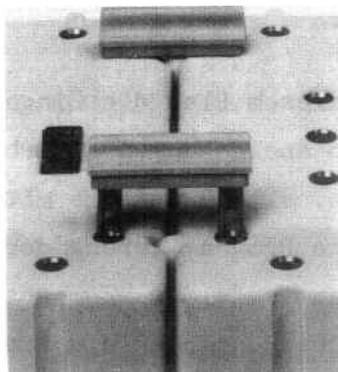


Abb. 11

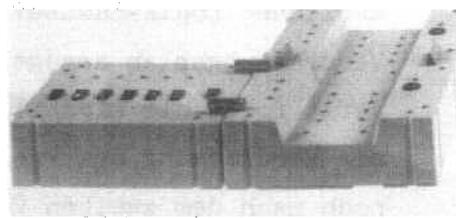


Abb. 12

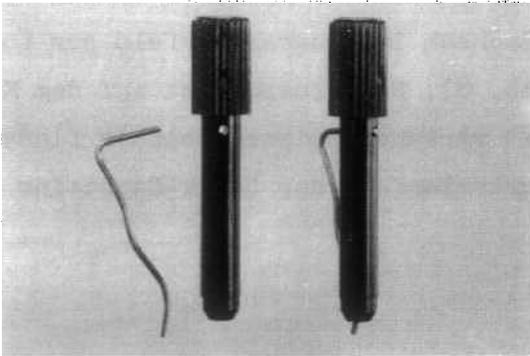


Abb.13

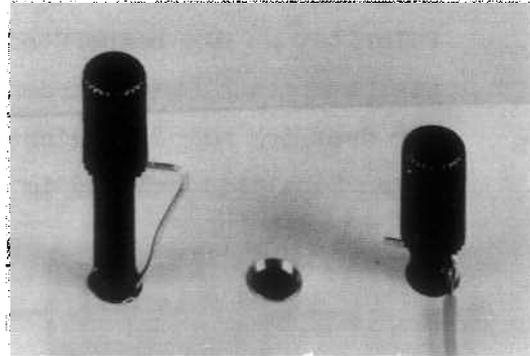


Abb.14

1.5. Stecker und Kontaktbügel (Abb. 13)

Außer bei der Stromversorgung werden alle anderen elektrischen Verbindungen mit Steckern und isolierten Schaltdrähten vorgenommen.

Stecken Sie das untere Ende des Kontaktbügels in den Schlitz des Kunststoffsteckers, und schieben Sie es durch das Loch an der Stirnseite des Steckerschaftes. Das abgewinkelte Drahtende können Sie nun durch das Querloch drücken. Mit diesen Handgriffen haben Sie den Stecker montiert und können ihn gem. Abb. 14 in die zu verbindenden Buchsen einsetzen. Die auf richtige Länge abgeschnittenen und an den Enden abisolierten Drähte können nun zwischen Kontaktbügel und Kunststoffteil gesteckt und eingeklemmt werden.

1.6. Verdrahtungspläne

Für alle Anwendungsbeispiele finden Sie im Anleitungsbuch einen übersichtlichen Verdrahtungsplan, der Ihnen in zwei Abbildungen alle Verbindungsleitungen des Programmierfeldes und die Verdrahtung der Ein- und Ausgangsbuchsen zeigt.

Reihen Sie zunächst nach diesem Plan die Eingabeeinheit mit den erforderlichen Logik-Bausteinen aneinander und verbinden Sie, wie im ersten Verdrahtungsplan angegeben, alle Bausteine mit Stromversorgungssteckern. Jetzt folgt die eigentliche Programmierung der Logik-Bausteine durch die Verdrahtung des Programmierfeldes. Zum Abschluß brauchen Sie nur noch nach dem zweiten Verdrahtungsplan die Eingangsbuchsen mit den Eingabeschaltern bzw. mit den Ausgangsbuchsen F zu verbinden, und der gesamte Aufbau ist einsatzbereit.

1.7. Papierschablonen

Für die wichtigsten logischen Grundfunktionen finden Sie am Schluß des Anleitungsbuches vorgedruckte Schablonen. Mit ihrer Hilfe lassen sich häufig wiederkehrende Schaltungen schneller verdrahten. Da Ein- und Ausgänge sowie das Programmierfeld getrennt verdrahtet werden, sind zwei Schablonen nötig. Schneiden Sie beide Teile aus dem Papierbogen und legen Sie die kleine Schablone auf das Programmierfeld. Mit einem spitzen Bleistift lassen sich jetzt alle gekennzeichneten Stellen leicht durchstoßen. Durch die entstandenen Löcher passen die erforderlichen vormontierten Stecker, und, wie auf der Schablone abgebildet, läßt sich die Verdrahtung ausführen.

Die große Schablone wird in gleicher Weise vorbereitet; sie erhält jedoch zusätzlich ein großes Loch für die rote Anzeigelampe.

1.8. Kontrolle des Logik-Bausteins

Verbinden Sie die Eingabeeinheit und einen Logik-Baustein und stellen Sie die Verdrahtung gem. Abb. 15 her. Achten Sie auf die Polarität des Elektrolytkondensators und kontrollieren Sie alle Verbindungen. Wenn Sie jetzt den EIN-AUS-Schalter der Eingabeeinheit in die Stellung EIN bringen, muß bei einem einwandfreien Logik-Baustein ständig die rote Lampe blinken.

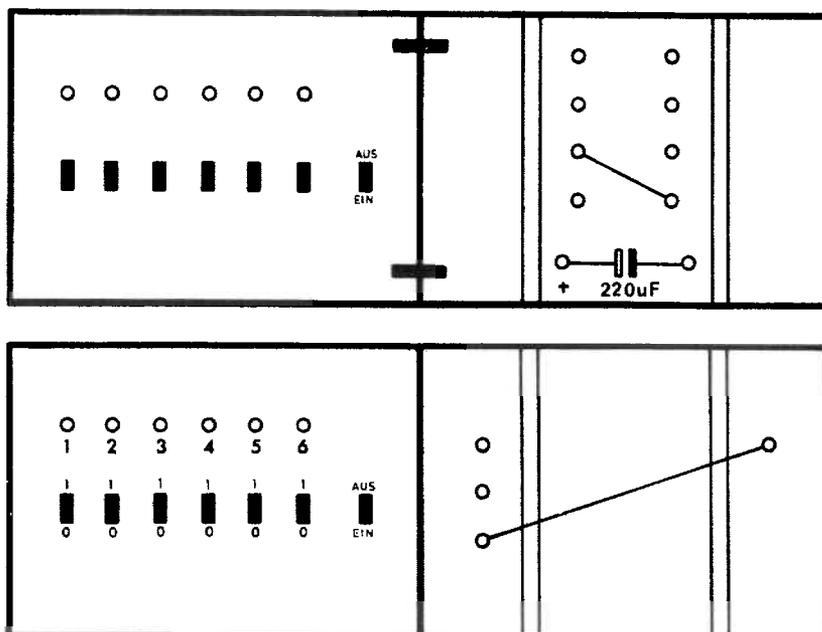


Abb.15

2. Begriffserklärungen

Vor dem ersten praktischen Einsatz des CL 1601 zunächst einige Begriffserklärungen.

Das Wort C o m p u t e r kommt aus dem Englischen. Zu deutsch heißt Computer R e c h n e r. Genaugenommen wird diese Bezeichnung dem Computer nicht gerecht, denn er kann mehr als nur rechnen. Er kann Daten verarbeiten. Und weil er das elektronisch macht, sagt man auch Elektronische Datenverarbeitungsanlage, abgekürzt EDVA. Dementsprechend wird Elektronische Datenverarbeitung kurz EDV genannt. Aber was ist Datenverarbeitung? Und was sind Daten? - Daten sind Informationen. Und Information ist alles das, was uns in irgendeiner Form eine Bedeutung übermittelt, also z.B. gesprochene oder geschriebene Wörter und Sätze, ferner Zahlen, Zeichnungen und Bilder. Einem Computer werden Daten zur Verarbeitung in Form von Zahlen bzw. Ziffern, Buchstaben und Zeichen (wie . , / + % usw.) eingegeben. Nach der Verarbeitung der Daten werden die Ergebnisse wiederum meist in Ziffern, Buchstaben und Zeichen ausgegeben, die zusammengenommen selbstverständlich irgendwelche sinnvollen Aussagen darstellen. Wie das bei großen Computern vor sich geht, werden Sie an anderer Stelle dieser Anleitung noch erfahren. Ähnlich wie bei großen Computern werden auch beim Philips Computer-Lehrbaukasten CL 1601 Daten eingegeben, verarbeitet und als Ergebnisse ausgegeben. Die Verarbeitung der Daten erfolgt aufgrund eines Programms. Auch darin stimmen große Computer und der CL 1601 überein. Veranschaulichen wir uns dieses Grundprinzip jeder Datenverarbeitung anhand einer schematischen Darstellung:



Der Ablauf ist leicht zu merken durch die Anfangsbuchstaben E V A.

Nun noch eine ganz besondere Kuriosität elektronischer Datenverarbeitung: Computer führen sämtliche Daten auf ganze 2 Zeichen zurück, auf die Zeichen 0 und 1. Wahrscheinlich können Sie sich im Moment nicht vorstellen, wie das möglich sein soll: Datenverarbeitung bis zu den kompliziertesten Aufgaben nur mit den beiden Zeichen 0 und 1? Aber es ist tatsächlich so, und Sie werden dieses Geheimnis bald ergründet haben - mit Ihrem CL 1601. Auch er arbeitet mit 0 und 1.

Vorerst einmal genug der Theorie. Jetzt soll Ihr Computer Daten verarbeiten.

3. Logische GrundfunktionenErster Einsatz des Philips Computer-Lehrbaukastens CL 1601

Bei den folgenden Versuchen können wir uns mit der Eingabeeinheit (mit Spannungsversorgung) und einem Logik-Baustein begnügen.

3.1. Darf ich nach der Party Auto fahren? - Was sagt der Computer dazu?Eine UND-Funktion

Sie sind auf einer feuchtfröhlichen Party. Ein Freund hat Sie mit seinem Wagen dorthin mitgenommen. Als zu vorgerückter Stunde die Heimfahrt beschlossen wird, ist Ihr Freund reichlich vollgetankt. Glücklicherweise ist er aber auch vernünftig: Er will sich mit Rücksicht auf das Maß seiner Alkoholisierung nicht ans Steuer setzen. Er fragt Sie, ob Sie in der Lage sind, bei der Heimfahrt das Steuer zu übernehmen. Daraufhin fragen Sie sich: 1. Bin ich nüchtern (Alkohol im Blut mit Sicherheit unter 0,8 ‰)? 2. Habe ich meinen Führerschein überhaupt dabei? Sehen wir mal, was Ihr Computer dazu sagt. Wann sagt er Ihnen, Sie dürfen Auto fahren bzw. Sie dürfen nicht Auto fahren? Gewiß trauen wir Ihnen zu, die richtige Antwort auch ohne Hilfe des Computers zu finden. Aber darum geht es ja schließlich nicht.

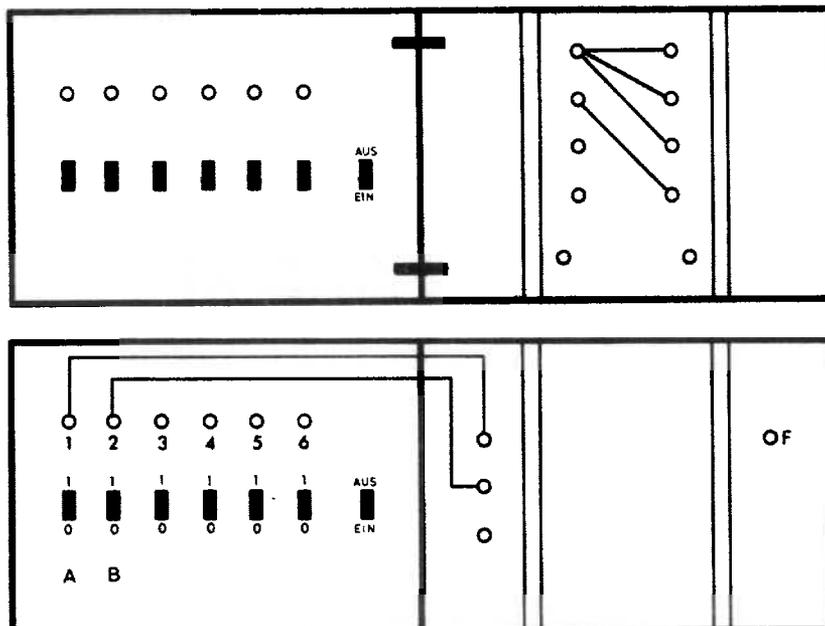


Abb.16

Verbinden Sie einen Logik-Baustein mit der Eingabeeinheit und verbinden Sie die Buchsen 1 und 2 der Eingabeeinheit mit den Buchsen A und B des Computerbausteins. Dann programmieren Sie den Baustein, indem Sie die Buchse 0 des Programmierfeldes mit den Buchsen f1, f2 und f3 des Pro-

grammierfeldes und 1 mit f⁴ verbinden (siehe Allgemeine Bauanleitung). Zur Durchführung unserer Datenverarbeitungsaufgabe brauchen wir von den Eingabeschaltern die Schalter A und B; sie sollten zu Beginn der Aufgabe auf 0 stehen. Schalten Sie jetzt den EIN-AUS-Schalter ein.

Der Schalter 1 wird der Frage "Bin ich nüchtern?" zugeordnet, der Schalter 2 der Frage "Habe ich meinen Führerschein dabei?". Auf die beiden Fragen gibt es als Antwortmöglichkeiten "JA" bzw. "NEIN". Treffen wir eine weitere Zuordnung: Für NEIN sagen wir 0 (Null), für JA sagen wir 1.

Ähnliche Zuordnungen nehmen wir für die Antwort vor: Für die Frage "Darf ich Auto fahren?" sei das rote Licht zuständig. Müssen wir die Frage mit NEIN beantworten, brennt das Licht nicht, können wir die Frage mit JA beantworten, leuchtet das Licht auf.

Schalten Sie jetzt anhand der folgenden Tabelle einmal alle Möglichkeiten durch, führen Sie also alle Kombinationen der Dateneingabe aus, und schreiben Sie dabei das jeweilige Ergebnis (ebenfalls in Form von 0 bzw. 1) in die Spalte für Datenausgabe:

Dateneingabe		Datenausgabe
Bin ich nüchtern?	Habe ich meinen Führerschein dabei?	Darf ich Auto fahren?
A	B	F
0	0	
0	1	
1	0	
1	1	

In welchen Fällen leuchtet die Lampe nicht auf?

Wann leuchtet sie auf?

Haben Sie für JA = "Lampe an" eine 1 eingetragen und für NEIN = "Lampe aus" eine 0 ?

Sieht Ihr Ergebnis auch so aus?

Dateneingabe		Datenausgabe
Bin ich nüchtern?	Habe ich meinen Führerschein dabei?	Darf ich Auto fahren?
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

In Worten ausgedrückt lautet das Ergebnis unserer Computerbefragung:
Wenn ich nüchtern bin und wenn ich meinen Führerschein dabei habe, dann
 darf ich Auto fahren.

Wir haben es hier mit einer sogenannten UND-Funktion zu tun. UND-Funktion deshalb, weil eine Bedingung und eine zweite Bedingung erfüllt sein müssen, damit ein bestimmtes Ergebnis zustande kommt.

Die dazugehörige Tabelle, deren Ergebniswerte Sie mit Ihrem Computer ermittelt haben, heißt Funktionstabelle.

Wenn wir die eine Bedingung A nennen und die zweite Bedingung B und wenn wir das Ergebnis mit F (Lampe) bezeichnen, läßt sich die Funktionstabelle so darstellen:

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Bevor wir zum nächsten Experiment kommen, wollen wir noch einige weitere Zusammenhänge klären.

Sehen Sie sich das folgende Schaltbild an:

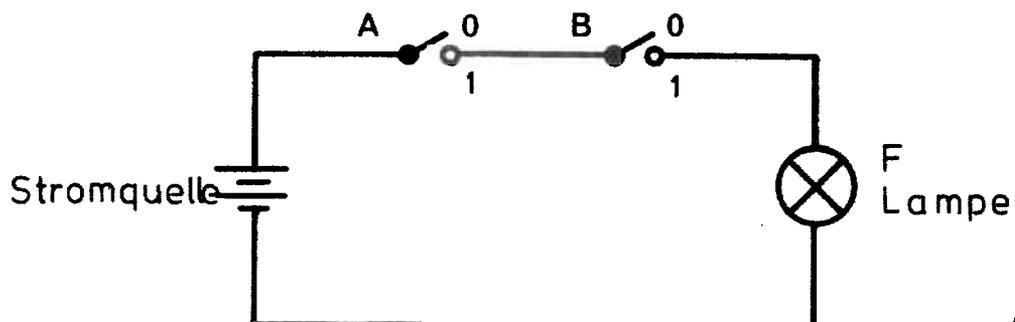


Abb.17

Eine Lampe ist mit einer Stromquelle verbunden. In den Stromkreis sind zwei Schalter eingebaut, und zwar hintereinander. Sie können leicht erkennen, daß die Lampe nur dann brennt, wenn Schalter A und Schalter B auf 1 - also eingeschaltet - sind.

Der Form nach haben wir es hier also ebenfalls mit einer UND-Funktion zu tun.

Damit haben wir eine UND-Funktion sprachlich ausgedrückt, in Form einer Funktionstabelle dargestellt und in einer elektromechanischen Schaltung veranschaulicht.

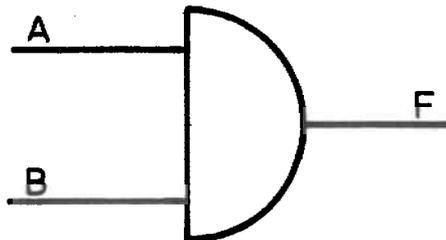
Eine entsprechende Schaltung haben Sie auch bei Ihrem Computer-Lehrbaukasten verwirklicht. Deshalb konnten Sie auch die Ergebnisse ermitteln, die mit den Zusammenhängen bei der elektromechanischen Schaltung übereinstimmen.

Aber: Bei dem Computer-Lehrbaukasten handelt es sich um eine elektronische Schaltung. Wie die Elektronik den großen Computern zu deren enormen Leistungen verhilft, so bietet die Elektronik im Philips Computer-Lehrbaukasten ganz andere Experimentiermöglichkeiten als ein "elektromechanischer Spielcomputer" sie bieten würde.

Kommen wir aber noch einmal auf die Gemeinsamkeit zwischen der oben gezeigten elektromechanischen Schaltung und der elektronischen Schaltung in Ihrem Lehrcomputer zurück: Beiden Schaltungen gemeinsam ist ihre Logik: Wenn A und wenn B, dann (und nur dann) F. So können wir den Sachverhalt abstrakt ausdrücken.

Die besprochene UND-Funktion ist eine logische Funktion. Außer der UND-Funktion gibt es natürlich noch andere logische Funktionen. Sie werden sie noch kennenlernen.

Wenn eine logische Funktion in Form einer Schaltung (einer mechanischen, einer elektromechanischen oder einer elektronischen Schaltung) verwirklicht ist, spricht man von einer logischen Schaltung. Für logische Schaltungen gibt es bestimmte Symbole. So sieht z.B. das Symbol für die besprochene UND-Schaltung so aus:



Wenn Sie jetzt einmal einen kurzen Blick auf den Schaltplan der Logik-Bausteine Ihres Computer-Lehrbaukastens werfen (Kap. 8), dann sehen sie dieses und ähnliche Symbole. Lassen Sie sich aber nicht dadurch verwirren, daß Sie den Schaltplan noch nicht durchschauen. Zunächst einmal wollen wir weitere logische Funktionen und Schaltungen kennenlernen.

3.2. Internationales Ferienlager

Durch ein weiteres Beispiel können Sie sich mit einer UND-Funktion und ihrer logischen Schaltung vertraut machen.

Schüler bewerben sich um die Teilnahme an einem internationalen Ferienlager. Voraussetzung für die Teilnahme ist, daß jeder mindestens englisch und französisch spricht. Da nicht alle Jugendlichen zwei Fremdsprachen beherrschen, sollen Sie mit Hilfe des Computers entscheiden, wer die Ferien dort verbringen darf.

Versuchen Sie einmal, die vier möglichen Kombinationen, die sich aus diesen beiden Sprachen ergeben, aufzustellen:

englisch	französisch	Ferienlager
A	B	F
nein	nein	nein
ja	ja	nein
		nein
		ja

Vielleicht hilft Ihnen beim Ausfüllen der Funktionstabelle die sprachliche Formulierung der UND-Funktion: Wenn der Schüler englisch und französisch spricht, dann darf er am Ferienlager teilnehmen.

Die Funktionstabelle läßt sich einfacher aufstellen, wenn für die Beherrschung der Sprachen (JA) = 1, andernfalls (NEIN) = 0 vereinbart wird. Die Zusage für die Teilnahme (JA) wird ebenfalls durch 1 ausgedrückt, die Ablehnung (NEIN) durch 0.

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Was Sie schon selbst bei der ausführlichen Form der Funktionstabelle herausgefunden haben, drückt auch die Kurzform aus: Nur im vierten Fall der möglichen Kombinationen, also wenn der Schüler englisch und französisch beherrscht, wird eine Zusage erteilt.

Schalten Sie jetzt das Beispiel mit Ihrem Computer durch. Verbinden Sie dazu die Buchse 1 der Eingabeeinheit mit der Buchse A eines Logik-Bausteins und 2 mit B. Der Logik-Baustein wird programmiert, indem Sie die Buchse 0 des Programmierfeldes mit f1, f2 und f3, die Buchse 1 mit f4 verbinden.

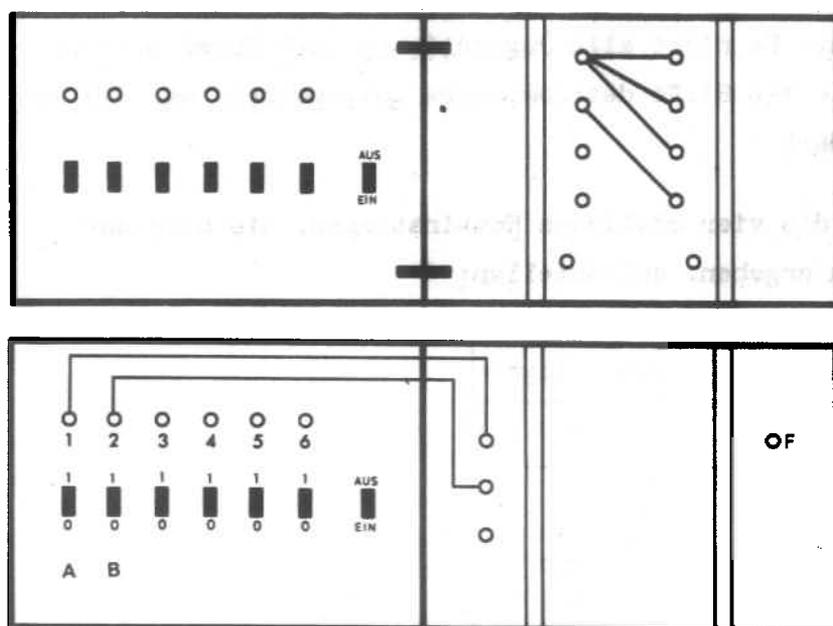


Abb.18

Schalter A	AUS	französisch	NEIN	(0)
Schalter A	EIN	französisch	JA	(1)
Schalter B	AUS	englisch	NEIN	(0)
Schalter B	EIN	englisch	JA	(1)
Lampe	AUS	nimmt nicht teil		(0)
Lampe	EIN	nimmt teil		(1)

Überqueren einer Straße

Nun ein letztes einfaches Beispiel zu einem logischen UND: Als Verkehrsteilnehmer ist Ihnen längst in Fleisch und Blut übergegangen, daß Sie eine Straße nur dann überqueren, wenn von links und von rechts kein Fahrzeug kommt. Überprüfen Sie doch einmal, ob Ihr Computer Ihnen denselben Rat gibt.

Bei diesem Beispiel sollen Sie aber weitgehend selbständig arbeiten, damit Sie prüfen können, ob Sie das Aufstellen der UND-Funktion beherrschen.

Einige Hilfen erhalten Sie allerdings noch:

Straße links frei	JA (1)
	NEIN (0)
Straße rechts frei	JA (1)
	NEIN (0)

Jetzt fällt es Ihnen bestimmt nicht schwer, die ausführliche Form der Funktionstabelle aufzustellen und die möglichen Kombinationen anzugeben.

Straße links frei	Straße rechts frei	Gehen
A	B	F

Zum Aufstellen der Kurzform der Funktionstabelle muß nur noch festgelegt werden, daß 1 GEHEN und 0 NICHT GEHEN zugeordnet werden soll.

A	B	F

Wenn Sie nach der folgenden Abbildung die Buchse 1 der Eingabeeinheit mit Buchse A und 2 mit B und im Programmierfeld des Logik-Bausteins 0 mit f1, f2, f3 und 1 mit f4 verbinden, können Sie mit Ihrem Computer überprüfen, ob Sie ihre Funktionstabelle richtig aufgestellt haben.

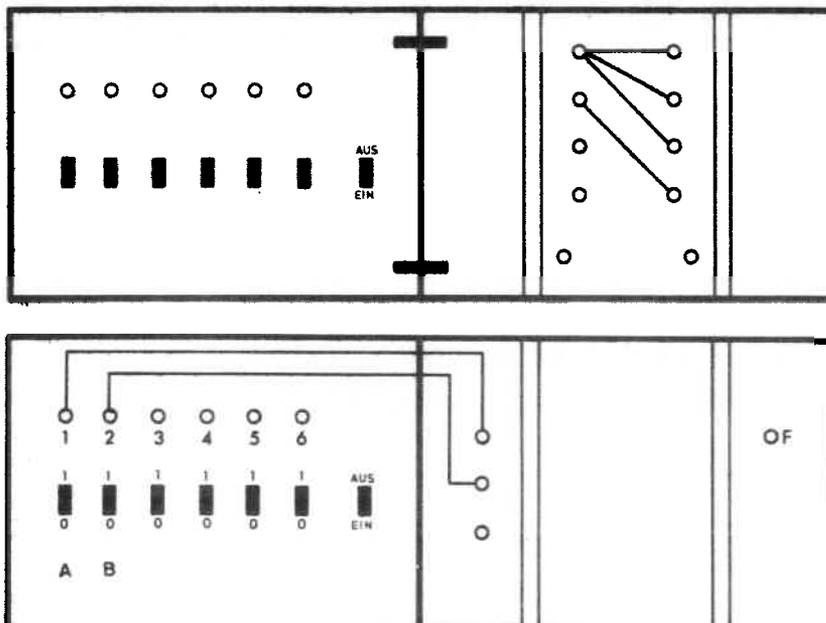


Abb.19

Wenn die Lampe aufleuchtet, gibt der Computer den Weg für die Straßenüberquerung frei.

3.3. Wann ist es hell im Zimmer? - Eine ODER-Funktion

Auch diese recht einfache Frage soll uns der Computer beantworten. Wir gehen dabei von zwei Ausgangsfragen aus:

Brennt im Zimmer eine Lampe?

Ist es draußen taghell?

Wir benötigen für die Beantwortung dieser Frage wiederum die Eingabeeinheit und einen Logik-Baustein. Programmieren Sie, wie auf der Abbildung angegeben.

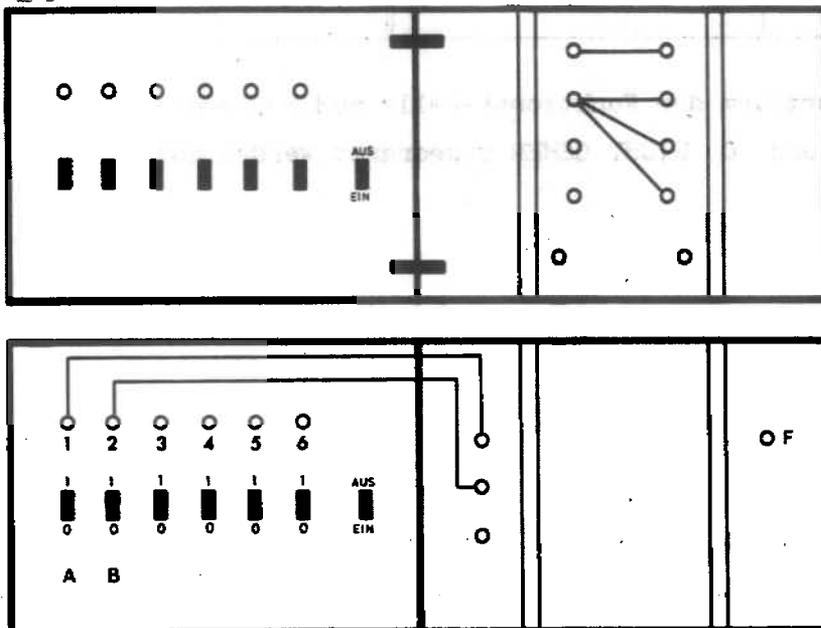


Abb.20

Der Schalter A entspricht der Frage: "Brennt im Zimmer eine Lampe?" Der Schalter B entspricht der Frage: "Ist es draußen taghell?" Für ein NEIN auf diese Fragen gelte wiederum 0, für ein JA = 1. Was sagt nun der Computer dazu? Wann ist es hell im Zimmer? Tragen Sie die Antworten des Computers in die Tabelle ein.

Dateneingabe		Datenausgabe
Brennt im Zimmer eine Lampe?	Ist es draußen taghell?	Ist es hell im Zimmer?
A	B	F
0	0	
0	1	
1	0	
1	1	

Und dies sind die Werte, die Ihr Computer ermittelt hat und durch die Lampe (0 = NEIN, 1 = JA) angezeigt hat:

Dateneingabe		Datenausgabe
Brennt im Zimmer eine Lampe?	Ist es draußen taghell?	Ist es hell im Zimmer?
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

In Worten ausgedrückt lautet das Ergebnis: Wenn im Zimmer eine Lampe brennt oder wenn es draußen taghell ist, dann ist es hell im Zimmer. Wenn wir uns das Ergebnis der Datenverarbeitung durch den Computer ansehen, müssen wir uns sprachlich noch etwas genauer ausdrücken: Im Zimmer ist es auch dann hell, wenn im Zimmer eine Lampe brennt und wenn es draußen taghell ist.

Es handelt sich hier um eine ODER-Funktion. Genau gesagt, um eine Einschließende ODER-Funktion, d.h. das Ergebnis - im Zimmer ist es hell - kommt zustande, wenn die eine oder die andere bzw. auch wenn die beiden der genannten Bedingungen erfüllt sind.

Wie schon bei der UND-Funktion fassen wir auch hier das Ergebnis in abstrakter Form in einer Funktionstabelle zusammen:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Die ODER-Funktion läßt sich elektromechanisch durch folgendes Schaltbild veranschaulichen:

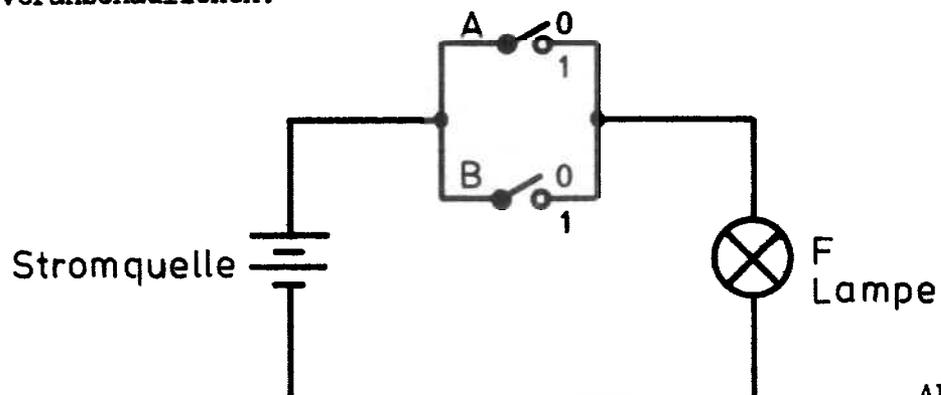
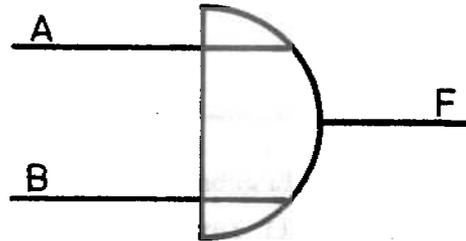


Abb. 21

Wie in Abb.21 ist hier eine Lampe mit einer Stromquelle verbunden. In den Stromkreis sind zwei Schalter eingebaut, aber dieses Mal sind die Schalter parallel angeordnet. Sie können aus der Darstellung ersehen, daß die Lampe bereits brennt, wenn nur einer der beiden Schalter A oder B eingeschaltet ist. Natürlich brennt die Lampe auch dann, wenn beide Schalter eingeschaltet sind.

Schließlich gibt es auch für die ODER-Funktion ein Schaltsymbol:



Wir haben also eine zweite logische Schaltung kennengelernt. Bei der UND-Funktion haben wir bei ihrer technischen Verwirklichung von UND-Schaltung gesprochen. Dementsprechend haben wir es mit einer ODER-Schaltung zu tun, wenn eine ODER-Funktion in Form einer Schaltung verwirklicht ist.

3.4. Wann muß ich an einer Ampel halten?

Als Teilnehmer am Straßenverkehr wissen Sie selbstverständlich, daß ROT oder GELB oder ROT und GELB an einer Verkehrsampel zum Halten zwingen. Sie sollen an diesem Beispiel auch nur versuchen, die Funktionstabelle für eine ODER-Funktion aufzustellen.

Sie können wählen, ob Sie zuerst die Tabelle aufstellen und diese dann mit Ihrem Computer überprüfen oder ob der Computer das Aufstellen übernehmen soll.

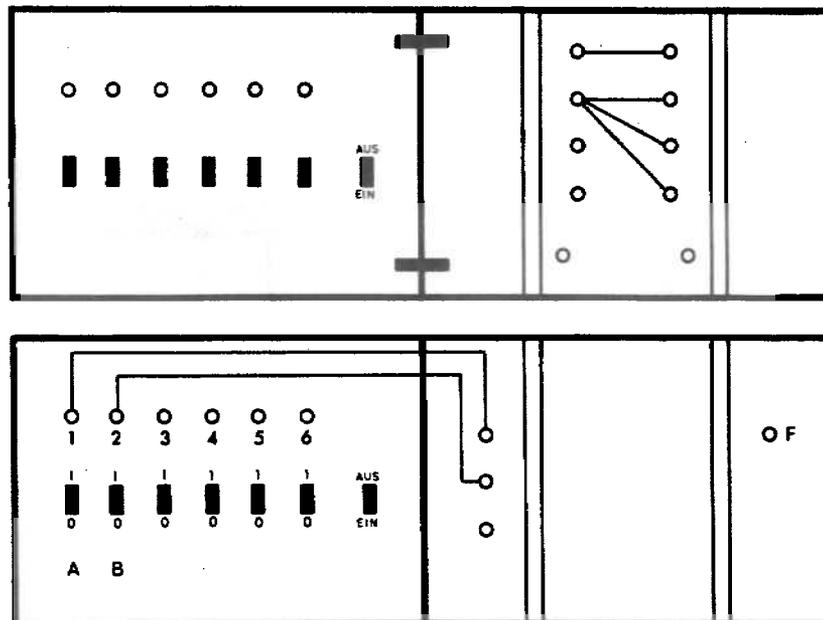


Abb.22

Bevor Sie mit dem Computer arbeiten, muß noch die Fragestellung festgelegt werden:

Schalter A:	Ampel ROT (A)	JA	1
		NEIN	0
Schalter B:	Ampel GELB (B)	JA	1
		NEIN	0
Datenausgabe:	Halten (L)	JA	1
		NEIN	0

Jetzt können Sie auch die ausführliche Funktionstabelle entwickeln.

Ampel ROT	Ampel GELB	Halten
A	B	F

Bitte, verwenden Sie in der Kurzform nur 0 und 1.

A	B	F

Leuchtet die Lampe der Datenausgabe auf, so müssen Sie vor der Ampel anhalten.

Wenn Ihnen das Aufstellen der Funktionstabelle und das Überprüfen Spaß gemacht haben, versuchen Sie vielleicht auch noch, ohne weitere Hilfe die nächste ODER-Funktion zu erproben. Sie können dazu dieselbe Schaltung und Programmierung einsetzen wie bei der vorigen Funktion.

Wann ist Abblendlicht einzuschalten? Die Straßenverkehrsordnung schreibt vor, Abblendlicht einzuschalten, wenn es dunkel oder neblig ist.

Funktionstabelle:

A	B	F

3.5. Wie komme ich nach München? - Eine Exklusiv-ODER-Funktion

Sie wollen von Hamburg nach München reisen. Dafür stellen Sie sich zwei Fragen:

1. Reise ich mit dem Auto?
2. Reise ich mit dem Flugzeug?

Wollen Sie das Ergebnis aus diesen beiden Fragen von Ihrem Computer erfahren, müssen Sie ihn so programmieren (Frage 1: Schalter A; Frage 2: Schalter B):

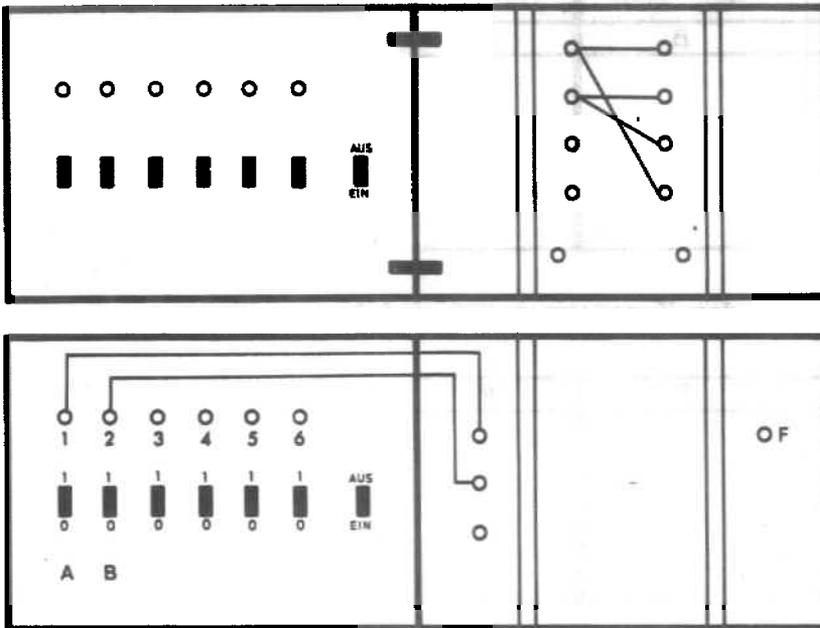


Abb.23

Hier wieder die entsprechende Tabelle, in die Sie die Computer-Antworten eintragen:

Dateneingabe		Datenausgabe
Reise ich mit dem Auto?	Reise ich mit dem Flugzeug?	Komme ich nach München?
A	B	F
0	0	
0	1	
1	0	
1	1	

Das ist die Lösung:

Dateneingabe		Datenausgabe
Reise ich mit dem Auto?	Reise ich mit dem Flugzeug?	Komme ich nach München?
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Sie kommen also nach München, wenn Sie entweder mit dem Auto oder mit dem Flugzeug reisen. Sollten Sie vorhaben, mit dem Auto und mit dem Flugzeug zu reisen, dann sagt Ihnen der Computer - und natürlich auch Ihre eigene Überlegung - daß Sie nicht nach München kommen. Denn mit Auto und Flugzeug zugleich, können Sie nicht reisen.

Im Unterschied zum einschließlichen ODER haben wir es hier mit einem ausschließlichen ODER zu tun. Die logische Funktion heißt dementsprechend ausschließliche ODER-Funktion bzw. Exklusiv-ODER-Funktion. Nur dann kommt das Ergebnis zustande, wenn entweder die eine Bedingung erfüllt ist oder die andere.

Die Funktionstabelle in Kurzform:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Da wir es auch hier mit einer logischen Schaltung zu tun haben, veranschaulichen Sie sich die Zusammenhänge wieder anhand der entsprechenden elektromechanischen Schaltung:

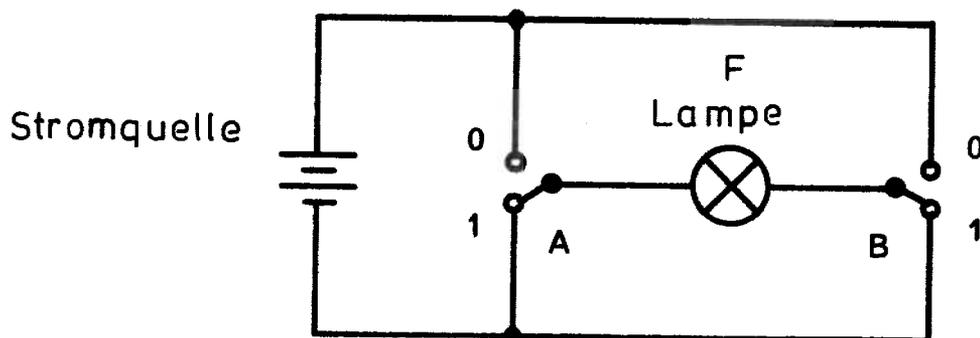
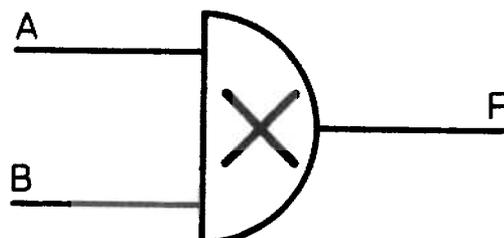


Abb. 24

Und dies ist das Symbol für die Exklusiv-ODER-Schaltung:



3.6. Sommerurlaub

Kurz vor Ihrem Sommerurlaub, den Sie im Ausland verbringen möchten, haben Sie sich immer noch nicht für ein Ziel entscheiden können. Im Reisebüro teilt man Ihnen bedauernd mit, daß für die vorgesehene Zeit nur noch Plätze in Jugoslawien und auf Mallorca frei sind. Jetzt müssen Sie sich schnell entschließen: Entweder Sie fliegen nach Jugoslawien oder nach Mallorca.

Natürlich kann Ihnen der Computer diese schwere Entscheidung nicht abnehmen, aber er kann Ihnen die Antwort geben, ob Sie den Urlaub im Ausland verbringen werden oder nicht. (Vorausgesetzt, Sie finden keine andere Möglichkeit, ins Ausland zu reisen.)

Sie merken sicherlich schon, daß auch dieser Fall ein Beispiel für eine Exklusiv-ODER-Funktion ist. Denn es gibt nur die Möglichkeit, nach Jugoslawien oder nach Mallorca zu reisen. Für beide Ziele können Sie sich eben nicht gleichzeitig anmelden. Es gilt also:

Reise ich nach Jugoslawien? (Schalter A)

Reise ich nach Mallorca? (Schalter B)

Wenn Sie den Logik-Baustein wie auf der folgenden Abbildung programmieren, gibt Ihnen der Computer die entsprechende Antwort.

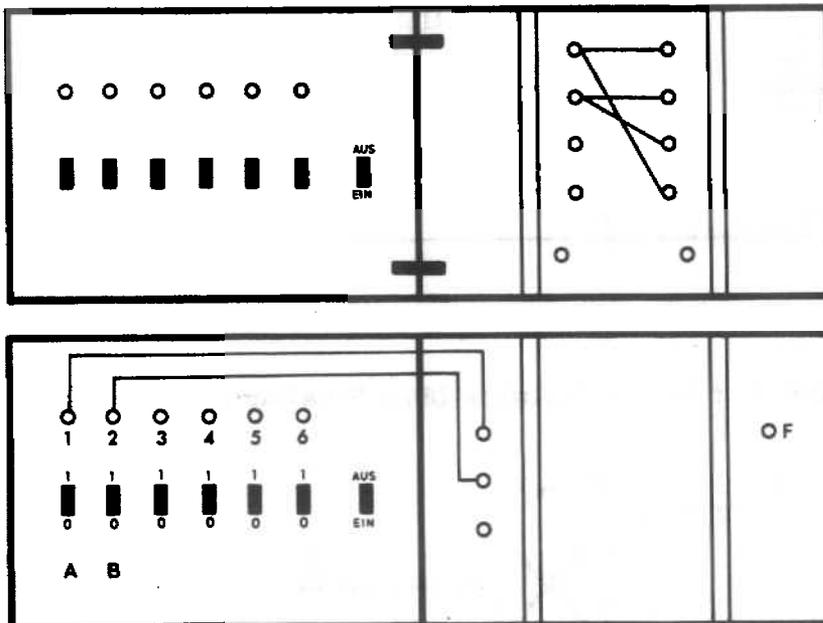


Abb.25

Dateneingabe		Datenausgabe
Jugoslawien	Mallorca	Auslandsaufenthalt
A	B	F
0	0	
0	1	
1	0	
1	1	

Das Aufleuchten der Lampe (Datenausgabe) bedeutet, in welchem Fall Sie Ihren Urlaub im Ausland verbringen werden. Ganz bestimmt gelingt es Ihnen, die Kurzform der Funktionstabelle zu ermitteln. Sie brauchen die Eingabebefehle und den Ausgang nur in der Form 0 oder 1 einzusetzen.

A	B	F

Sicherlich fallen Ihnen noch weitere Beispiele zu einem Exklusiv-ODER ein. Versuchen Sie doch einmal, dazu die Funktionstabellen aufzustellen und diese mit dem Computer zu überprüfen. Der umgekehrte Weg ist natürlich auch möglich.

3.7. Wann ist die Waage im Gleichgewicht?

Für die hier gezeigte Balkenwaage haben wir zwei gleichschwere Gewichte zur Verfügung.

Selbstverständlich ist leicht zu erkennen, in welchen Fällen sich diese Waage im Gleichgewicht befindet. Probieren wir die verschiedenen Möglichkeiten mit unserem Computer durch. Programmieren Sie dazu wie folgt:

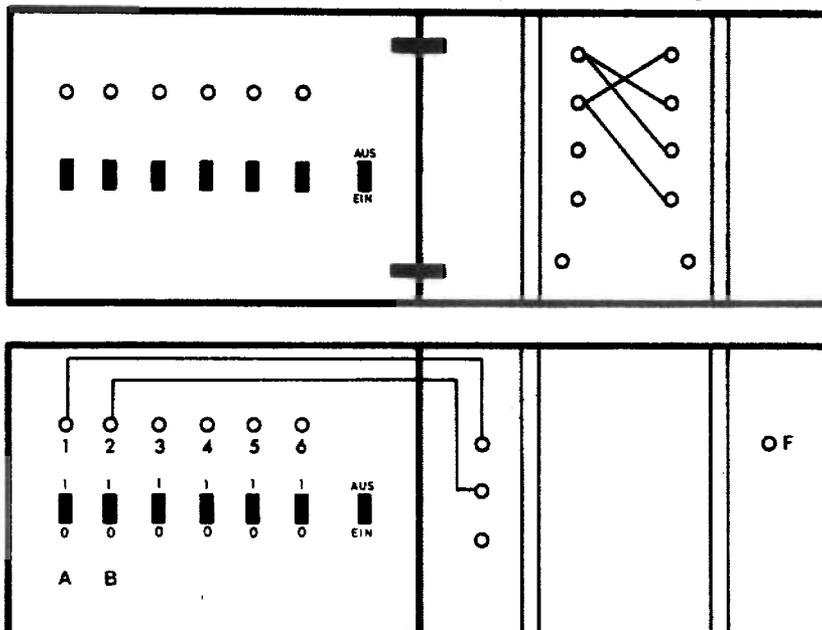


Abb. 26

Treffen wir nun die notwendigen Zuordnungen:

Kein Gewicht auf der Waagschale : 0
 Gewicht auf der Waagschale : 1
 Waage nicht im Gleichgewicht : 0
 Waage im Gleichgewicht : 1

Ermitteln Sie nun, in welchen Fällen sich die Waage im Gleichgewicht befindet:

Dateneingabe		Datenausgabe
Liegt ein Gewicht auf der linken Waagschale?	Liegt ein Gewicht auf der rechten Waagschale?	Ist die Waage im Gleichgewicht?
A	B	F
0	0	
0	1	
1	0	
1	1	

Die Lösung ist sehr einleuchtend:

Dateneingabe		Datenausgabe
Liegt ein Gewicht auf der linken Waagschale?	Liegt ein Gewicht auf der rechten Waagschale?	Ist die Waage im Gleichgewicht?
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Die Waage ist nur dann im Gleichgewicht, wenn beide Seiten gleich stark belastet (oder nicht belastet) sind. Mit anderen Worten: Wenn die linke Waagschale nicht belastet ist und die rechte Waagschale nicht belastet ist bzw. wenn die linke Waagschale belastet ist und wenn die rechte Waagschale belastet ist (gleiche Gewichte), dann befindet sich die Waage im

Gleichgewicht. Es handelt sich hier um die sog. Äquivalenz-Funktion, was zu deutsch soviel wie Gleichheitsfunktion heißt.

Hier wieder die entsprechende Funktionstabelle in abstrahierter Form:

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Und dies ist die
Äquivalenz-Schaltung
 in elektromechanischer
 Form:

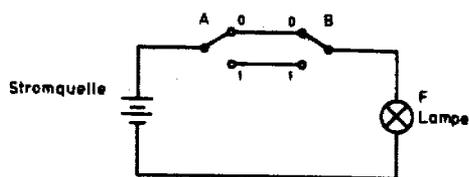
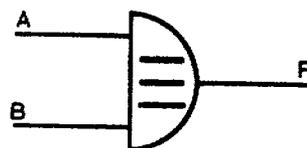


Abb.27

Das Schaltsymbol für die
Äquivalenz-Schaltung
 sieht so aus:



3.8. Bleibt die Erde bei Regen trocken? - Identität und Negation

Was für eine törichte Frage! - Aber lassen Sie uns trotzdem kurz bei dieser Frage verweilen, um die zwei einfachsten logischen Funktionen kennenzulernen.

Stellen wir uns die Frage zunächst anders herum: Wird die Erde naß, wenn es regnet? Wollen wir die Antwort auf diese "schwierige" Frage von unserem Computer haben, dann müssen wir ihn so programmieren:

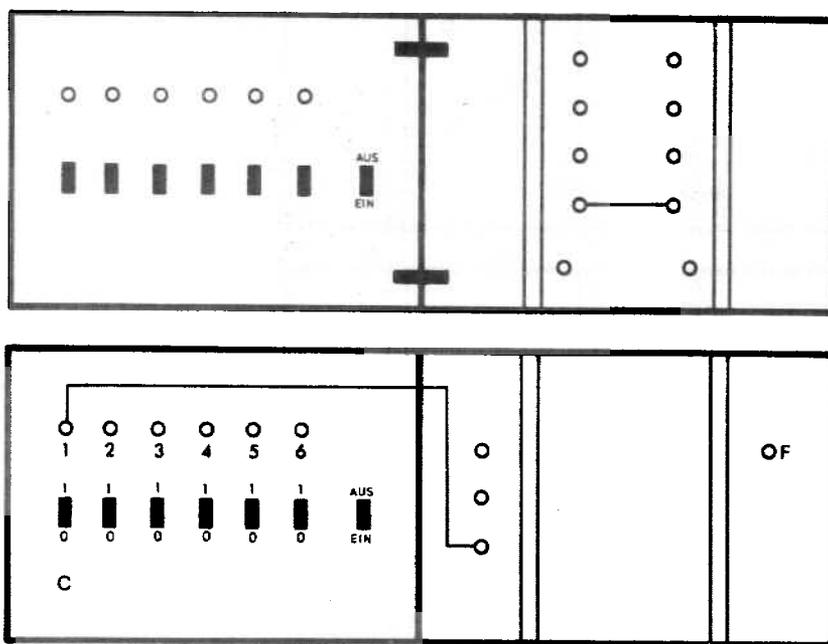


Abb.28

Wenn wir unsere Ergebnisse wieder in eine Tabelle eintragen und dabei wieder für NEIN = 0 und für JA = 1 setzen, dann ergibt sich folgendes Bild:

Regnet es?	Wird die Erde naß?
C	F
0	0
1	1

Mit anderen Worten: Wenn es regnet, dann wird die Erde naß, bzw.: Wenn es nicht regnet, dann wird die Erde nicht naß. Diese logische Funktion ist die sog. Identitäts-Funktion, zu deutsch könnte man sagen, Übereinstimmungsfunktion.

Als Schaltbild sieht das so aus:

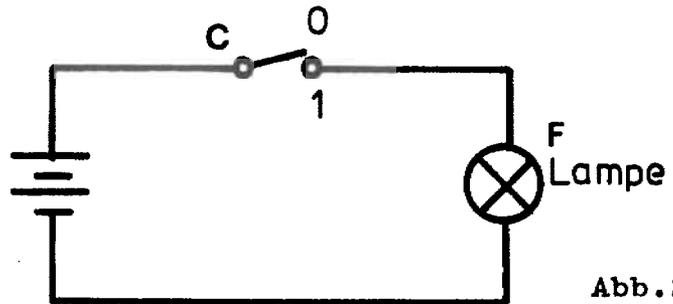


Abb. 29

Wenn der Schalter geschlossen ist, dann brennt die Lampe, und wenn der Schalter nicht geschlossen ist, dann brennt die Lampe nicht.

Wie steht es nun mit der Frage: "Bleibt die Erde trocken, wenn es regnet?" Programmieren Sie den Logik-Baustein jetzt wie folgt:

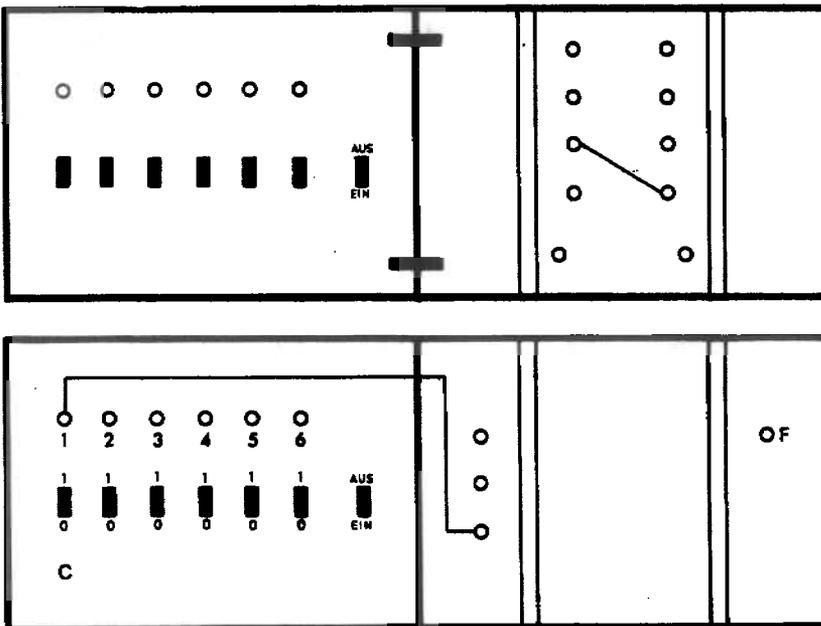


Abb. 30

Die zugehörige Funktionstabelle:

Regnet es?	Bleibt die Erde trocken?
C	F
0	1
1	0

Wenn es nicht regnet, dann bleibt die Erde trocken. Wenn es regnet, dann bleibt die Erde nicht trocken. Bei dieser logischen Funktion handelt es sich um eine Negations-Funktion, was soviel heißt wie Verneinungsfunktion.

Die Funktionstabellen
für Identität
und Negation:

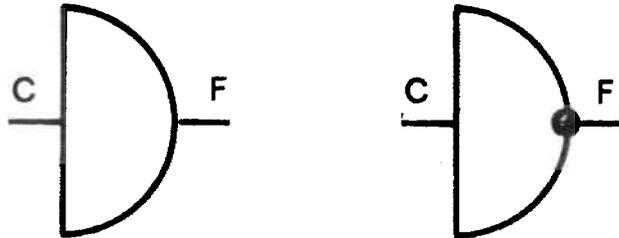
Identität

C	F
0	0
1	1

Negation

C	F
0	1
1	0

und die Schaltsymbole:



3.9. Welches ist der richtige Weg?

Ein Wanderer trifft an einer Wegegabelung auf einen Mann. Über diesen Mann ist bekannt, daß er einer von zwei Brüdern ist, die je eine sehr ausgeprägte Eigenschaft besitzen: Einer der Brüder lügt immer, der andere sagt stets die Wahrheit. Dem Wanderer - er kennt nämlich den Weg nicht - stellt sich das Problem durch Befragung des Mannes, von dem er nicht weiß, ob er der Lügner (L) oder der Wahrheitssager (W) ist, auf den richtigen Weg zu gelangen.

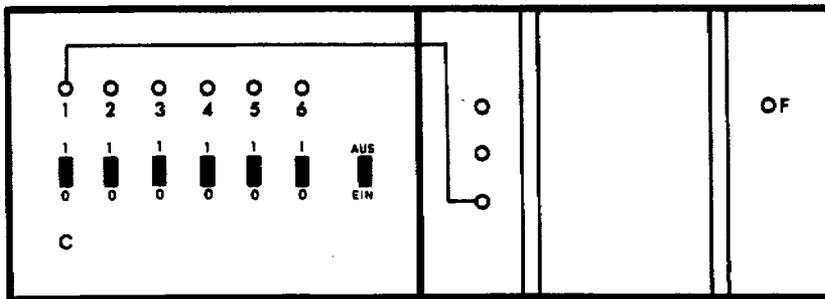
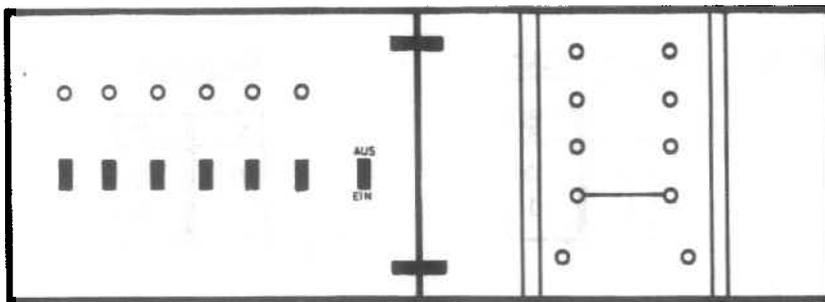
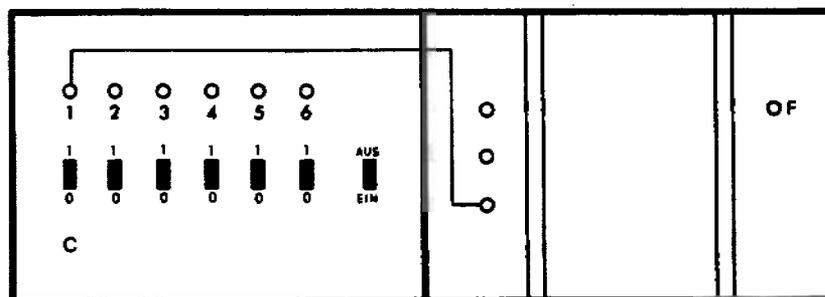
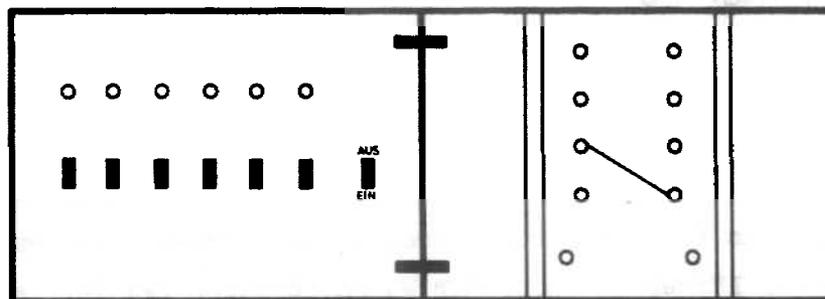
Fragt er Bruder W, ob dies der richtige Weg sei, indem er gleichzeitig auf den richtigen Weg weist, so wird der mit JA antworten. Zeigt er auf den falschen Weg, so wird die Antwort NEIN sein.

Dieser Bruder W verändert also die mit der Frage gegebene Information nicht, sondern er bestätigt sie oder bestätigt sie auch nicht.

Richtiger Weg	Bruder W
C	F
0	0
1	1

Es handelt sich also um eine Identitätsfunktion. - Bruder L dagegen würde die Information verändern: Stellt ihm der Wanderer dieselbe Frage und zeigt dabei auf den richtigen Weg, dann antwortet L mit NEIN. Weist er bei dieser Fragestellung auf den falschen Weg, lautet die Antwort JA.

Richtiger Weg	Bruder L
C	F
0	1
1	0

Abb. 31
IdentitätAbb. 32
Negation

Er kehrt also die gegebene Information um, so daß es sich um eine Negationsfunktion handelt. Der Ausgang des Logik-Bausteins stellt die Umkehrung des Eingangs dar.

Der arme Wanderer muß sich also eine Frage einfallen lassen, auf die er eine eindeutige Antwort bekommt. Das ist nur möglich, wenn eine Frage beide Brüder gleichzeitig einschließt. Sie muß also lauten: "Würde dein Bruder zustimmen, daß dies der richtige Weg ist?", wobei er auf irgendeinen Weg deutet.

Bruder W antwortet, sofern es sich um den richtigen Weg handelt, mit NEIN. Denn sein Bruder L würde dann nicht zustimmen. Bruder L dagegen antwortet im gleichen Fall auch mit NEIN. Denn da er immer lügt, wird er die Antwort von W wieder umkehren.

Durch diese geschickte Frage gelangt der Wanderer doch noch an sein Ziel.

3.10. Eine Badewanne wird gefüllt

Für das Füllen einer Badewanne kommen folgende Eingangsbedingungen in Frage:

- Der Abfluß ist geöffnet (A = 0)
- Der Abfluß ist geschlossen (A = 1)
- Der Kaltwasserhahn ist geschlossen (B = 0)
- Der Kaltwasserhahn ist geöffnet (B = 1)
- Der Warmwasserhahn ist geschlossen (C = 0)
- Der Warmwasserhahn ist geöffnet (C = 1)

Bitte, achten Sie darauf, daß wir bei den Wasserhähnen sprachlich anders verfahren als bei den Stromschaltern unserer vorangegangenen Beispiele: Wenn wir sagen, ein Stromschalter ist geöffnet, dann heißt das: Es fließt kein Strom (0). Wenn wir dagegen von einem Wasserhahn sagen, daß er geöffnet ist, so bedeutet das: Das Wasser fließt (1).

Wollen Sie nun mit Hilfe eines Logik-Bausteins erfahren, unter welchen Bedingungen sich die Badewanne mit Wasser füllt, so nehmen Sie folgende Programmierung vor:

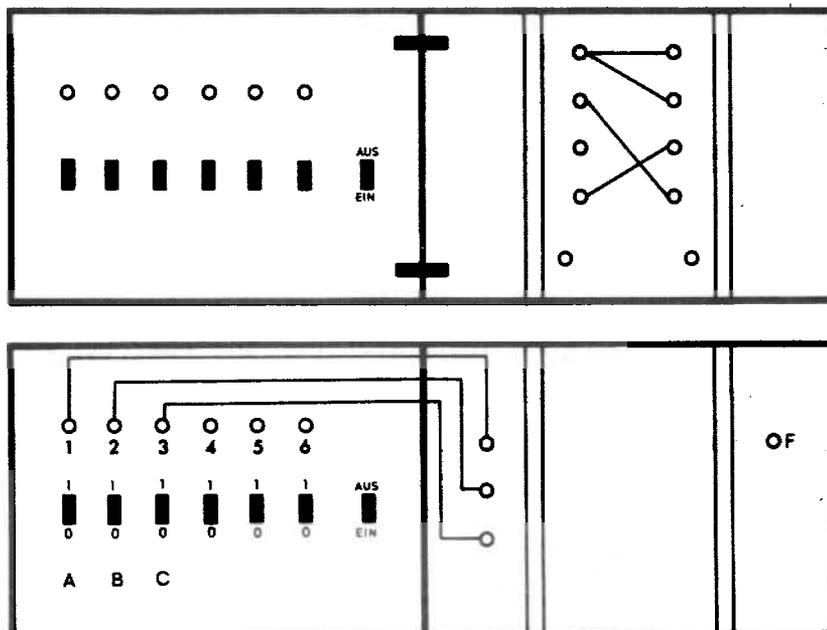


Abb. 33

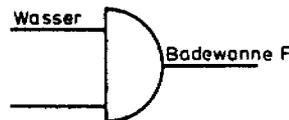
Tragen Sie die Ergebnisse wieder in die Tabelle ein:

Abfluß geschlossen?	Kaltwasserhahn geöffnet?	Warmwasserhahn geöffnet?	Wird Badewanne gefüllt
A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

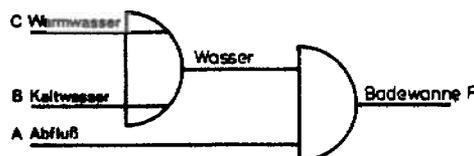
Das Ergebnis:

Abfluß geschlossen?	Kaltwasserhahn geöffnet?	Warmwasserhahn geöffnet?	Wird Badewanne gefüllt
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Das Ergebnis ist ganz einleuchtend, wenn Sie sich die Zusammenhänge sprachlich vor Augen halten: Wenn der Abfluß geschlossen ist und wenn der Kaltwasserhahn oder der Warmwasserhahn geöffnet ist, dann wird die Wanne mit Wasser gefüllt. Mit anderen Worten: Der Abfluß muß auf jeden Fall geschlossen sein, und es muß in jedem Fall Wasser laufen. Damit haben wir eine UND-Funktion.



Das Wasser fließt, wenn der Kaltwasserhahn geöffnet ist oder wenn der Warmwasserhahn geöffnet ist oder auch wenn beide geöffnet sind. Damit sind Kaltwasserhahn und Warmwasserhahn in einer ODER-Funktion verbunden. So ergibt sich insgesamt folgendes Schaltbild:



Zum Abschluß dieses Beispiels noch die Funktionstabelle in allgemeiner Form:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

3.11. Ein Lagerfeuer wird entzündet

Ein weiteres Beispiel für die Verknüpfung einer UND-Funktion mit einer ODER-Funktion:

Sie möchten auf einem freien Platz ein kleines Lagerfeuer entzünden. Voraussetzung dafür sind in jedem Fall, daß Brennmaterial und Streichhölzer vorhanden sind. Diese beiden Eingangsbedingungen stellen in diesem Fall eine UND-Funktion dar.

Nun müssen Sie das Feuer aber nicht unbedingt mit Streichhölzern entzünden, sondern Sie können natürlich auch ein Feuerzeug einsetzen. Dies ist ein Einschließendes ODER. Denn es müssen entweder Streichhölzer oder ein Feuerzeug vorhanden sein. (Beides ist natürlich auch möglich und in der Praxis manchmal sehr nützlich.) Fassen wir also alle Eingangsbedingungen zusammen:

Brennmaterial nicht vorhanden	0	Schalter A
Brennmaterial vorhanden	1	Schalter A
Streichhölzer nicht vorhanden	0	Schalter B
Streichhölzer vorhanden	1	Schalter B
Feuerzeug nicht vorhanden	0	Schalter C
Feuerzeug vorhanden	1	Schalter C

Das Aufleuchten der Lampe des Logik-Bausteins zeigt an, unter welchen Bedingungen Sie ein Feuer entzünden können. Doch zunächst gilt es, den Logik-Baustein zu programmieren und mit der Eingabeeinheit zu verbinden.

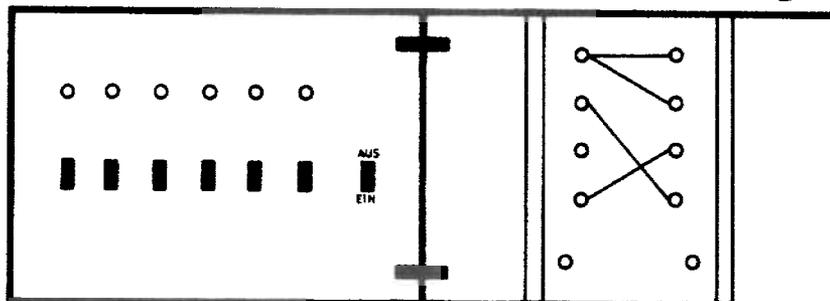


Abb. 34

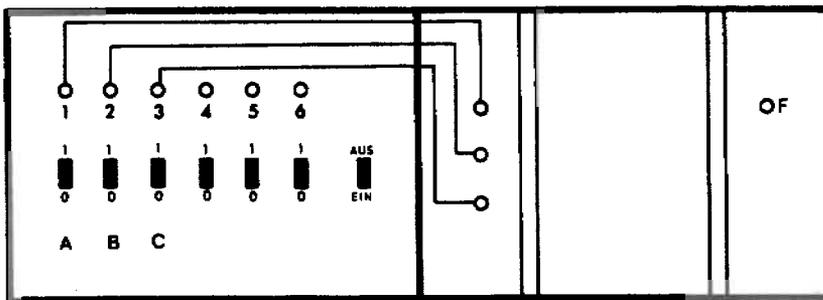


Abb.34

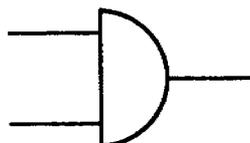
Die Ausgabedaten können Sie in die Funktionstabelle eintragen. (Lampe AUS = 0, Lampe EIN = 1)

Brennmaterial vorhanden?	Streichhölzer vorhanden?	Feuerzeug vorhanden?	Feuer brennt
A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Formuliert lautet das Ergebnis: Wenn Brennmaterial und Streichhölzer oder ein Feuerzeug vorhanden sind, lässt sich das Lagerfeuer entzünden. In der Kurzform sieht die Funktionstabelle so aus:

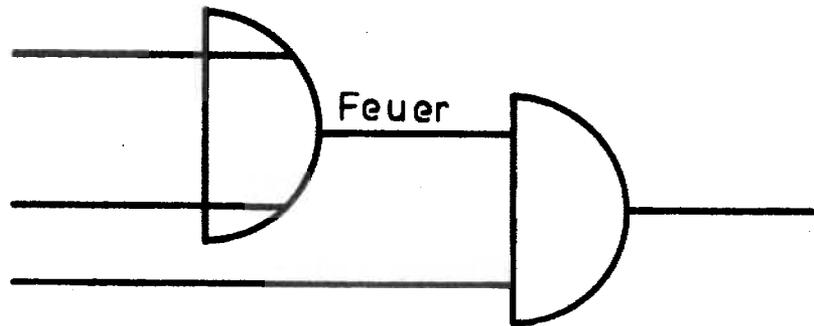
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Versuchen wir auch für dieses Beispiel, die Schaltsymbole für die UND-Funktionen einzubeziehen.



Sicherlich bereitet es Ihnen keine Mühe, die Eingänge und den Ausgang dieses UND-Symbols - bezogen auf den beschriebenen Fall - zu beschriften.

Verknüpft man die UND-Funktion mit der ODER-Funktion und stellt diese Verknüpfung durch die Symbole dar, entsteht folgendes Bild:



Wie müssen die Eingänge und der Ausgang beschriftet werden?

Bevor wir nun weiter in die Computerlogik eindringen und bevor wir mit mehreren Logik-Bausteinen experimentieren, lassen Sie uns noch einige grundsätzliche Betrachtungen über Datenverarbeitung anstellen.

4. Aufbau eines Computers

4.1. Ist der Mensch ein Computer?

Diese Frage drängt sich uns unweigerlich auf, wenn wir uns die erstaunlichen Ähnlichkeiten zwischen menschlichem Denken und Datenverarbeitung durch einen Computer veranschaulichen.

In dem Abschnitt über Begriffserklärungen hatten wir schon besprochen, was Daten sind. Daten sind Informationen. Information ist alles das, was uns in irgendeiner Form eine Bedeutung übermittelt, also z.B. gesprochene oder geschriebene Wörter und Sätze, Zahlen, Zeichnungen und Bilder. Datenverarbeitung ist die Verarbeitung von Eingabedaten mittels eines Programms zu Ausgabedaten. Statt Dateneingabe können wir auch Datenaufnahme sagen: Ein Computer nimmt Daten auf, die ihm eingegeben werden, und auch Sie selbst nehmen Daten auf.

Was geschieht z.B. jetzt im Augenblick? Sie lesen den Text dieser Anleitung. Sie nehmen die Buchstaben bzw. Wörter, die Zahlen und Abbildungen auf. Das ist kurz gesagt Datenaufnahme. Dafür, also für Sehen und Lesen, haben Sie ein spezielles Instrumentarium: Ihre Augen. Ebenso ist Hören Datenaufnahme, durch die Ohren als Datenaufnahmeinstrumente. Wir können auch die anderen drei unserer fünf Sinne als Funktionen mit Datenaufnahme verstehen: Fühlen, Schmecken, Riechen. Unsere Instrumente dazu: Haut, Zunge, Nase bzw. die entsprechenden Nerven.

Und was geht beim Sprechen vor sich? - Es werden Daten ausgegeben. Sprechen ist also eine Funktion der Datenausgabe. Das entsprechende Datenausgabeinstrument: Der Mund. Daten geben wir auch aus, wenn wir etwas schreiben oder zeichnen. Im allgemeinen ist unsere rechte Hand - mit Schreib- oder Zeichenstift - dabei das Datenausgabeinstrument. Wir haben noch weitere Möglichkeiten, uns verständlich zu machen. Bekanntlich kann auch Gestikulieren etwas Bestimmtes bedeuten. Gestikulieren ist also ebenfalls Datenausgabe, und zwar durch das Instrumentarium der Hände. Dann haben wir noch das Gesicht als Ausdrucksmöglichkeit, indem wir durch die Mimik Informationen, also Daten, weitergeben.

Zwischen der Datenaufnahme und der Datenausgabe liegt die eigentliche Datenverarbeitung. Wir können auch von Datenverarbeitung im engeren Sinn sprechen, wenn wir Datenaufnahme, Datenverarbeitung (im engeren Sinn) und Datenausgabe zusammen als Datenverarbeitung im weiteren Sinn verstehen. Wo findet die Datenverarbeitung im engeren Sinn bei uns statt? - Im Gehirn. - Und was ist das nun im einzelnen, Datenverarbeitung im

engeren Sinn? - Veranschaulichen Sie sich das am besten anhand Ihrer gegenwärtigen Situation. Sie lesen dieses Anleitungsbuch, Sie lesen den Text und schauen sich die Abbildungen an. Sie wissen bereits: Das bedeutet zunächst einmal Datenaufnahme. Einen Teil des Inhalts dieses Buches nehmen Sie in Ihr Gedächtnis auf, Sie speichern Daten. Aber das ist noch das wenigste. Was Sie insgesamt bei der Datenverarbeitung tun, bezeichnen wir als "denken". Und Denken ist eine überaus komplizierte Angelegenheit. Wir wollen dem Begriff Denken nicht bis auf den Grund gehen, das würde viel zu kompliziert werden; wir wollen uns statt dessen mit einigen Andeutungen zufrieden geben.

Funktionen des Denkens, also der Datenverarbeitung im engeren Sinn, sind:

<u>Ordnen:</u> F	A	<u>Vergleichen:</u> 6	?	8
D	B			
C	C			
A	D			
E	E			
B	F			

Kontrollieren: $3 \times 4 = 13$ Kontrolle: Korrektur $3 \times 4 = 12$

Schlußfolgern: Der Badewannenabfluß ist geschlossen und der Kaltwasserhahn ist geöffnet. Daraus folgt: Die Badewanne wird mit Wasser gefüllt.

Entscheiden: Beschäftige ich mich auch weiterhin mit meinem Philips Computer-Lehrbaukasten? - Ja/Nein.

Mit diesen Beispielen für Funktionen des Denkens wollen wir es genug sein lassen. Fassen wir diese Funktionen unter einem neuen Oberbegriff zusammen: Steuern.

Unter der Funktion Steuern wollen wir hauptsächlich jene Funktionen verstehen, die dazu dienen, Datenverarbeitung durchzuführen bzw. Verarbeitung von Daten zu steuern. Das mag Ihnen im Moment etwas undurchsichtig erscheinen, klarer wird es jedenfalls noch im Zusammenhang mit Datenverarbeitung durch den Computer.

Eine Funktion der Datenverarbeitung im engeren Sinn bleibt aber noch zu erwähnen: Rechnen.

Die drei Hauptfunktionen der Datenverarbeitung im engeren Sinn sind also: Speichern, Steuern, Rechnen.

Wir sprachen vorhin davon, daß Sie sich einen Teil dieses Anleitungsbuches im Gedächtnis merken werden. Wenn Sie sich ausgiebig mit dem Anleitungsbuch und Ihrem Philips Computer-Lehrbaukasten befassen, werden Ihnen die wichtigsten Zusammenhänge und sicherlich auch manch ein Detail

im Gedächtnis haften bleiben. Aber Sie werden sich den Text nicht wortwörtlich einverleiben, das wäre auch nicht unbedingt sinnvoll. Gewiß ist es eine nützliche Einrichtung, daß wir nicht jede Kleinigkeit für alle Zeiten im Gedächtnis behalten, das würde unsere Gedanken nur belasten. Auf der anderen Seite aber gibt es viele Dinge, die wir gerne behalten würden. Das ist jedoch nicht immer ohne weiteres möglich, da wir erstens in einer bestimmten Zeit nur eine bestimmte Menge von Daten aufnehmen können und da wir zweitens auch vergeßlich sind. Der Schwierigkeit kann abgeholfen werden, denn unser Gedächtnis ist ja nicht die einzige Möglichkeit zur Speicherung von Daten.

Nehmen wir an, Sie hören einen interessanten Vortrag; vielleicht machen Sie sich Notizen. Oder ein Freund teilt Ihnen seine neue Telefonnummer mit; Sie notieren sich die Nummer. Oder aber Sie haben nach einiger Zeit Dinge aus diesem Anleitungsbuch, die Sie besonders interessieren, nicht mehr ganz gegenwärtig. Was werden Sie tun? Sie nehmen sich die Anleitung wieder zur Hand und lesen nach. In allen drei angeführten Fällen speichern Sie Daten auf externen Datenspeichern im Unterschied zu Ihrem internen Datenspeicher, Ihrem Gedächtnis. Natürlich können Sie zur externen Datenspeicherung auch Tonbandgerät und Tonbänder benutzen. In jedem Fall: Durch externe Datenspeicherung können wir unser Gedächtnis entlasten.

Nicht nur die Datenspeicherung können wir uns durch Hilfsmittel erleichtern, auch die Datenverarbeitung muß nicht in allen Einzelheiten in unserem Gehirn vor sich gehen. Einfache Hilfsmittel zur externen Datenverarbeitung sind z.B. Rechenschieber oder die Registriertafel mit Additionswerk. Externe Datenverarbeitung betreiben Sie aber auch, wenn Sie mit Ihrem Philips Computer-Lehrbaukasten experimentieren. Darüber hinaus stehen uns heute Hilfsmittel zur Verfügung, die komplizierteste Datenverarbeitungsaufgaben vollautomatisch durchführen, und das in unvorstellbaren Geschwindigkeiten: Computer.

4.2. Ein großer Computer wird vorgestellt

Befassen wir uns ein wenig mit dem Aufbau eines großen Computers. Nicht zuletzt deshalb, weil es für Sie interessant ist zu wissen, in welcher Beziehung die Funktionen Ihres Computer-Lehrbaukastens zu den Funktionen einer großen Datenverarbeitungsanlage stehen.

Wie Sie bereits wissen, beginnt jede Datenverarbeitung mit der Datenaufnahme. Nachdem wir bei der Datenverarbeitung des Menschen von Datenaufnahme gesprochen haben, wollen wir beim Computer wieder Dateneingabe

sagen, um damit anzudeuten, daß sich der Computer die Daten nicht selbst besorgt, sondern daß der Mensch ihm die Daten eingibt. Das geht natürlich nicht auf beliebigen Notizzetteln oder etwa im zwanglosen Gespräch mit dem Computer. Wir können zwar mit Datenträgern wie Notizzetteln und Büchern etwas anfangen, der Computer aber braucht seine Daten auf maschinell lesbaren Datenträgern. Der bekannteste maschinell lesbare Datenträger ist die Lochkarte.

Daten, die für den Computer lesbar auf Lochkarten verschlüsselt bzw., wie man auch sagt, codiert sind, werden über einen Lochkartenleser in den Computer eingegeben. Ein anderer, maschinell lesbarer Datenträger ist der Lochstreifen. Der Inhalt von Lochstreifen wird über einen Lochstreifenleser eingelesen. Außer der Dateneingabe über Lochkarten und Lochstreifen ist auch eine direkte Dateneingabe über eine Tastatur möglich. Eine solche Tastatur ist mit Zahlen- und Buchstabentasten versehen.

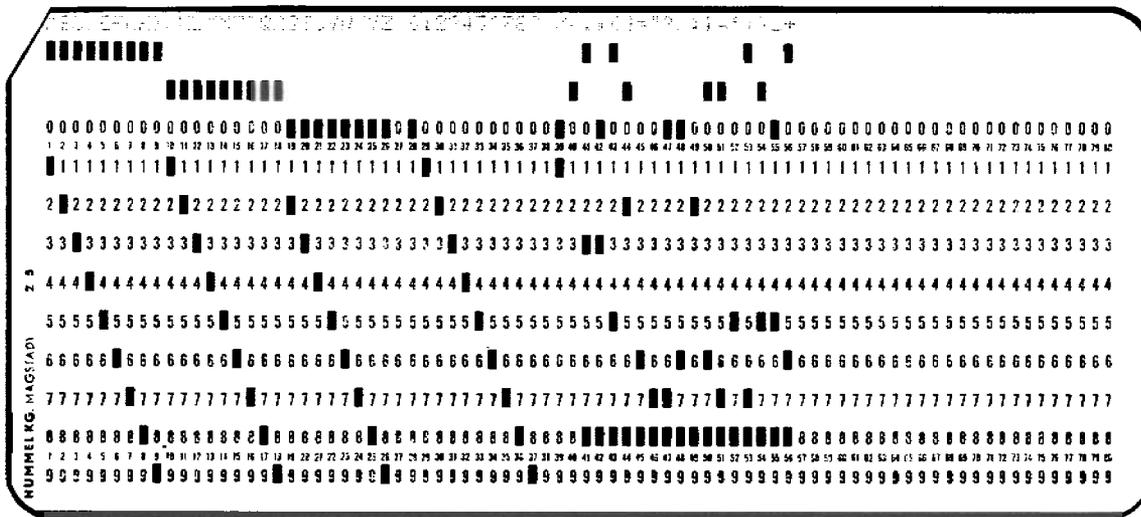
Wie beim Menschen gibt es auch beim Computer Datenausgabe. Computer können Daten beispielsweise über Lochkartenstanzer auf Lochkarten ausgeben. Außerdem können Ausgabedaten durch Lochstreifenstanzer in Lochstreifen gestanzt werden, und schließlich können Daten auch in Schrift, also Buchstaben, Ziffern und in Zeichen wie . , ; + % usw. über Schnelldrucker auf Listen ausgegeben werden.

Die eigentliche Datenverarbeitung, also die Datenverarbeitung im engeren Sinn, findet beim Computer in der sog. Zentraleinheit statt. Die Zentraleinheit ist gewissermaßen das Gehirn des Computers. In der Zentraleinheit hat die Elektronik ihren Hauptwirkungsbereich. Hier finden, oft in Bruchteilen von Sekunden, die kompliziertesten Datenverarbeitungsvorgänge statt - mit einer Zuverlässigkeit, wie sie der Mensch niemals erreichen könnte. Wenn wir die Zentraleinheit als das Gehirn des Computers verstehen, so entspricht der Zentralspeicher dem Gedächtnis. Im Zentralspeicher werden Daten intern gespeichert.

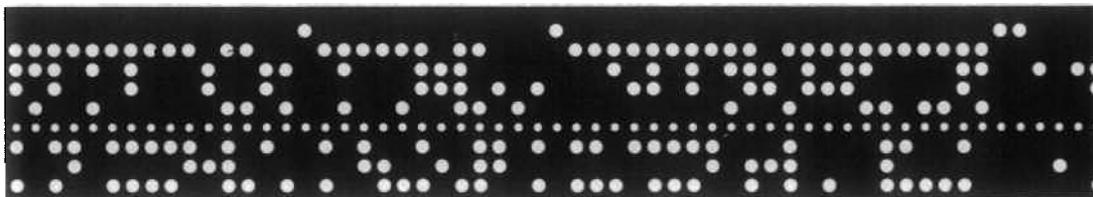
Das Rechenwerk ist für alle Rechenoperationen zuständig. Dabei mag folgender Sachverhalt erstaunlich erscheinen: Das Rechenwerk verfährt auch mit Buchstaben wie mit Zahlen, da Buchstaben in einem Computer wie Zahlen verschlüsselt werden. Aber das sei nur am Rande vermerkt. Die Rechenoperationen beruhen zum größten Teil wiederum auf logischen Funktionen, von denen Sie ja schon einige kennengelernt haben.

Das Steuerwerk schließlich führt gewissermaßen Regie bei der Datenverarbeitung. Das Steuerwerk sorgt dafür, daß Daten eingelesen, nach Programm verarbeitet und ausgegeben werden.

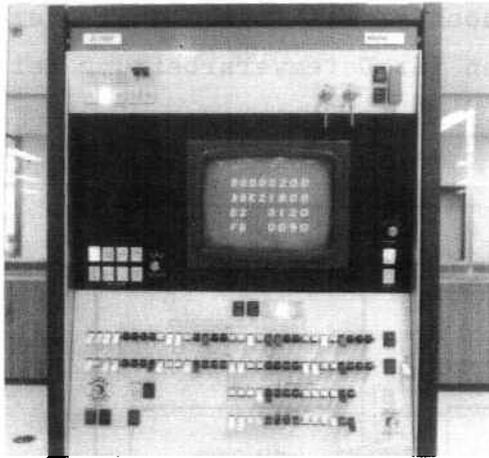
Der Mensch kann sein Gedächtnis durch externe Datenspeicherung entlasten. Auch ein Computer braucht nicht alle Daten in seinem Gedächtnis, dem Zentralspeicher, aufzubewahren. Auch der Computer kann sich externer Datenspeicher bedienen. Er kann Daten z.B. über Magnetbandstationen auf Magnetbänder abgeben, um sie im Bedarfsfall von dort wieder abzurufen. Auch Magnetplatten sind Datenträger für externe Datenspeicherung, die entsprechenden Maschineneinheiten sind Magnetplattenstationen. Hat ein Computer Daten auf Lochkarten oder Lochstreifen ausgegeben, so können diese Daten für eine neue Datenverarbeitungsaufgabe über Lochkarten- bzw. Lochstreifenleser als Eingabedaten wieder eingelesen werden. So gesehen, können auch Lochkarte und Lochstreifen als externe Datenspeicher verstanden werden.



Der Grundtyp der Lochkarte. Durch ein, zwei oder drei Lochungen je Spalte werden die in der EDV (Elektronische Datenverarbeitung) verwendeten Zeichen codiert.



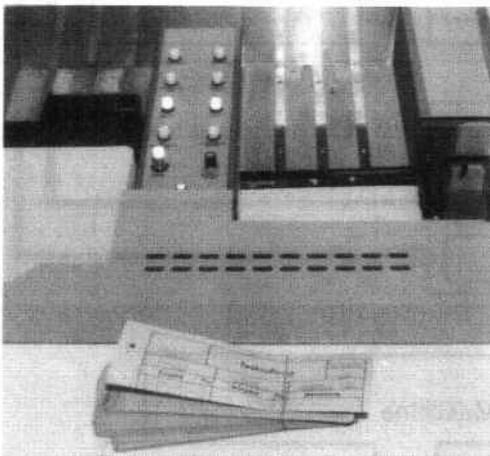
Ein Lochstreifen.



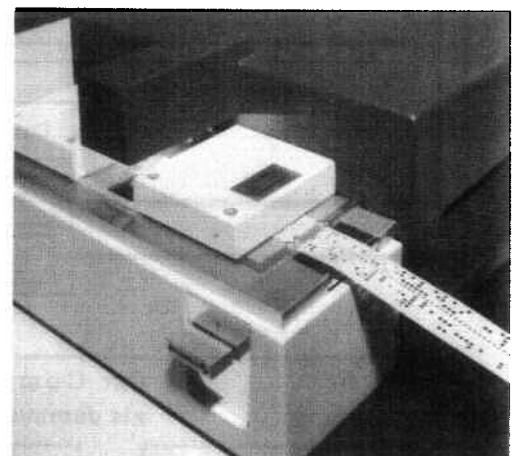
Frontseite Zentraleinheit
- das Gesicht des Computers



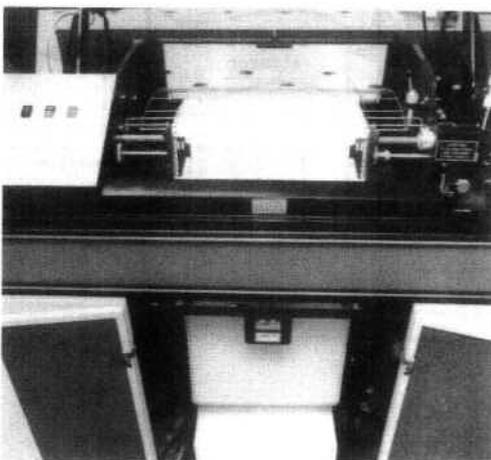
Kommunikation
Mensch - Maschine



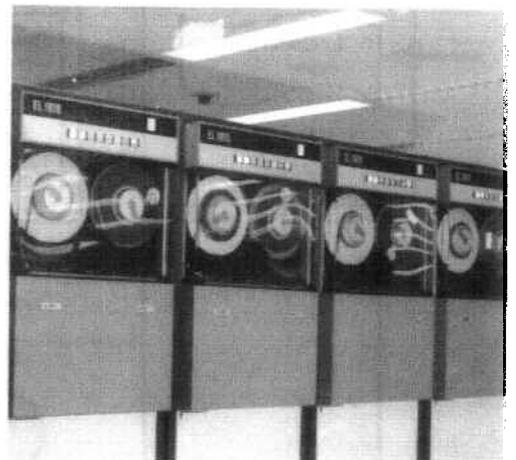
Lochkartenleser



Lochstreifenleser

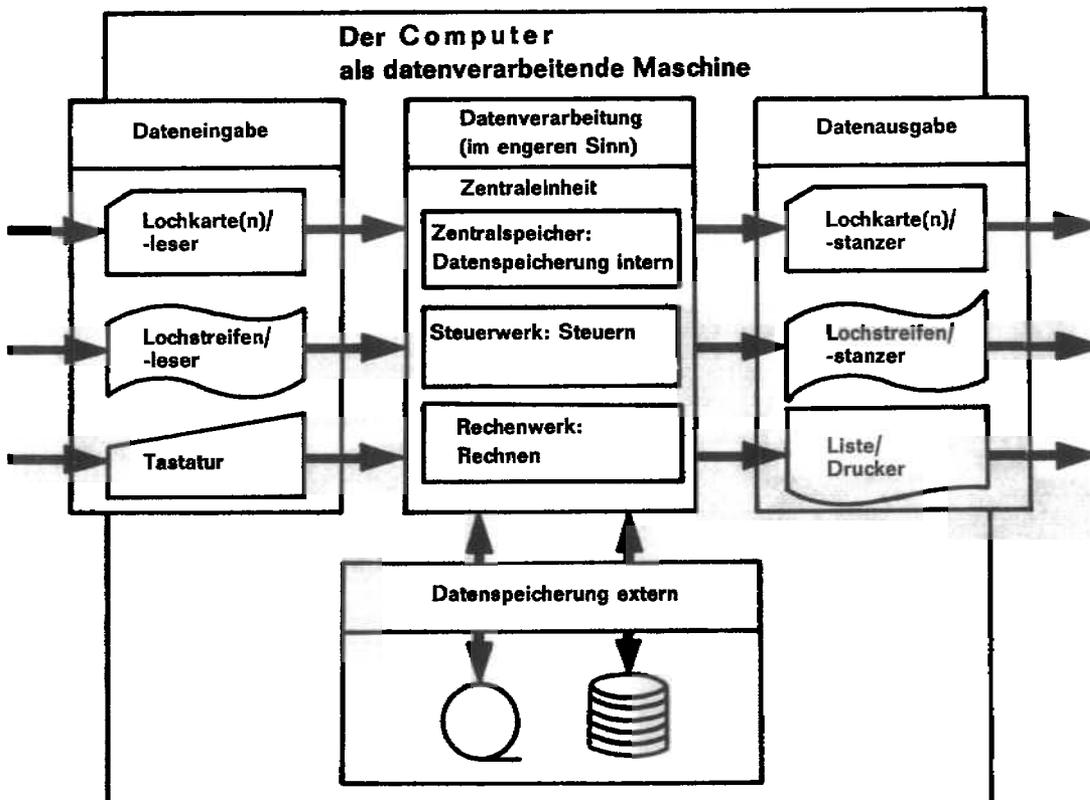
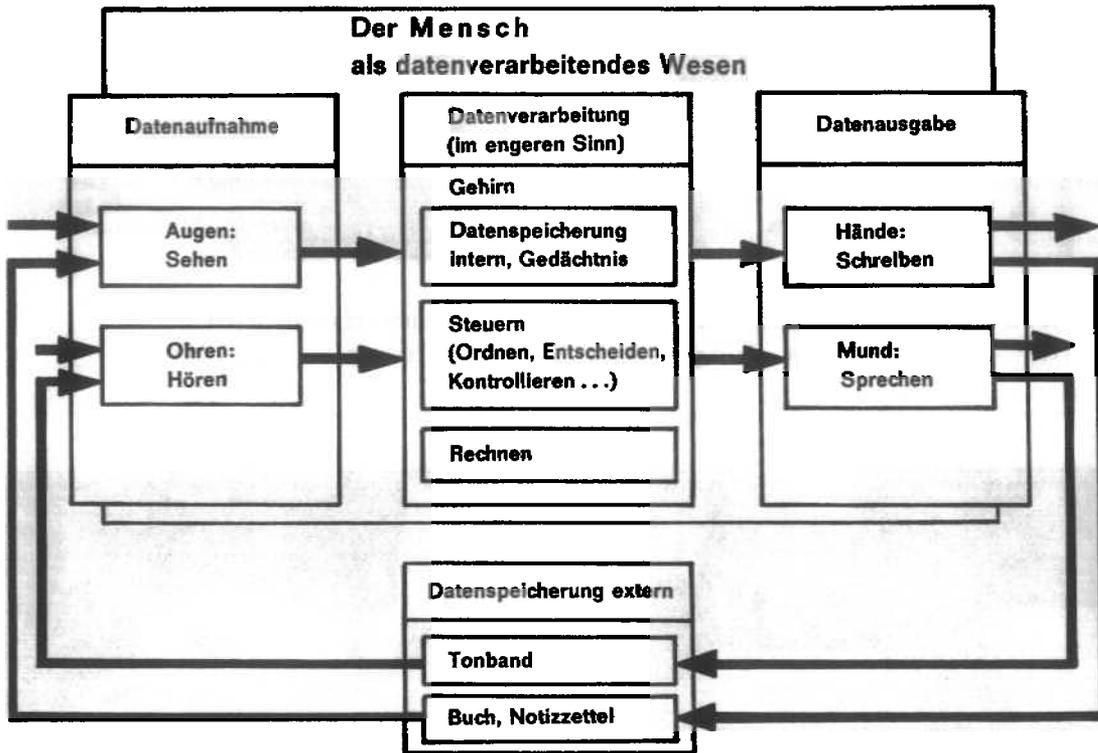


Ein Schnelldrucker



Magnetbandstation

Die beiden Abbildungen zeigen Ihnen noch einmal in schematischer Übersicht die Funktionen und Einheiten für Datenverarbeitung bei Mensch und Computer.



4.3. Das Programm

Übertragen wir nun einem Computer eine bestimmte Datenverarbeitungsaufgabe, beispielsweise die Lohnabrechnung eines Betriebes. Dazu benötigt der Computer Daten, wie die Namen der Lohnempfänger, Zahl der geleisteten Arbeitsstunden, Stundenlohn, Lohnsteuersätze usw. Diese Daten stellen wir auf Lochkarten zur Verfügung, und der Computer kann an die Arbeit gehen. Oder? - Nein, so einfach geht es nun doch nicht, denn noch "weiß" der Computer gar nicht, was er mit den Daten anfangen soll; er "weiß" ja nicht, wie eine Lohnabrechnung vor sich geht; er "weiß" sogar nicht einmal, daß er eine Lohnabrechnung machen soll. Daten allein genügen also nicht. Der Computer braucht ein Programm: Ein Programm, das ihm in allen Einzelheiten Anweisungen gibt, wie er mit welchen Daten verfahren soll. Bevor der Computer das Programm nicht kennt, kann er die Aufgabe nicht durchführen.

Und wie ist das beim Menschen? - Auch der Mensch braucht bestimmte Programme, um Datenverarbeitung durchführen zu können. Wenn wir die Lohnabrechnung z.B. durch einen Buchhalter vornehmen lassen, so ist es der Buchhalter, der nach einem Programm arbeitet, nämlich nach einem Lohnabrechnungsprogramm, das er kennt, das er also im Gedächtnis hat.

So wie der Mensch sein Programm, nach dem er eine Datenverarbeitungsaufgabe durchführen will, im Gedächtnis hat, muß auch der Computer sein Programm im Gedächtnis, also im Zentralspeicher, haben. Natürlich muß das Programm erst entworfen, dann z. B. auf Lochkarten übertragen und schließlich über den Lochkartenleser eingelesen werden - in den Zentralspeicher. Dann kann der Computer mit der Datenverarbeitung beginnen.

Sie wissen nun - in groben Zügen - was Datenverarbeitung ist. Sie kennen den Menschen als datenverarbeitendes Wesen, und Sie haben einen Eindruck vom Computer als datenverarbeitende Maschine. Beide, Mensch und Computer, haben Ähnlichkeit miteinander. Beide brauchen zur Datenverarbeitung Daten und Programme. - Gibt es überhaupt noch einen wesentlichen Unterschied zwischen Mensch und Computer? Kann ein Computer etwa denken? Oder ist der Mensch nichts weiter als ein Computer?

Ehe wir diese schwierige Frage beantworten können, müssen wir uns darüber im klaren sein, was wir unter "Denken" verstehen wollen. Wir können die Frage kurz beantworten, ohne in philosophische Schwierigkeiten zu geraten. Wenn wir die Fähigkeit, Verknüpfungen im Sinne der Datenverarbeitung

durchzuführen, als "Denken" bezeichnen wollen, so können wir jedem komplexen Datenverarbeitungssystem (Mensch, Tier und Computer) die Fähigkeit zu denken zugestehen. Dabei abstrahieren wir allerdings vom Bewußtsein. Bewußtsein aber als die Tatsache, von sich selbst zu wissen, zu wollen, zu empfinden und einen Spielraum zu freier Entscheidung zu haben, kann kein Computer für sich in Anspruch nehmen. Auch nicht das gigantischste künstliche Supergehirn der Zukunft. Wenn wir also Denken als bewußtes Denken, als Bewußtsein von sich selbst und als Fähigkeit zu empfinden und zu wollen, auffassen, kann kein Computer denken. Und ganz abgesehen davon: Ihr eigenes Gehirn ist allein rein schaltungstechnisch komplizierter als sämtliche heute in der Welt arbeitenden Computer zusammen.

4.4. Der Philips Computer-Lehrbaukasten CL 1601

Hier liegen im Prinzip die gleichen Zusammenhänge vor wie beim großen Computer. Bei Ihrem CL 1601 erfolgt die Dateneingabe über die sechs Eingabeschalter. Die Dateneingabe erfolgt direkt, d.h. die Daten werden nicht auf irgendwelchen Belegen eingegeben.

Die Datenausgabe erfolgt über die Lampen der Logik-Bausteine. Diese Art der Datenausgabe ist vergleichbar mit Datenausgabe durch optische Anzeige (ebenfalls Lampen oder Bildschirm) bei großen Computern.

Zwischen Dateneingabe und Datenausgabe liegt die eigentliche Datenverarbeitung. Sie erfolgt beim CL 1601, wie bei den großen Computern, nach der entsprechenden Programmierung in den IC's (Integrierten Schaltkreisen) der Logik-Bausteine. Einen Teil der Steuerwerk- und Rechenwerk-Funktionen können Sie selbst durch entsprechende Schaltvorgänge an den Eingabeschaltern übernehmen.

Schließlich bliebe noch die externe Datenspeicherung zu erwähnen. Selbstverständlich kann die externe Datenspeicherung bei dem CL 1601 nicht so komfortabel vor sich gehen wie bei großen Computern. Beim CL 1601 übernehmen Sie selbst die externe Datenspeicherung, wenn Sie z.B. durch die Anzeigelampen ausgegebene Ergebnisse auf einem Blatt Papier notieren, also extern speichern, um sie für einen neuen Datenverarbeitungsvorgang wieder über die Eingabeschalter einzugeben.

Ähnlich wie ein großer Computer, so braucht auch Ihr CL 1601 sein Programm. Das Programm geben Sie ein, indem Sie die entsprechende Verdrahtung vornehmen. Die Programmierung der Logik-Bausteine des CL 1601 legt jeweils bestimmte logische Beziehungen fest, die eben so lange bestehen

bleiben wie die Verdrahtung beibehalten wird. Ein Programm für einen großen Computer kann und muß selbstverständlich mehr vollbringen: Das Programm gibt dem Computer eine Abfolge einzelner Befehle - das können mehrere hundert oder tausend sein - nach denen der Computer vollautomatisch die anstehenden Daten verarbeitet. Mit dem CL 1601 können Sie solche umfangreichen Aufgaben nicht ausführen, aber Sie lernen im Spiel und im Experiment Computer-Logik und Computer-Arithmetik kennen.



5. Logische Funktionen mit zwei Eingängen

5.1. Das Mantel-Problem

An einem alltäglichen Entscheidungsproblem soll einmal gezeigt werden, wie viele Möglichkeiten es gibt, Elemente einer logischen Funktion miteinander zu verbinden. Das "Problem" ist folgendes: Sie möchten spazieren gehen und nach Möglichkeit den Mantel zu Hause lassen. Allerdings wollen Sie weder frieren noch naß werden. Sie machen also Ihre Entscheidung, den Mantel zu Hause zu lassen, von zwei Bedingungen abhängig: Die Wettervorhersage muß gut sein, und es muß wärmer als 10° sein. Mit anderen Worten:

Wenn die Wettervorhersage gut ist und wenn die Temperatur höher als 10° C ist, dann lasse ich meinen Mantel zu Hause.

Bezeichnen wir die Aussage "Die Wettervorhersage ist gut" mit A und die Aussage "Die Temperatur ist höher als 10° C" mit B und die Schlußfolgerung "Ich lasse meinen Mantel zu Hause" mit F. Dann läßt sich die Aussage abgekürzt so schreiben:

Wenn A und wenn B, dann F.

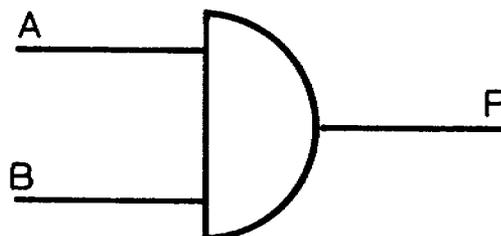
Oder noch kürzer: Wenn A und B, dann F.

Sie erkennen, daß es sich hier um eine UND-Funktion handelt. Hier die Funktionstabelle - einmal ausführlich mit Text, einmal in Kurzform - und das Schaltsymbol:

A B F

Wettervorhersage gut?	Temperatur über 10° C?	Mantel zu Hause lassen?
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



Mit einem Logik-Baustein läßt sich die UND-Funktion so programmieren:

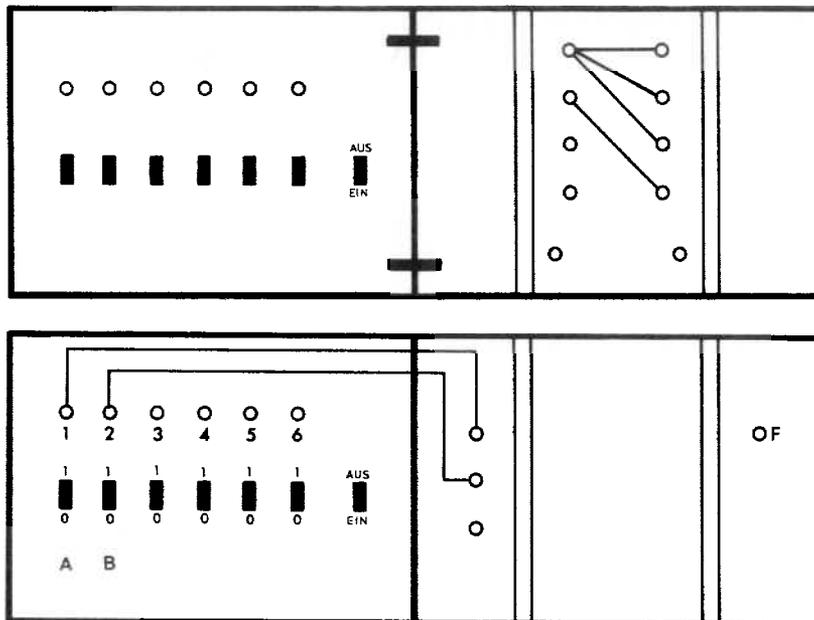


Abb. 35

Formulieren wir die Zusammenhänge jetzt einmal anders: Wenn das Wetter nicht gut ist oder wenn die Temperatur nicht über 10° C ist, dann lasse ich meinen Mantel nicht zu Hause.

Im Grunde genommen haben wir die gleichen Zusammenhänge, nur haben wir die einzelnen Elemente der Aussage umgekehrt (negiert) und erhalten damit eine ODER-Funktion. Die Umkehrung bzw. die Verneinung der einzelnen Aussageelemente wollen wir folgendermaßen kennzeichnen:

- Wettervorhersage nicht gut: \bar{A}
- Temperatur nicht über 10° C: \bar{B}
- Ich lasse den Mantel nicht zu Hause: \bar{F}

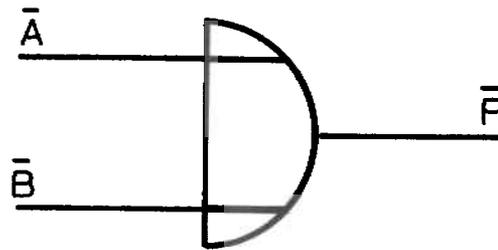
Der Querstrich über den Eingangs- bzw. Ausgangs-Symbolen deutet an, daß das Signal negiert (verneint, umgekehrt) ist. Man liest z.B. $\bar{A} = A$ quer.

Die Funktionstabellen und das Schaltsymbol:

$\bar{A}, \bar{B}, \bar{F}$

Wettervorhersage nicht gut?	Temperatur nicht über 10° C?	Mantel nicht zu Hause lassen?
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

\bar{A}	\bar{B}	\bar{F}
0	0	0
0	1	1
1	0	1
1	1	1



Um keine Mißverständnisse aufkommen zu lassen, müssen wir noch erwähnen, wie 0 und 1 bzw. im Falle von Fragestellungen NEIN und JA hier zu verstehen sind. Wenn wir z.B. fragen: "Ist die Wettervorhersage nicht gut?" und wir antworten darauf JA bzw. 1, so soll das heißen: "Ja, es stimmt, die Wettervorhersage ist nicht gut". Beantworten wir dagegen die Frage mit NEIN oder 0, dann heißt das: "Nein, es stimmt nicht, die Wettervorhersage ist doch gut." Die dazugehörige Programmierung sieht so aus:

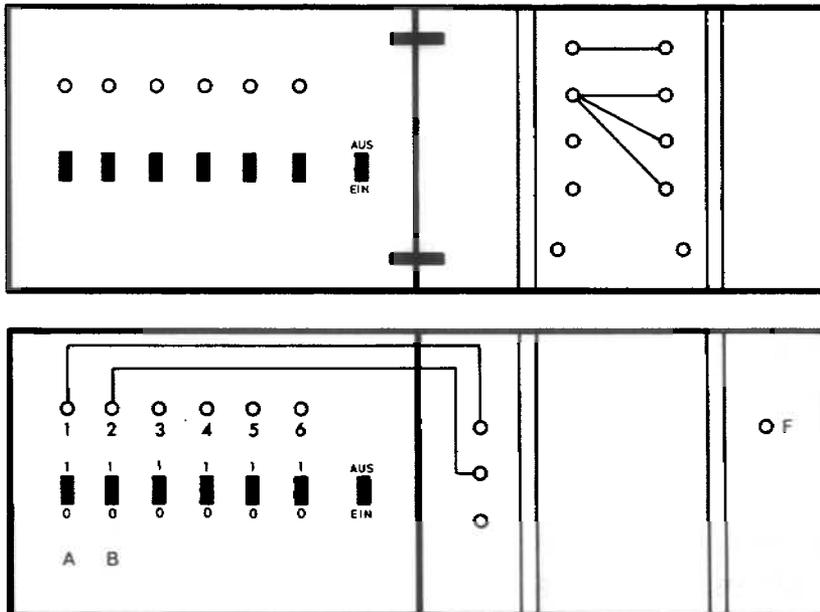


Abb.36

Wir haben einen wichtigen Grundsatz der Logik kennengelernt: Eine UND-Funktion entspricht einer ODER-Funktion, wenn die einzelnen Elemente der UND-Funktion negiert, also verneint werden.

Bevor wir weitere Kombinationsmöglichkeiten zu diesem Beispiel durchdenken, noch einige neue Begriffe:

Eine logische Funktion hat als Elemente Bedingungen und ein Ereignis. Im Falle unseres Mantel-Problems sind die Bedingungen die gute Wettervorhersage und die Mindesttemperatur; das Ereignis ist, ob ich meinen Mantel mitnehme.

Wenn wir von der sprachlichen Ausdrucksmöglichkeit für eine logische Funktion absehen und uns eine Schaltung bzw. das entsprechende Schalt-

symbol vornehmen, dann sprechen wir nicht von Bedingungen und Ereignis, sondern von Eingängen und Ausgang. Eingänge und Ausgang sind veränderliche, also variable Größen. Sie können die Werte 0 und 1 haben. Eine veränderliche Größe wird kurz auch eine Variable genannt. Was wir eben als Bedingungen bezeichnet haben, können wir nunmehr als Eingangsvariable bezeichnen; was wir als Ereignis gekennzeichnet haben, stellt eine Ausgangsvariable dar. Diese Ausdrucksweise können wir nun ein für allemal verwenden, ganz gleich ob es sich bei einer logischen Funktion um eine sprachliche Darstellung oder um eine elektrische oder elektronische Schaltung handelt.

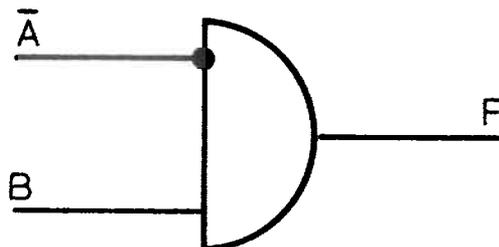
Die vorhin gewonnene Erkenntnis über den Zusammenhang zwischen UND-Funktion und ODER-Funktion drücken wir mit den neuen Bezeichnungen nun so aus: Eine UND-Funktion entspricht einer ODER-Funktion der negierten Variablen.

Das klingt reichlich abstrakt, läßt sich aber insbesondere im Zusammenhang mit unserem konkreten Beispiel des Mantel-Problems durchaus begreifen. Die Eingangsvariablen und die Ausgangsvariable unseres Mantelproblems und deren Negationen (Verneinungen, Umkehrungen) lassen sich beliebig miteinander kombinieren. Es gibt insgesamt acht Kombinationsmöglichkeiten, von denen wir zwei schon kennengelernt haben, nämlich: A, B, F und \bar{A} , \bar{B} , \bar{F} . Streifen wir kurz auch noch die anderen Möglichkeiten:

\bar{A} , B, F

Wettervorhersage nicht gut?	Temperatur über 10° C?	Mantel zu Hause lassen?
A	B	F
0	0	0
0	1	1
1	0	0
1	1	0

A	B	F
0	0	0
0	1	1
1	0	0
1	1	0



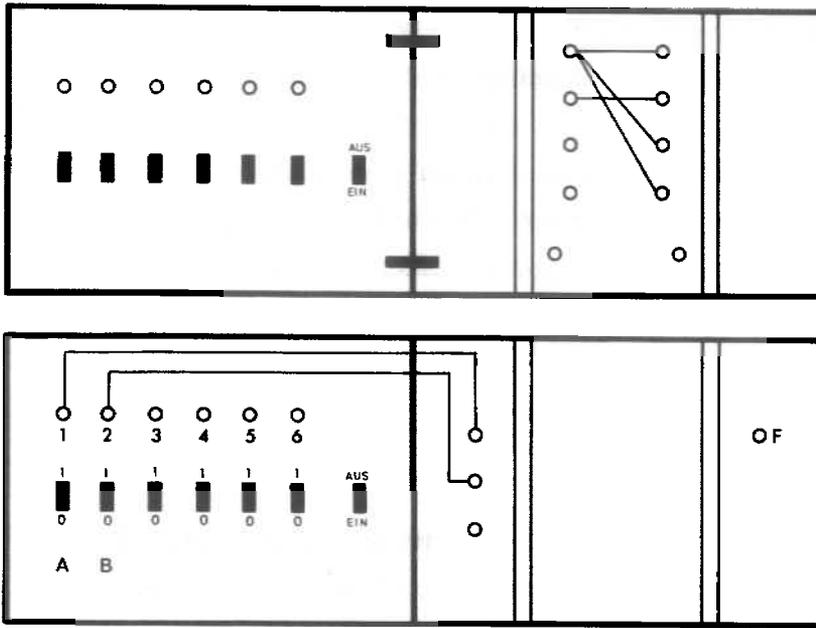


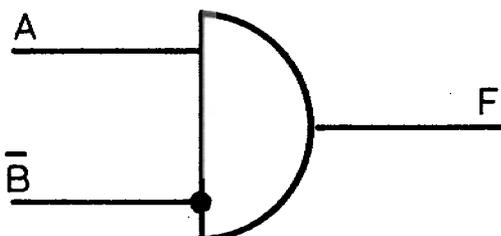
Abb.37

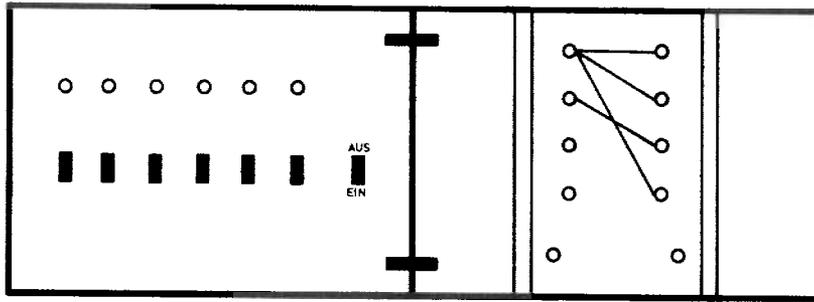
Es handelt sich hier um eine UND-Funktion mit einem negierten Eingang.

A, \bar{B} , F

Wettervorhersage gut?	Temperatur nicht über 10°C?	Mantel zu Hause lassen?
A	\bar{B}	F
0	0	0
0	1	0
1	0	1
1	1	0

A	\bar{B}	F
0	0	0
0	1	0
1	0	1
1	1	0





Auch dies ist eine UND-Funktion mit einem negierten Eingang.

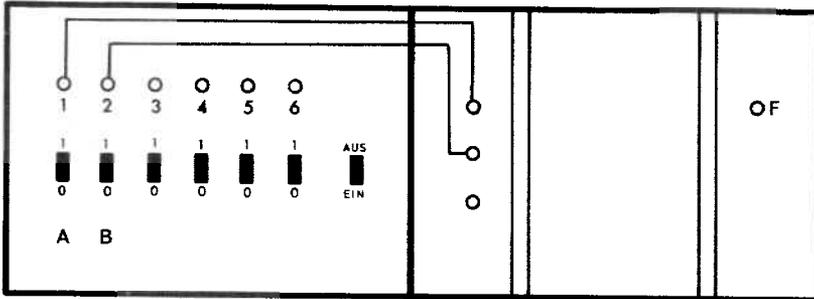


Abb. 38

A, B, \bar{F}

Wettervorhersage gut?	Temperatur über 10°C?	Mantel nicht zu Hause lassen?
A	B	\bar{F}
0	0	1
0	1	1
1	0	1
1	1	0

A	B	\bar{F}
0	0	1
0	1	1
1	0	1
1	1	0

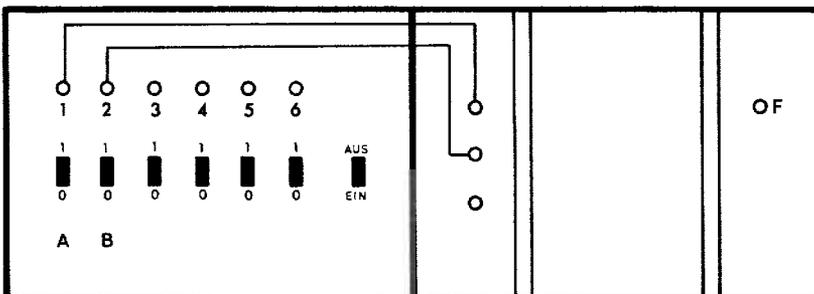
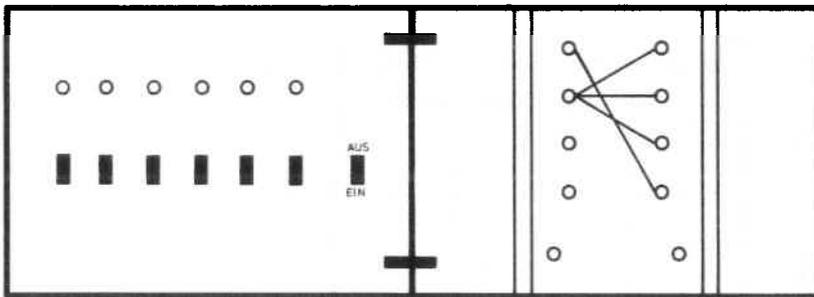
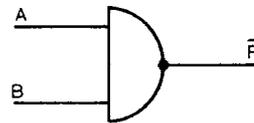


Abb. 39

Diese logische Funktion hat den eigenartigen Namen NAND-Funktion. NAND ist eine Abkürzung von "not and" (englisch), was zu deutsch "nicht und" heißt. Eine NAND-Funktion kann man also als eine UND-Funktion mit negiertem Ausgang betrachten.

A, \bar{B} , \bar{F}

Wettervorhersage gut?	Temperatur nicht über 10° C?	Mantel nicht zu Hause lassen?
A	\bar{B}	\bar{F}
0	0	1
0	1	1
1	0	0
1	1	1

A	\bar{B}	\bar{F}
0	0	1
0	1	1
1	0	0
1	1	1

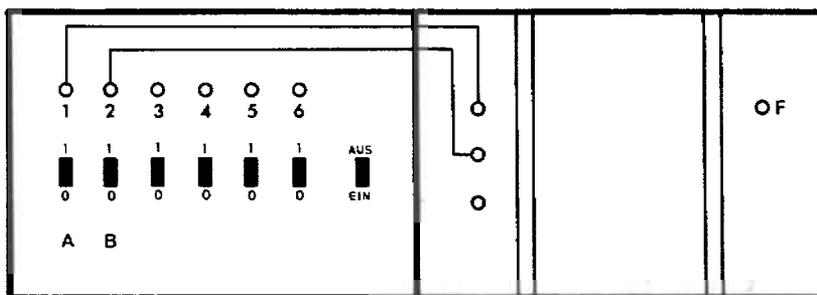
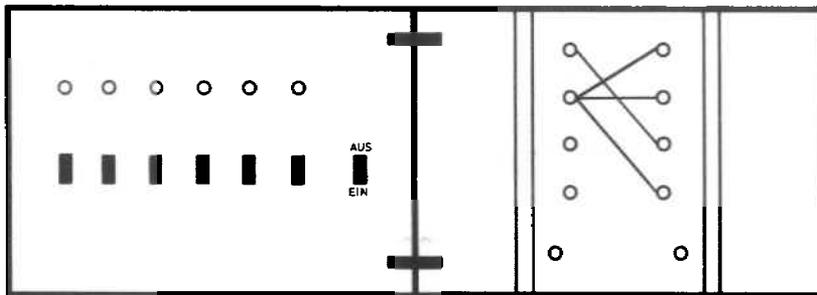
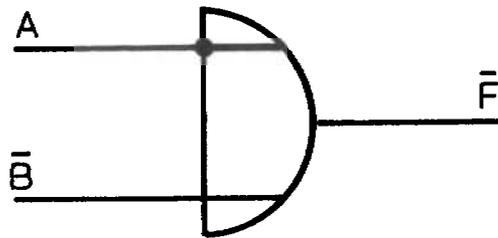


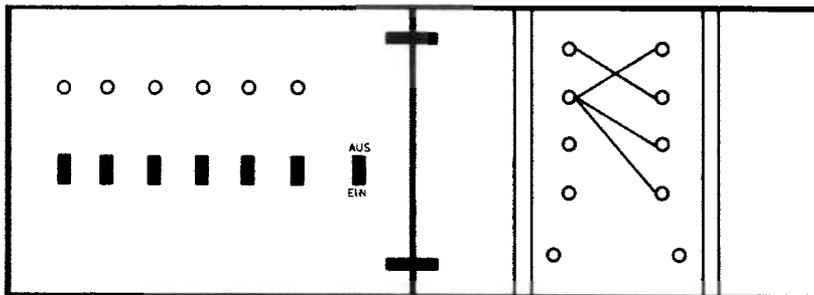
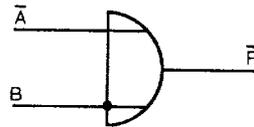
Abb. 40

Hier haben wir es mit einer ODER-Funktion mit einem negierten Eingang zu tun.

\bar{A}, B, \bar{F}

Wettervorhersage nicht gut?	Temperatur über 10° C?	Mantel nicht zu Hause lassen?
\bar{A}	B	\bar{F}
0	0	1
0	1	0
1	0	1
1	1	1

\bar{A}	B	\bar{F}
0	0	1
0	1	0
1	0	1
1	1	1



Dies ist ebenfalls eine ODER-Funktion mit einem negierten Eingang.

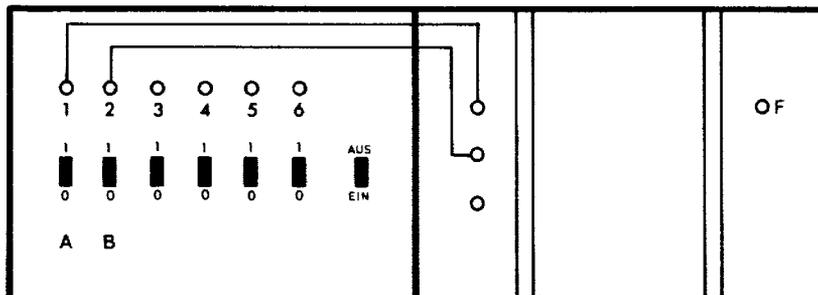
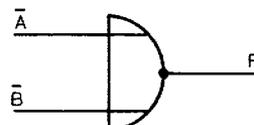


Abb. 41

\bar{A}, \bar{B}, F

Wettervorhersage nicht gut?	Temperatur nicht über 10° C?	Mantel zu Hause lassen?
\bar{A}	\bar{B}	F
0	0	1
0	1	0
1	0	0
1	1	0

\bar{A}	\bar{B}	F
0	0	1
0	1	0
1	0	0
1	1	0



In diesem Sinne ist auch der Aufbau der Logik-Bausteine Ihres CL 1601 zu verstehen. Wenn Sie sich jetzt einmal den Schaltplan eines Logik-Bausteins im Kapitel ansehen, so erkennen Sie, daß sehr viele Negationen eingebaut sind. Machen Sie sich nichts daraus, wenn Sie den Schaltplan nicht ganz durchschauen. Nur so viel sei gesagt: Die Logik-Bausteine des CL 1601 sind so konstruiert - bei Verwendung von standardisierten integrierten Schaltkreisen, die auch im Groß-Computer Verwendung finden - daß möglichst viele Programme zu verwirklichen sind.

Durch Negieren einer oder mehrerer der Variablen wurde aus unserer UND-Funktion eine UND-Funktion mit einem negierten Eingang, eine NAND-Funktion, eine ODER-Funktion mit einem negierten Eingang sowie eine NOR-Funktion. Von solchen Umformungen macht man in der Computer-Technik häufig Gebrauch, wenn man mit wenig logischen Verknüpfungselementen möglichst vielfältige Schaltungen verwirklichen will.

Dies ist das Gegenstück zur NAND-Funktion: die NOR-Funktion. NOR ist die Abkürzung von "not or" (englisch), was zu deutsch "nicht oder" heißt. Eine NOR-Funktion ist also eine ODER-Funktion mit negiertem Ausgang.

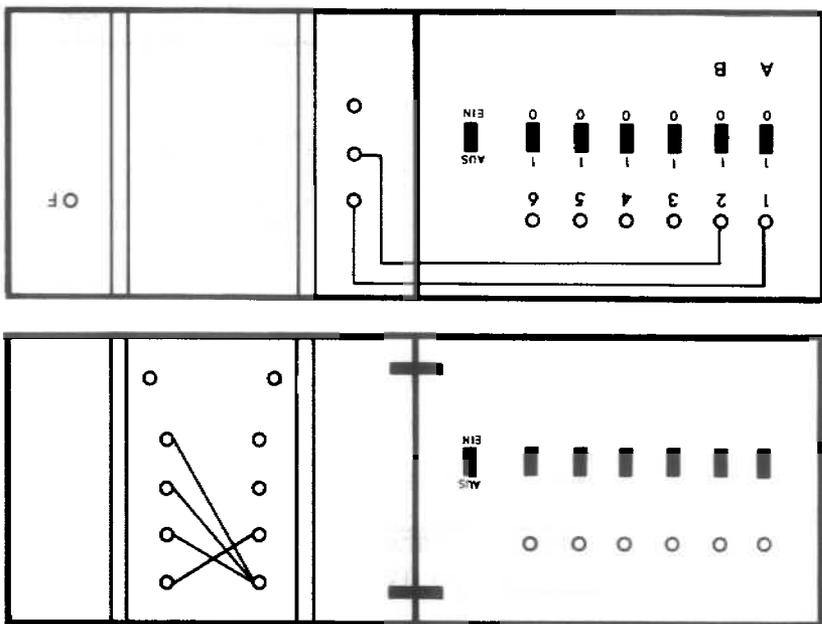


Abb. 42

6. Logische Funktionen mit 3 Eingängen

Wir haben bisher logische Funktionen und logische Schaltungen mit zwei Eingängen kennengelernt:

UND-Funktion	bzw.	UND-Schaltung
ODER-Funktion	"	ODER-Schaltung
Exklusiv-ODER-Funktion	"	Exklusiv-ODER-Funktion
Äquivalenz-Funktion	"	Äquivalenz-Schaltung
NAND-Funktion	"	NAND-Schaltung
NOR-Funktion	"	NOR-Schaltung

Die Identitäts-Funktion bzw. Identitäts-Schaltung sowie die Negations-Funktion bzw. Negations-Schaltung hatten jeweils nur einen Eingang.

6.1. UND-Funktion

Die genannten Funktionen bzw. Schaltungen können - mit Ausnahme der Identitäts- und Negations-Funktion - auch mit mehr als zwei Eingängen vorkommen. Im folgenden geben wir Ihnen eine Übersicht über logische Funktionen mit drei Eingängen.

Selbstverständlich sind auch hierzu sprachliche Beispiele denkbar, wie z.B. für die UND-Funktion mit 3 Eingängen: Wenn ich durstig bin und wenn ich Geld bei mir habe und wenn ich an einem Wirtshaus vorbeikomme, dann trinke ich Bier.

Die Ausgangsvariable F kann also nur den Wert 1 annehmen, wenn alle drei Eingangsvariablen ebenfalls den Wert 1 haben. Auf unser Beispiel übertragen bedeutet das, daß ich nur Bier trinken kann ($F = 1$), wenn ich durstig bin ($A = 1$) und Geld besitze ($B = 1$) und an einem Wirtshaus vorbeikomme ($C = 1$). Nimmt nur eine der Eingangsvariablen den Wert 0 an, so muß die Ausgangsvariable F ebenfalls 0 sein.

Bei drei variablen Eingängen nimmt die Zahl der Möglichkeiten zu, und die Funktionstabelle muß diese größere Zahl auch ausdrücken.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Programmieren Sie Ihren Logik-Baustein und verbinden Sie ihn wie auf der folgenden Abbildung mit der Eingabeeinheit.

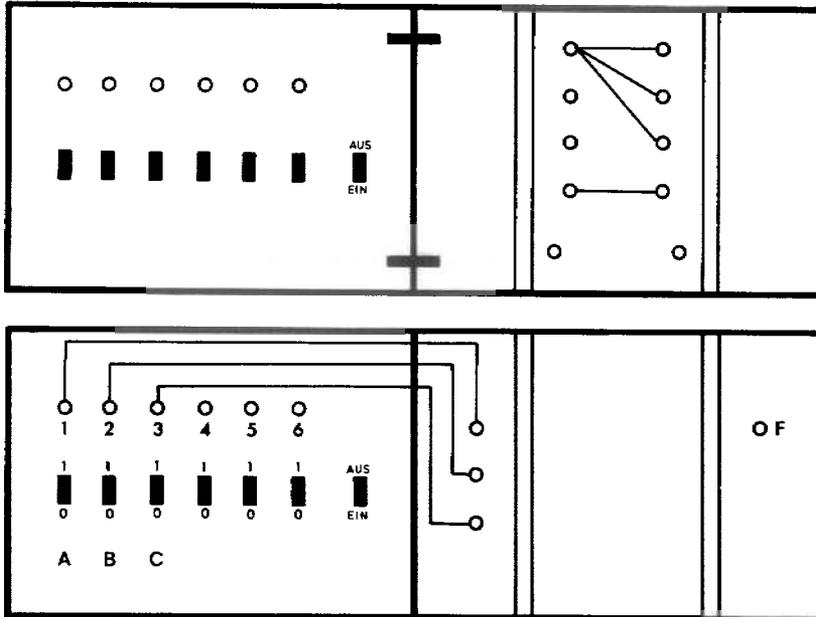
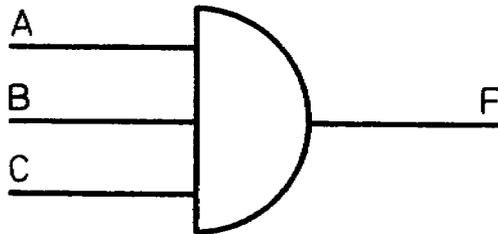


Abb.43

Probieren Sie alle in der Funktionstabelle angegebenen Möglichkeiten und tragen Sie die angezeigten Werte ein.

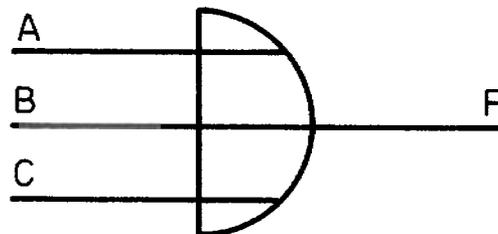
- Schalter A — Variable A
- Schalter B — Variable B
- Schalter C — Variable C

Auch für eine UND-Funktion mit drei Eingängen gibt es ein Schaltsymbol:



6.2. Einschließende ODER-Funktion

Symbol ODER-Funktion:



Sie möchten einem Freund eine kurze schriftliche Mitteilung zukommen lassen. Sie zücken also Ihren Federhalter und suchen Schreibpapier. Haben Sie eine einfache Postkarte oder eine Ansichtskarte oder Briefpapier gefunden, so können Sie die Mitteilung zu Papier bringen. Das

ist natürlich auch möglich, wenn Sie z.B. Briefpapier und eine Ansichtskarte zur Hand haben. Ist dagegen nichts greifbar, dann müssen Sie sich erst Papier beschaffen.

Für die Funktionstabelle bedeutet das: Hat mindestens eine Eingangsvariable den Wert 1, dann ist F auch 1. Versuchen Sie doch wieder einmal, die Funktionstabelle für dieses Beispiel aufzustellen. Dazu noch ein Hinweis: Bei drei Eingangsvariablen existieren immer acht mögliche Kombinationen.

A	B	C	F

Wenn Sie nach der folgenden Abbildung verdrahten, können Sie überprüfen, ob die Funktionstabelle richtig aufgestellt wurde.

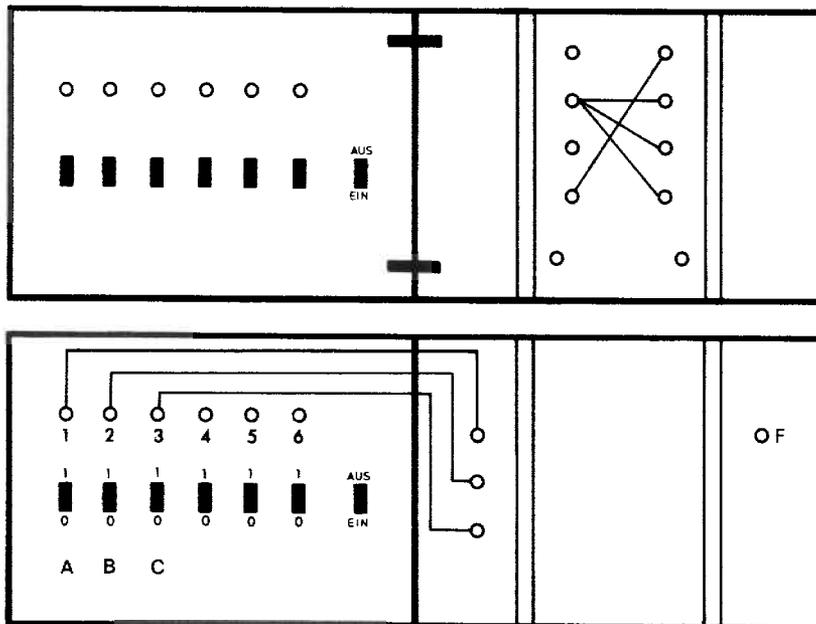
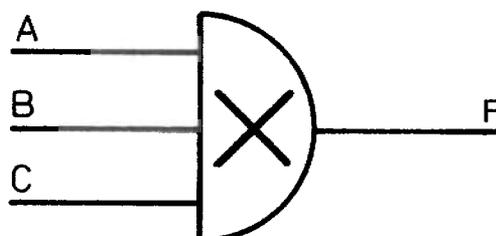


Abb. 44

6.3. Exklusiv-ODER-Funktion

Symbol Exklusiv-ODER-Funktion:



Als Bastler stehen Sie vor dem Problem, ein Gehäuse für ein elektronisches Gerät herzustellen. Drei verschiedene Materialien stehen Ihnen dafür zur Verfügung: Holz, Metall oder Kunststoff.

Zu diesem Problem soll zuerst Ihr Computer in Betrieb gesetzt werden, und anschließend wenden wir uns der sprachlichen Lösung zu.

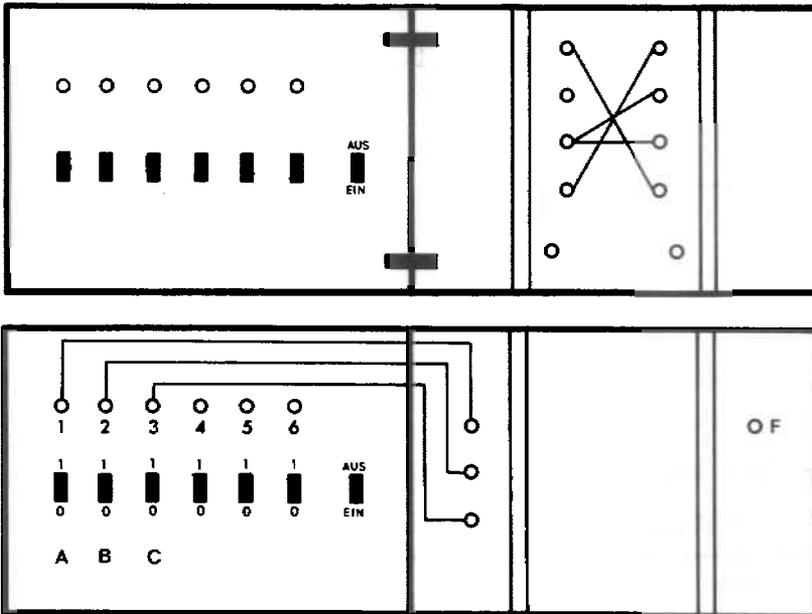


Abb. 45

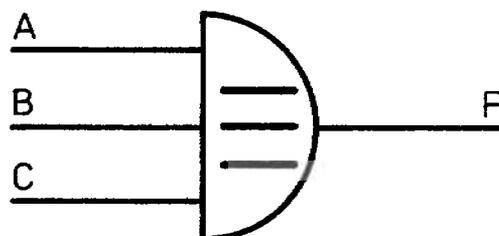
Tragen Sie bitte auch die Werte in die Funktionstabelle ein.

A	B	C	F

Die Auswertung der Funktionstabelle ist nun sprachlich kein Problem mehr. Nur wenn höchstens eine der Eingangsvariablen den Wert 1 erhält, ist auch der Ausgang 1. Für Sie als Bastler bedeutet das, Sie bekommen Ihr Gehäuse nur dann fertiggestellt, wenn Sie entweder Holz oder Metall oder Kunststoff verwenden. Zwei oder drei Materialien können Sie nicht verwenden.

6.4. Äquivalenz-Funktion

Symbol Äquivalenz-Funktion:



"Alles oder nichts" - das ist die einfachste Erklärung für eine Äquivalenz-Funktion mit drei Eingängen.

Wie Sie der Funktionstabelle entnehmen und mit Ihrem Computer überprüfen können, ist der Ausgang F nur dann 1, wenn alle Eingangsvariablen den Wert 0 oder 1 besitzen. Bei den übrigen sechs Möglichkeiten ist der Ausgang 0.

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

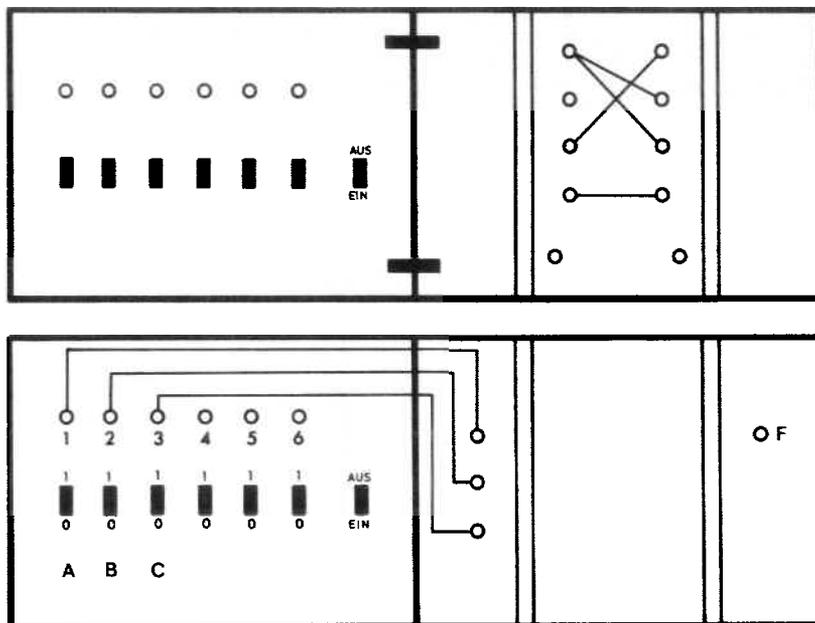
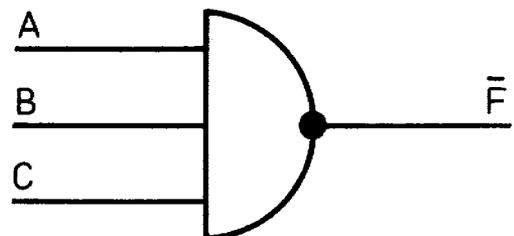


Abb. 46



6.5. NAND-Funktion

Symbol NAND-Funktion:

Den Besuch eines großen internationalen Fußballspiels machen Sie von drei Bedingungen abhängig:

1. Das Wetter ist gut (A).
2. Ein Freund kommt mit (B).
3. Es sind noch Eintrittskarten vorhanden (C).

Sie nehmen sich fest vor, nicht hinzugehen, wenn nicht alle drei Bedingungen erfüllt sind.

Bevor Sie die Funktionstabelle betrachten und überprüfen, überlegen Sie einmal die Bedeutung der Werte 0 und 1: Ist eine Bedingung erfüllt, gilt 1, ist sie nicht erfüllt, bedeutet das 0. Bei der Ausgangsvariablen F ist das umgekehrt: Nicht zum Fußball gehen, bedeutet 1, hingehen erhält den Wert 0 - heißt also JA. Denn die Frage wurde in der Verneinungsform gestellt, und ein NEIN (0) als Antwort darauf - doppelte Verneinung - heißt JA.

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Überprüfen Sie, ob Ihr Computer zu demselben Ergebnis gelangt.

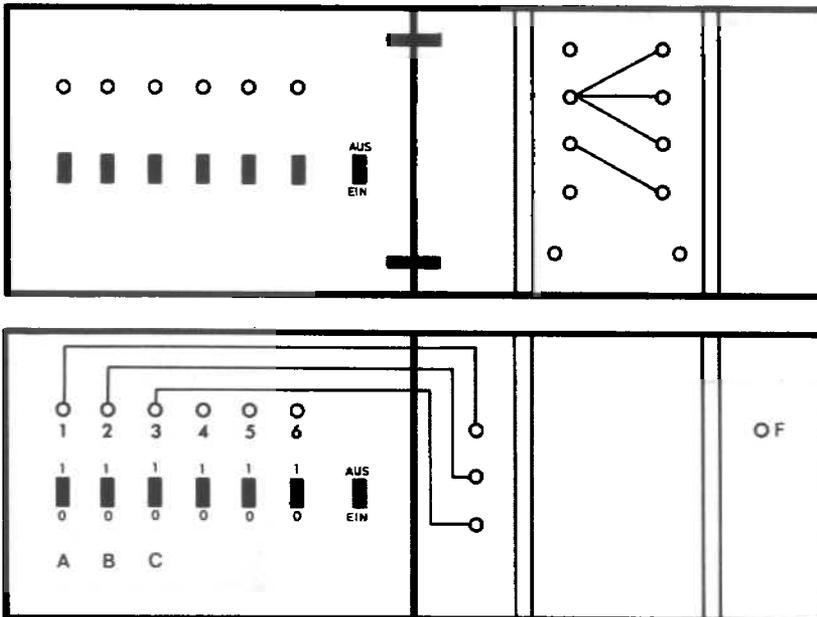
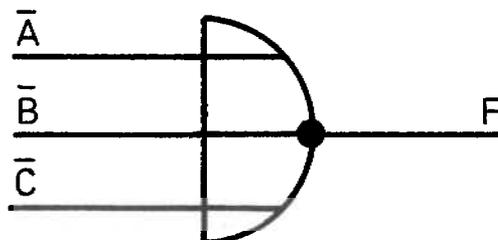


Abb. 47

6.6. NOR-Funktion

Symbol NOR-Funktion:



Vielleicht mögen Sie gern mit einem kleinen Boot fahren, vielleicht auch nicht. Nehmen wir einmal an, das letztere sei der Fall, und Sie haben eine deutliche Abneigung gegen Bootsfahrten. Von einem Freund, der das nicht weiß, werden Sie dazu eingeladen für das nächste Wochenende. Sie machen eine Zusage von drei Bedingungen abhängig:

1. Es darf nicht regnen (\bar{A}).
2. Es darf nicht windig sein (\bar{B}).
3. Sie wollen nicht weit vom Ufer entfernt fahren (\bar{C}).

Wenn alle negierten Bedingungen mit NEIN erfüllt sind (0) - dadurch hebt sich die doppelte Verneinung wieder auf - sagen Sie JA zur Bootsfahrt (1).

\bar{A}	\bar{B}	\bar{C}	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Die Funktionstabelle macht es deutlich: Wenn es nicht regnet oder nicht stürmt oder nicht zu weit vom Ufer entfernt gefahren wird, dann erteilen Sie eine Zusage.

Nach der folgenden Schaltung können Sie die Tabelle überprüfen. Die Lampe leuchtet nur ($F = 1$), wenn \bar{A} oder \bar{B} oder \bar{C} nicht 1 sind.

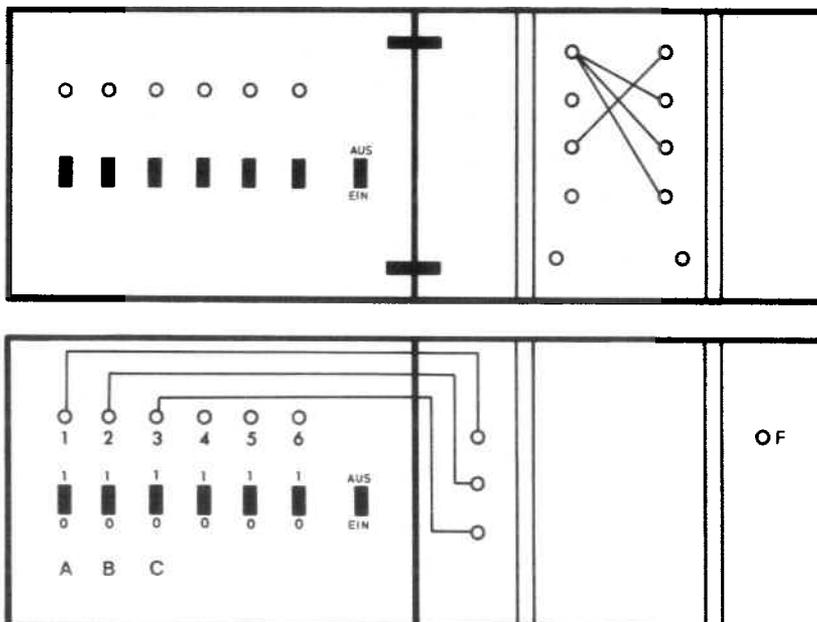


Abb. 48

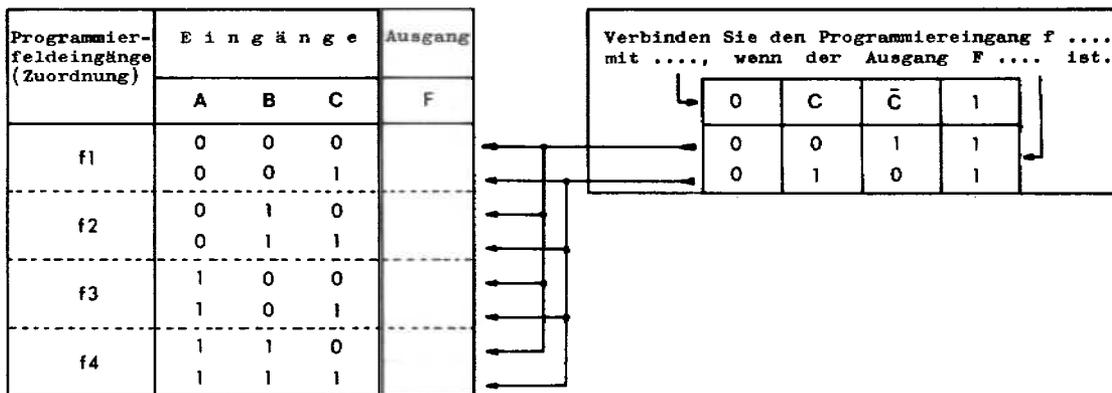
7. Programmierung der Logik-Bausteine

Sie haben eine Menge logischer Funktionen kennengelernt. Nun ist es an der Zeit, daß wir Ihnen das Schema erklären, nach dem Sie die Programmierung der Logik-Bausteine, also die Verdrahtung des Programmierfeldes, selbst ableiten können. Dazu wollen wir die einzelnen Buchsen des Programmierfeldes mit Bezeichnungen versehen, wie in der folgenden Abbildung aufgeführt:

	00	0f1	
A0	10	0f2	OF
B0	\bar{C} 0	0f3	
C0	C0	0f4	
	0	0	

Die Buchsen f1 bis f4 sind in bestimmter Weise - wie aus den folgenden Schemata zu ersehen ist - den Wertekombinationen der Eingangsvariablen von logischen Funktionen zugeordnet. Haben wir es beispielsweise mit einer logischen Funktion mit drei Eingangsvariablen zu tun, so ist die Buchse f1 den ersten beiden Zeilen der Funktionstabelle, also mit den Wertekombinationen 000 und 001, zugeordnet. Je nach dem, welche Ausgangswerte bei F entsprechend der logischen Funktion anliegen, ist die Buchse f1 mit der Buchse 0 bzw. C, \bar{C} oder 1 zu verbinden.

Programmierung mit den Eingangsvariablen A, B und C



Angenommen, Sie wollen das Wissen eines Bekannten testen und das Ergebnis mit einem Logik-Baustein anzeigen. Sie stellen die Frage: "Was hat Thomas A. Edison erfunden?" Ihr Bekannter kann jetzt eine lange Reihe von Erfindungen aufzählen, die Sie jedoch nicht mit dem Logik-Baustein

verarbeiten können. So stellen Sie Ihre allgemeine Frage etwas differenzierter und unterteilen sie:

1. Teilfrage: (A) "Ist Thomas A. Edison der Erfinder des Phonografen?"
2. " : (B) "Ist Thomas A. Edison der Erfinder des Tonbandgerätes?"
3. " : (C) "Ist Thomas A. Edison der Erfinder der Kohlefadenlampe?"

Auf diese Fragen kann nur mit JA oder NEIN geantwortet werden. Ferner stellen Sie die Forderung, daß nur bei 3 richtigen Antworten die Lampe des Logik-Bausteins aufleuchten soll. Jetzt müssen Sie Ihren Logik-Baustein programmieren.

Ordnen Sie zunächst die drei Teilfragen in der angegebenen Reihenfolge den Eingangsvariablen A, B und C zu, und legen Sie für die möglichen Antworten JA und NEIN fest, daß der betreffende Eingabeschalter auf 1 (JA) bzw. auf 0 (NEIN) zu stellen ist. Daraus ergeben sich folgende 8 verschiedene Möglichkeiten für die Stellung der 3 Eingabeschalter.

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Setzen Sie in die Spalte F überall dort eine 1 ein, wo nach der eingangs geforderten Bedingung die Lampe brennen soll.

Da die Teilfragen

- (A) "Ist Thomas A. Edison der Erfinder des Phonografen?" mit JA (1)
 (B) "Ist Thomas A. Edison der Erfinder des Tonbandgerätes?" mit NEIN(0)
 (C) "Ist Thomas A. Edison der Erfinder der Kohlefadenlampe?" mit JA (1)

richtig beantwortet sind, müssen Sie überall dort eine 1 für F setzen, wo alle Eingangswerte mit der Lösung übereinstimmen. Sind nur eine oder zwei Fragen richtig beantwortet, schreiben Sie in die Spalte F eine 0. Ihre vollständige Funktionstabelle muß nun so aussehen:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Aus dem Programmierschema für 3 Eingangsvariablen können Sie jetzt die Verdrahtung des Logik-Bausteins ableiten. Da die ersten beiden Zeilen dem Programmiereneingang f1 zugeordnet sind und F die Kombination 00 angenommen hat, ist f1 mit 0 zu verbinden. In den nächsten 2 Zeilen ist der Ausgang auch 00, also wird f2 wieder mit 0 verbunden. f3 (5. und 6. Zeile) ist mit C zu verbinden, da F in diesen beiden Zeilen die Werte 01 annimmt. Die Ausgangsfunktion der letzten beiden Möglichkeiten bestimmt der Programmiereneingang f4; da F 00 ist, muß f4 auch mit 0 verbunden werden. Und so sieht die Verdrahtung des Logik-Bausteins aus:

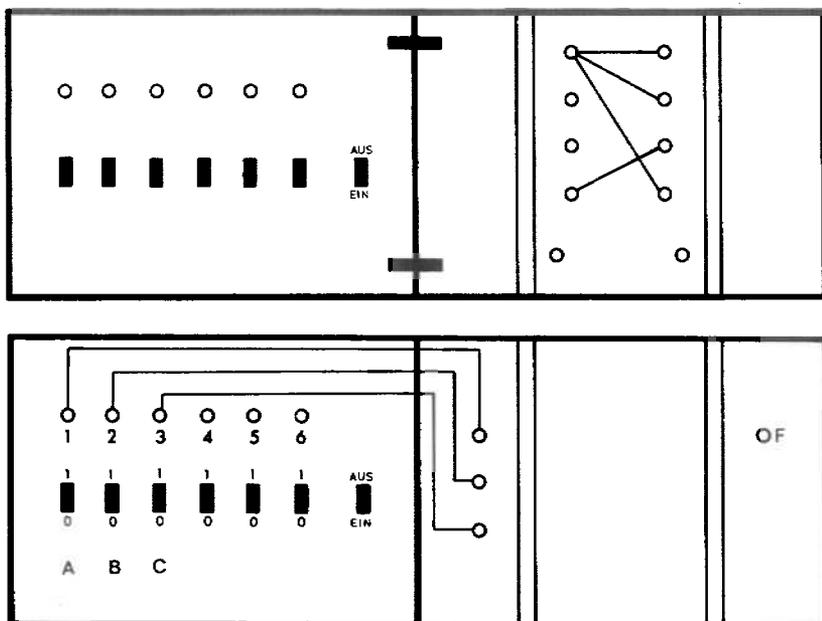


Abb. 49

Jetzt können Sie Ihrem Bekannten die Fragen stellen. Er muß sein Wissen dem Logik-Baustein durch Betätigung der Eingabeschalter mitteilen, die Elektronik in dem Logik-Baustein verarbeitet nach Ihrer Programmierung die eingegebenen Informationen und zeigt 3 richtige Antworten durch das Aufleuchten der Lampe an.

Sinngemäß lassen sich auch andere Fragen, die mit JA oder NEIN beantwortet werden können, in eine Funktionstabelle verwandeln. Mit Hilfe des entsprechenden Programmierschemas, das je nach Anzahl der Eingangsvariablen und der dabei verwendeten Eingangsbuchsen von dem Grundschema für 3 Eingangsvariablen abweicht, können Sie die aufgestellte Funktionstabelle auf den Logik-Baustein übertragen. Dazu noch ein Beispiel mit 3 Eingangsvariablen:

Frage A: "Ist der Mount Everest 8848 m hoch?"	JA (1)	NEIN (0)
Frage B: "Ist die Zugspitze 2964 m hoch?"	JA (1)	NEIN (0)
Frage C: "Ist der Feldberg (Schwarzw.) 1768 m hoch?"	JA (1)	NEIN (0)

Bedingung: Wenn zwei oder drei Fragen richtig beantwortet werden, soll die Lampe des Logik-Bausteins leuchten.

Richtig gelöst sind alle Fragen, wenn $A = 1$, $B = 1$ und $C = 0$ ist. Es ergibt sich die folgende Funktionstabelle und Programmierung:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

→ f1 wird 0
 → f2 wird \bar{C}
 → f3 wird \bar{C}
 → f4 wird 1

Die Verdrahtung von Eingabeeinheit und Logik-Baustein sieht so aus:

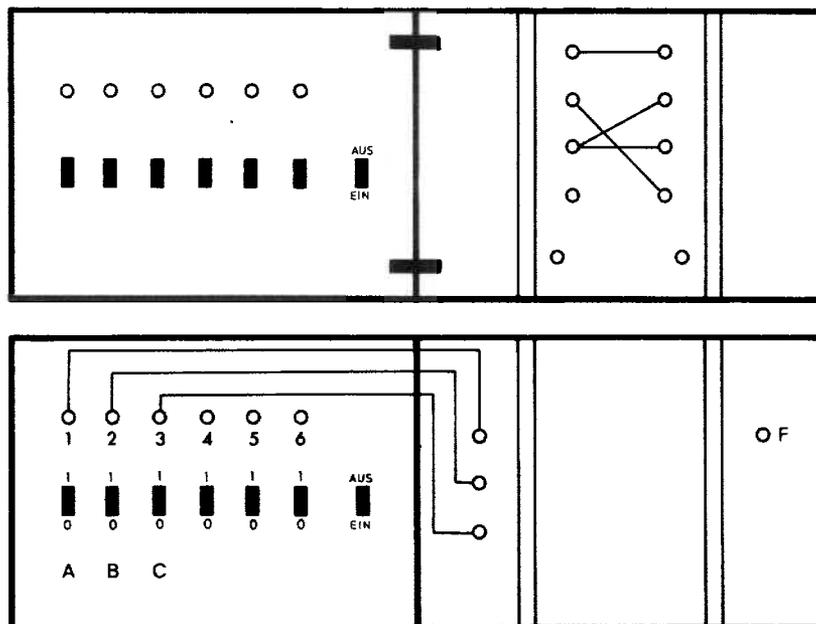


Abb. 50

Durch dieses Programmierschema sind Sie wie gesagt in der Lage, für die 3 Eingangsvariablen jede gewünschte Funktion zu programmieren. Sie können also auch z.B. die Forderung stellen, daß in der Spalte F jede dritte Ziffer 1 sein soll oder nur die zweite und die fünfte eine 0 und alle anderen 1. Dann müssen Sie wieder ermitteln, mit welchen Buchsen f1, f2, f3 und f4 zu verbinden sind.

Weitere Beispiele:

UND-Funktion

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

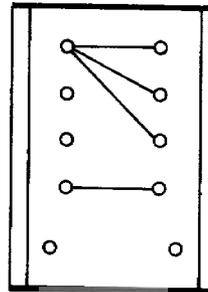
→ $f_1 = 0$

→ $f_2 = 0$

→ $f_3 = 0$

→ $f_4 = C$

Verdrahtung

ODER-Funktion

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

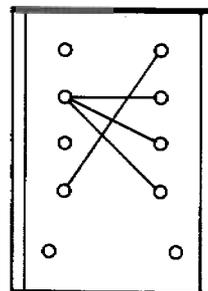
→ $f_1 = C$

→ $f_2 = 1$

→ $f_3 = 1$

→ $f_4 = 1$

Verdrahtung

Exklusiv-ODER-Funktion

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

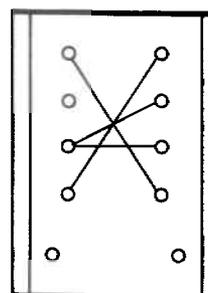
→ $f_1 = C$

→ $f_2 = \bar{C}$

→ $f_3 = \bar{C}$

→ $f_4 = 0$

Verdrahtung



Einige Seiten vorher hatten wir Ihnen eine Übersicht über die verschiedenen logischen Funktionen mit drei Eingangsvariablen gezeigt. Vielleicht versuchen Sie einmal, die Verdrahtung für die Äquivalenz-Funktion, die NAND-Funktion und die NOR-Funktion selbst herauszufinden; das Ergebnis können Sie dann später kontrollieren.

Äquivalenz-Funktion

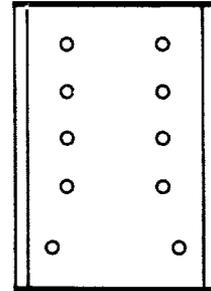
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

f1 =

f2 =

f3 =

f4 =

VerdrahtungNAND-Funktion

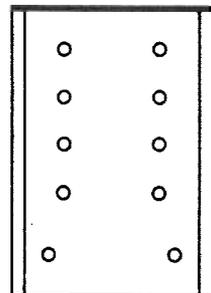
A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

f1 =

f2 =

f3 =

f4 =

VerdrahtungNOR-Funktion

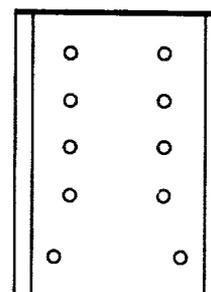
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

f1 =

f2 =

f3 =

f4 =

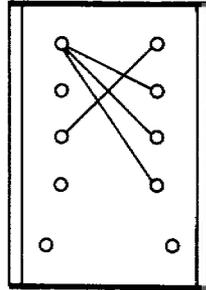
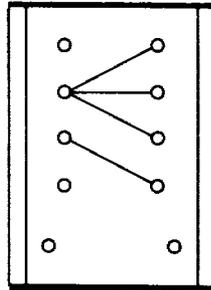
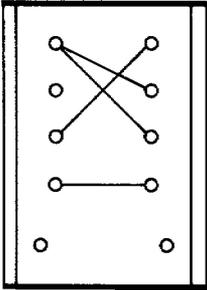
Verdrahtung

Auflösungen der vorgenannten Programmierungen:

Äquivalenz-Funktion

NAND-Funktion

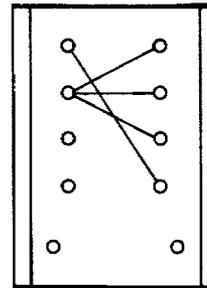
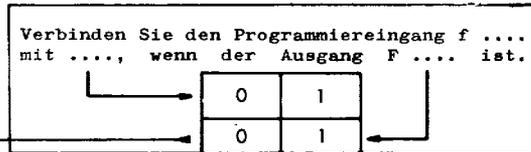
NOR-Funktion



Wie bei der Verdrahtung bei logischen Funktionen mit zwei Eingangsvariablen bzw. mit einer Eingangsvariablen zu verfahren ist, ersehen Sie aus den folgenden Erläuterungen mit je einem Beispiel.

Programmierung mit den Eingangsvariablen A und B
(C bleibt frei)

Programmierungsfeldeingänge (Zuordnung)	Eingänge		Ausgang
	A	B	
f1	0	0	
f2	0	1	
f3	1	0	
f4	1	1	



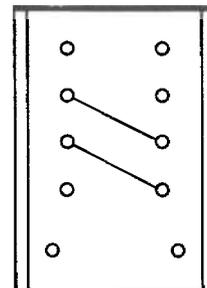
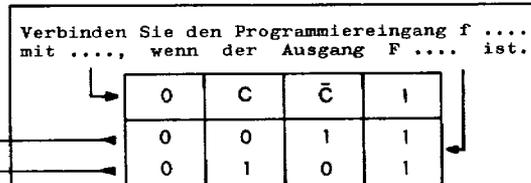
Beispiel:

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

f1 = 1
f2 = 1
f3 = 1
f4 = 0

Programmierung mit den Eingangsvariablen B und C
(A bleibt frei)

Programmierungsfeldeingänge (Zuordnung)	Eingänge		Ausgang
	B	C	
f3	0	0	
f4	1	0	
	1	1	

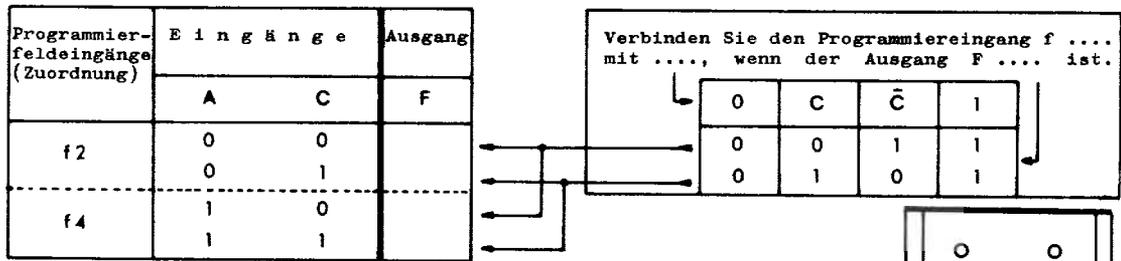


Beispiel:

B	C	F
0	0	1
0	1	1
1	0	1
1	1	0

f3 = 1
f4 = \bar{C}

Programmierung mit den Eingangsvariablen A und C
(B bleibt frei)

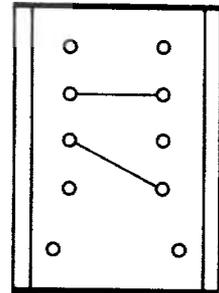


Beispiel:

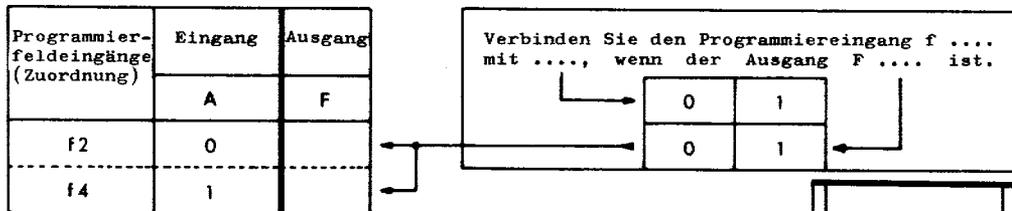
A	C	F
0	0	1
0	1	1
1	0	1
1	1	0

f2 = 1

f4 = \bar{C}



Programmierung mit der Eingangsvariablen A
(B und C bleiben frei)

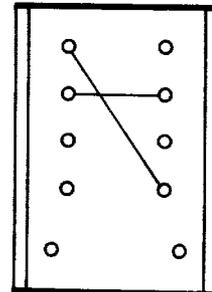


Beispiel:

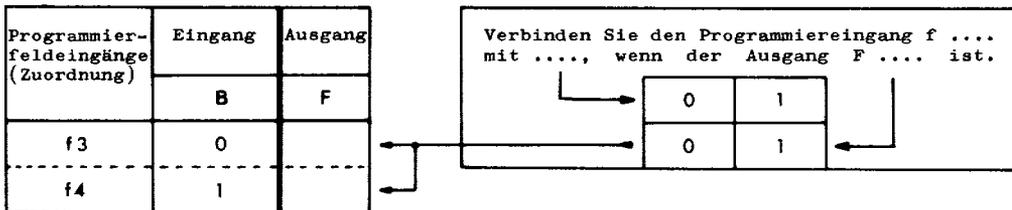
A	F
0	1
1	0

f2 = 1

f4 = 0



Programmierung mit der Eingangsvariablen B
(A und C bleiben frei)

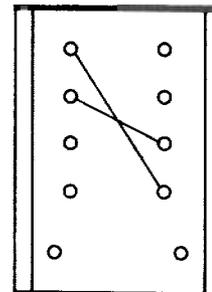


Beispiel:

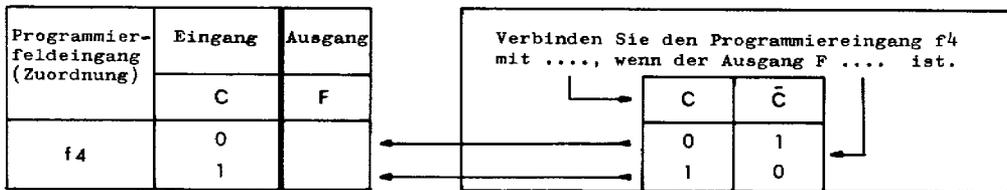
B	F
0	1
1	0

f3 = 1

f4 = 0



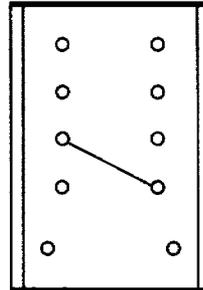
Programmierung mit der Eingangsvariablen C
(A und B bleiben frei)



Beispiel:

C	F
0	1
1	0

$$f4 = \bar{C}$$



Stellen Sie anhand der folgenden Verdrahtungen die Funktionstabellen auf.

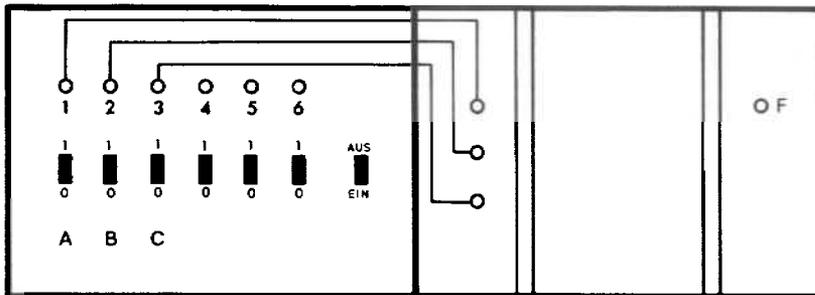


Abb. 51

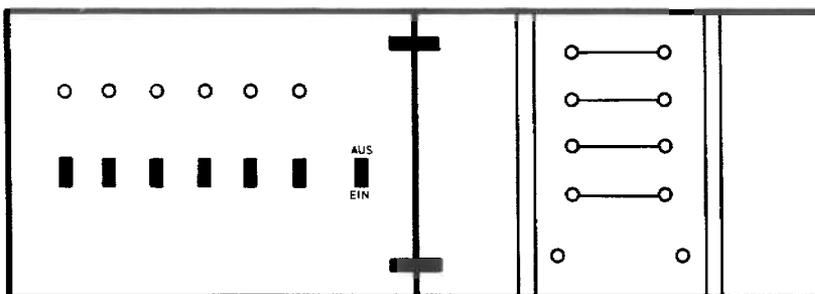


Abb. 52

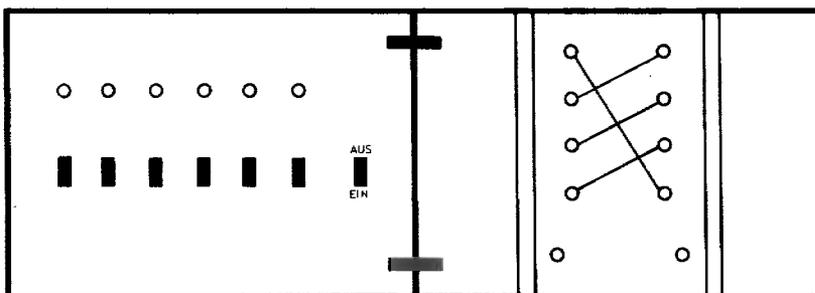
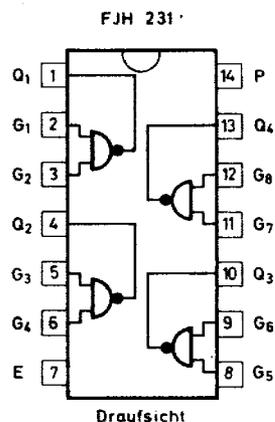


Abb. 53

8. Aufbau der Logik-Bausteine

In jedem der Logik-Bausteine Ihres CL 1601 sind drei integrierte Schaltungen eingebaut. Integrierte Schaltungen, die auch in großen Computern Verwendung finden, sind Bauelemente, die in sich die verschiedensten elektronischen Einzelteile enthalten. Ein großer Vorteil der integrierten Schaltung, auch kurz IC genannt (Abkürzung von englisch "integrated circuit"), besteht neben ihren geringen Abmessungen darin, daß sich mit ihnen die unterschiedlichsten logischen Funktionen verwirklichen lassen. Sehen wir uns als Beispiel eine integrierte Schaltung aus einem Logik-Baustein näher an.

Die folgende Abbildung zeigt die Gehäuse-Anschlüsse der integrierten Schaltung FJH 231:



Wie Sie leicht erkennen können, enthält die integrierte Schaltung FJH 231 vier NAND-Schaltungen. Je nach dem wie nun die verschiedenen Anschlüsse untereinander oder mit anderen Bauteilen verbunden werden, lassen sich die unterschiedlichen logischen Schaltungen verwirklichen. Wir wollen uns im Rahmen dieser Ausführungen zwar nicht mit Elektronik beschäftigen, doch sei für besonders Interessierte ein Beispiel einer NAND-Schaltung aus der integrierten Schaltung FJH 231 angeführt:

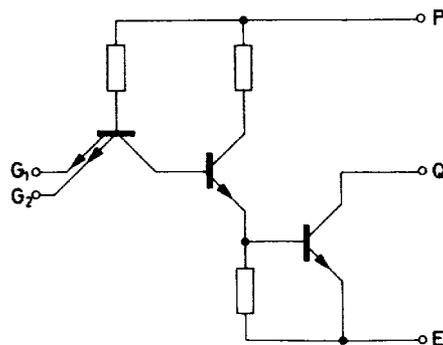
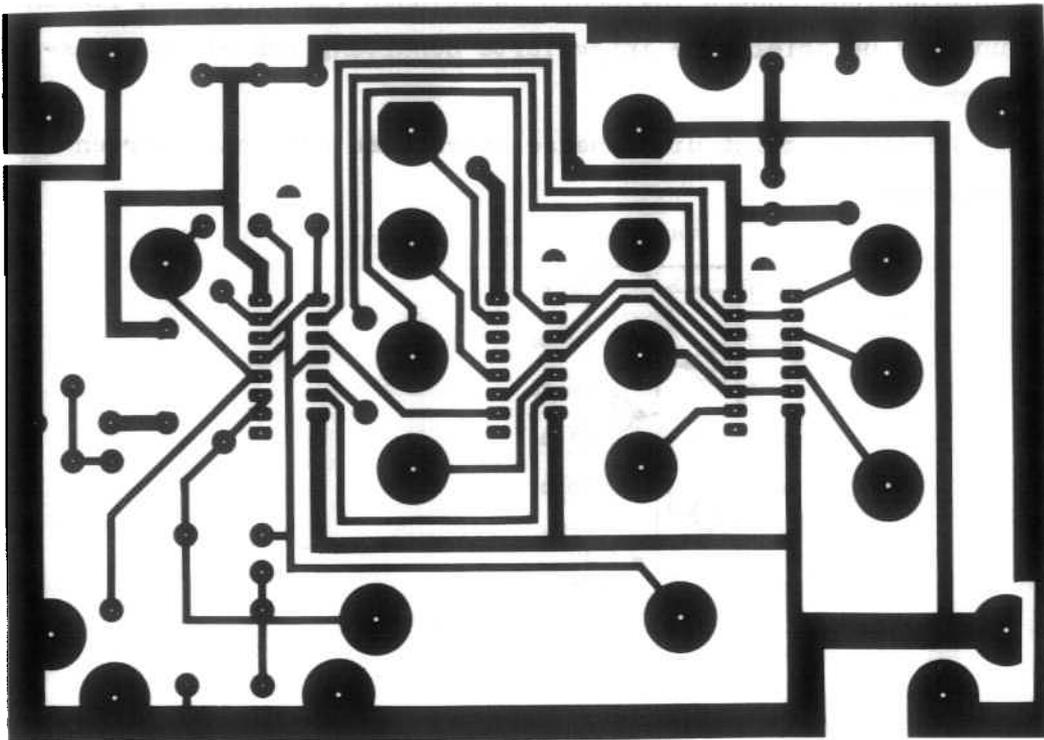


Abb. 54

Dieses NAND besteht aus mehreren Transistoren, es hat daher seinen Namen TTL-NAND = Transistor-Transistor-Logik-NAND.

Die folgende Abbildung zeigt Ihnen die gedruckte Leiterplatte, die zusammen mit den integrierten Schaltungen Hauptbestandteil der Logik-Bausteine ist.



Hier können Sie deutlich die Anschlußstellen der drei integrierten Schaltungen FJH 241, FJH 151 und FJH 231 erkennen.

Wenn Sie es ganz genau wissen wollen, beschäftigen Sie sich ein wenig mit dem folgenden logischen Schaltplan. Er zeigt Ihnen im Zusammenhang die logischen Schaltungen der drei integrierten Schaltkreise sowie die Anschlüsse zur Ein- und Ausgabe (A, B, C bzw. F) und zur Programmierung (0, 1, \bar{C} , C bzw. f1, f2, f3 und f4). Im rechten Teil der Abbildung erkennen Sie die vier NAND-Schaltungen der integrierten Schaltung FJH 231 wieder.

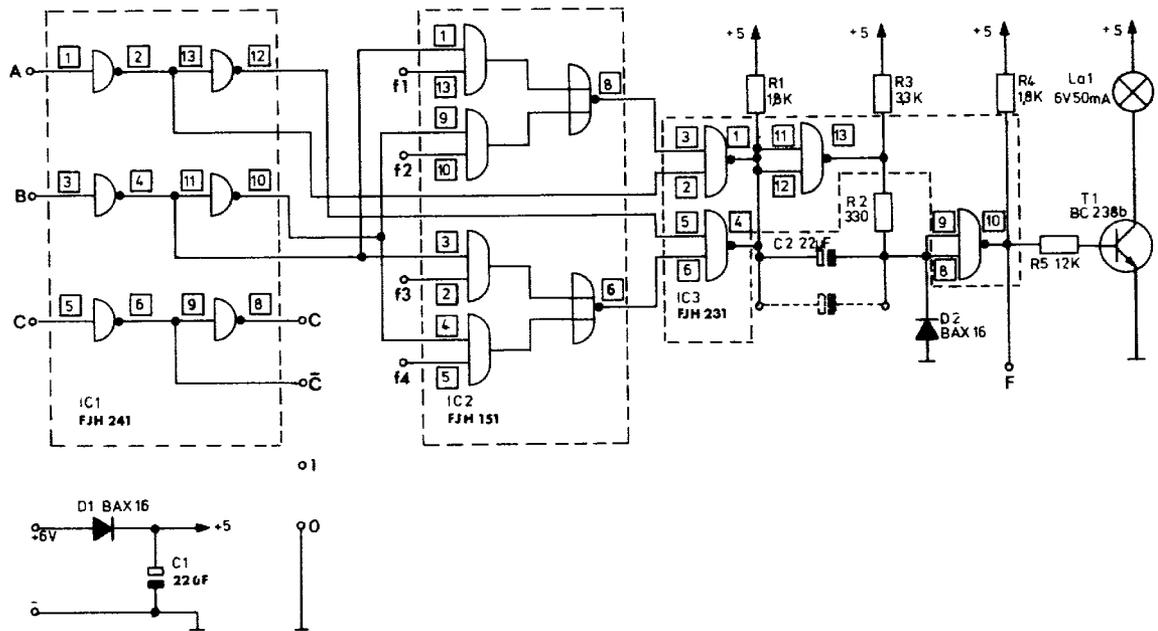


Abb. 55

Sollten Sie mit der Elektronik nicht vertraut sein, so zerbrechen Sie sich deshalb nicht den Kopf, denn wie Sie gewiß schon gemerkt haben, ist die Kenntnis der Elektronik für die Durchführung unserer logischen Experimente nicht erforderlich.

9. Blinkschaltungen

Wir programmieren zunächst einen einfachen Blinkgenerator. Er besteht erstens aus einem rückgekoppelten Inverter, d.h. einer NICHT-Schaltung, deren Ausgang wiederum mit dem Eingang verbunden, also rückgekoppelt ist, und zweitens aus einer Verzögerung.

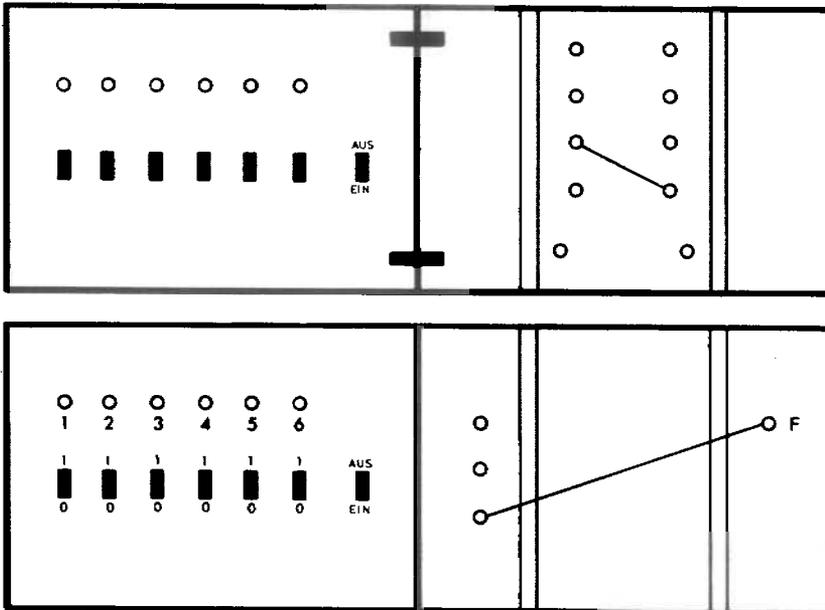


Abb. 56

Betrachten Sie zunächst noch einmal die Funktionstabelle für eine Negation.

C	F
0	1
1	0

Hieraus geht hervor, daß beim Eingang $C = 0$ der Ausgang $F = 1$ ist und umgekehrt. Da in dieser Blinkschaltung der Ausgang F mit dem Eingang C rückgekoppelt ist, muß F immer C sein. Wenn also $F = 1$ ist, ist auch $C = 1$, das bedeutet aber nach der Funktionstabelle, daß beim Eingang $C = 1$ der Ausgang $F = 0$ wird. $F = 0$ bedeutet wiederum, Eingang $C = 0$, das bedeutet, daß der Ausgang nun wieder 1 wird. Sie erkennen, daß diese Schaltung nicht stabil ist. Sie schwingt. Da in dem Baustein eine Verzögerung eingebaut ist (siehe hierzu Kapitel 10), erscheint nachdem der Eingang umgepolt wurde, das neue Ausgangssignal erst nach etwa 20 ms. Im rückgekoppelten Inverter unserer Schaltung liegt also 20 ms, nachdem das Signal C am Eingang anliegt, das entgegengesetzte Signal \bar{C} an F und damit wiederum am Eingang (durch die Rückkopplung). Weitere 20 ms später geht das Signal bei F wieder auf C und erscheint somit auch am Eingang,

so daß insgesamt 40 ms das Signal am Ausgang einmal von beispielsweise 1 nach 0 und von 0 nach 1 gewechselt hat. Wir erkennen dies daran, daß in dieser Zeit die Lampe 20 ms leuchtet und 20 ms nicht leuchtet. Da sich dies 25-mal in der Sekunde wiederholt, beobachten wir ein rasches Flimmern der Lampe.

Wollen wir nun eine Blinkschaltung bauen, die so langsam blinkt, daß wir jedes Aufleuchten und Verlöschen der Lampe gut verfolgen können, so wählen wir eine größere Verzögerung des Ausgangssignals und schalten den Kondensator von 220 uF an die Kondensatorbuchsen des Logik-Bausteins. Die Verzögerung erhöht sich dadurch auf etwa 200 ms, die Lampe leuchtet und erlischt daher nur noch etwa 2 1/2 mal in der Sekunde.

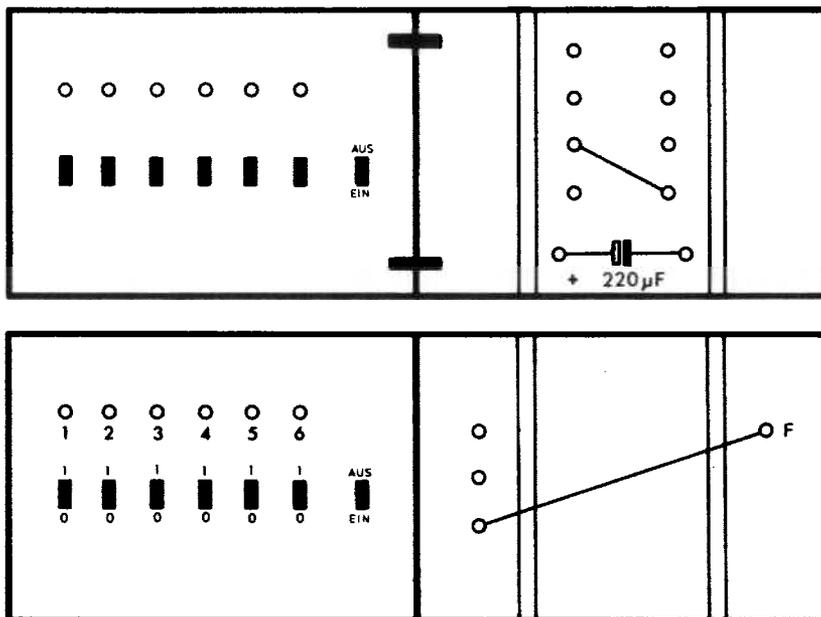


Abb. 57

Nehmen Sie sich nun einen zweiten Logik-Baustein und programmieren Sie ihn folgendermaßen mit dem ersten zusammen:

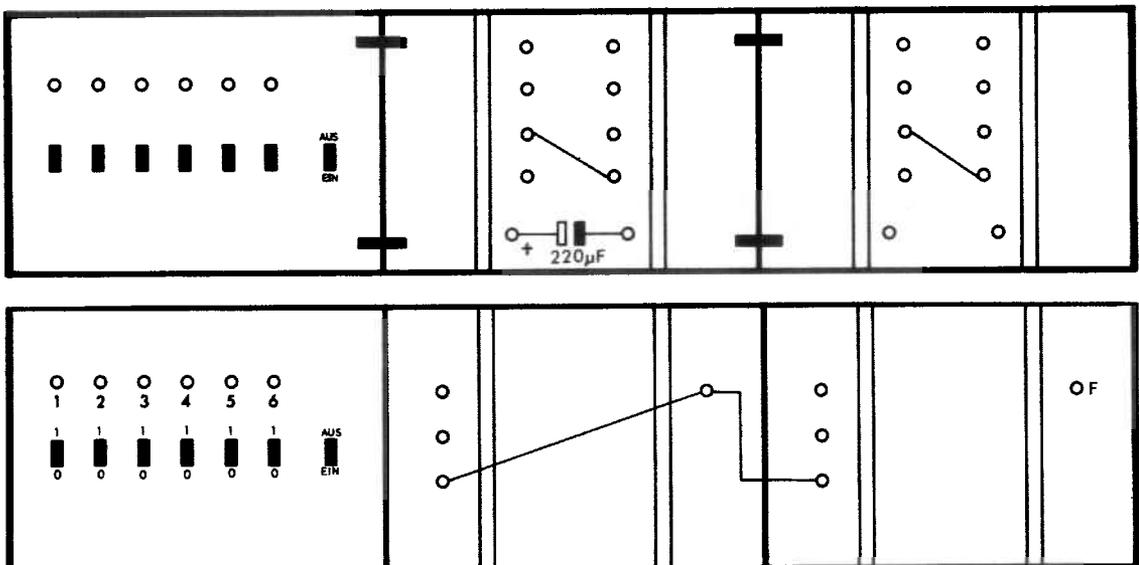


Abb. 58

Wir haben einem Blinkgenerator eine weitere Inversionsschaltung (Inverter) angebaut. Wenn wir den Hauptschalter wieder einschalten, dann leuchten die Lampen des Generators und des Inverters abwechselnd auf, also im Gegenteil.

Wenn wir jetzt bei dem Inverter im Programmierfeld den Draht von \bar{C} nach C verlegen, also auf diese Weise f^4 mit C verbinden, dann leuchten beide Lampen gleichzeitig auf. Wir haben die Inversionsschaltung zu einer Identitätsschaltung umgebaut.

Nehmen Sie nun noch einen weiteren Baustein zur Hand und programmieren Sie ihn ebenfalls als Inversionsschaltung. Programmieren Sie auch den zweiten Baustein wieder als Inversionsschaltung:

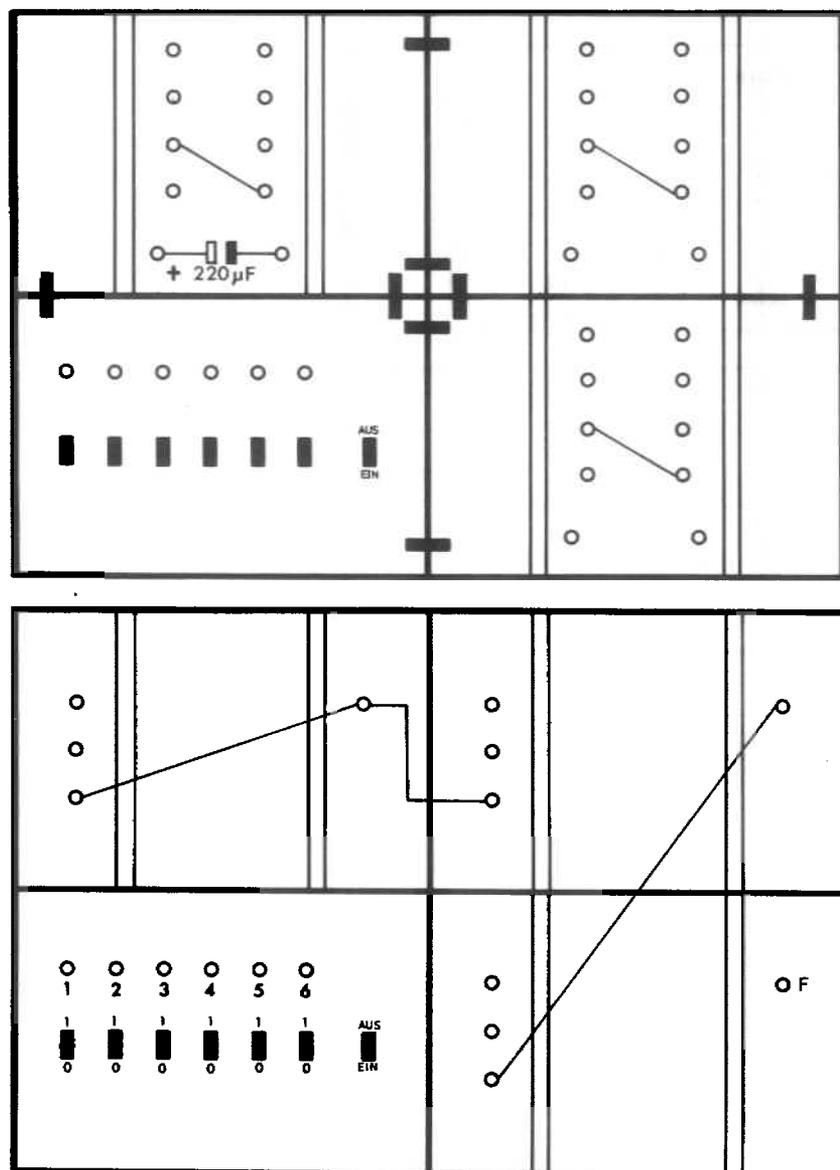


Abb.59

Sie erkennen, daß der zweite Inverter jetzt im Takt mit dem Generator blinkt, der erste Inverter im Gegenteil.

Das macht uns übrigens anschaulich, daß doppelte Verneinung eine Bejahung gibt; beide Inverter bilden sozusagen eine NICHT-NICHT-Schaltung.

Wir können dieses Spiel fortsetzen, bis wir alle fünf Logik-Bausteine aneinandergeschaltet haben. Von den fünf Bausteinen blinken nur der erste, dritte und fünfte im Takt und der zweite und vierte im Gegenteil. Wollen wir erreichen, daß z.B. der erste Baustein zusammen mit dem vierten und fünften blinkt und der zweite und dritte Baustein im Gegenteil blinken, so schalten wir den dritten und fünften Baustein nicht als Inverter, sondern als einfache Identitäts-Schaltung, d.h. wir verbinden f4 mit C:

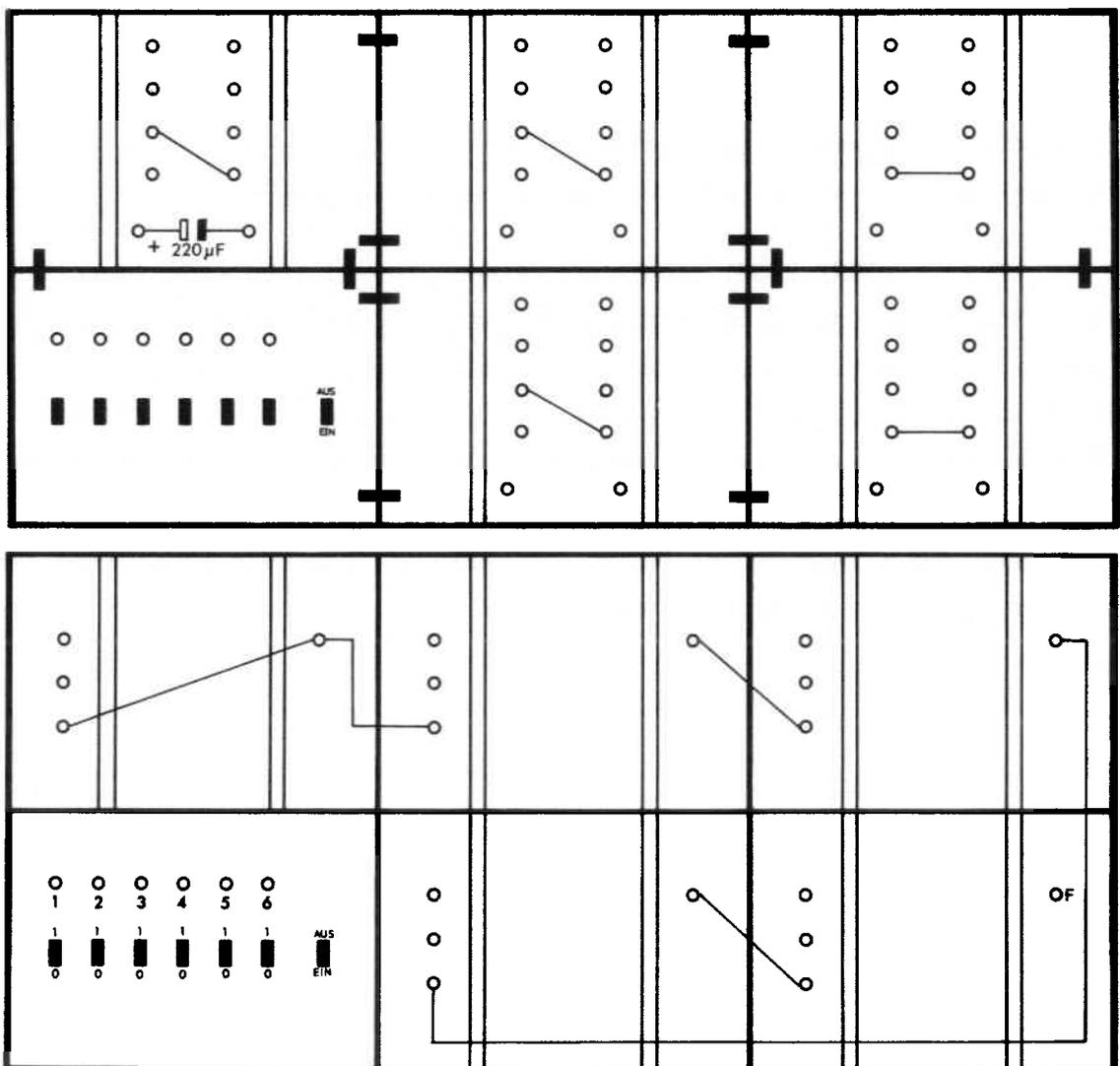


Abb. 60

Durch wahlweises Umstecken der Programmierverbindungen von f4 an die Klemmen C oder \bar{C} des Programmierfeldes können wir eine beliebige Verteilung der im Takt mit dem Blinkgenerator und im Gegenteil blinkenden Lampen erreichen. Bei der zuletzt gezeigten Anordnung erhalten wir den Eindruck, daß die roten Lampen zwischen den Bausteinen 2 und 3 oben und 4 und 5 unten hin- und herspringen. Ebenso können wir sie auch von rechts nach links und zurück oder über Kreuz springen lassen. Oder wir lassen alle fünf Lampen im Gleichtakt blinken.

10. Demonstration der Baustein-Laufzeiten

Jeder Baustein benötigt eine bestimmte Zeit, um ein oder mehrere Eingangssignale an den Ausgang weiterzugeben. Diese Zeit ist von den technischen Eigenschaften der Schaltung abhängig, z.B. von der Art der verwendeten Transistoren, von der Zahl der hintereinandergeschalteten Transistoren usw. Die Zeit, die ein oder mehrere Impulse brauchen, um vom Eingang eines Gatters an den Ausgang zu gelangen - mit dem der logischen Schaltung entsprechenden Ausgangswert - nennt man die Baustein-Laufzeit.

Wie wir bei den Versuchen mit dem Blinkgenerator gesehen haben, beträgt die Baustein-Laufzeit eines Logik-Bausteins ungefähr 20 ms. Sie läßt sich durch außen anzuschließende Kondensatoren verlängern.

Wir hatten eine NICHT-NICHT-Schaltung kennengelernt, also eine Identitätsschaltung, die das Eingangssignal unverändert bzw. doppelt invertiert an den Ausgang weitergibt, allerdings um die interne Baustein-Laufzeit von rund 20 ms verzögert. Wenn wir nun eine oder mehrere dieser NICHT-NICHT-Schaltungen zwischen den Ausgang F und den Eingang C unseres Blinkgenerators legen, an den wir jetzt keinen zusätzlichen Kondensator außen anschließen, so können wir erkennen, wie sich die Baustein-Laufzeiten der einzelnen Bausteine sichtbar addieren.

Blinkgenerator + Verzögerungsglieder	Gesamt- verzögerungszeit	Blinktakt Hz (Hertz = Takte/sec.)
0	ca. 20 ms	ca. 25 Hz
1	ca. 40 ms	ca. 12 Hz
2	ca. 60 ms	ca. 8 Hz
3	ca. 80 ms	ca. 6 Hz
4	ca. 100 ms	ca. 5 Hz

Baustein 1 bildet also den Blinkgenerator (rückgekoppelter Inverter), die Bausteine 2 bis 5 wirken als Verzögerungsglieder. Jede einzelne Baustein-Laufzeit können wir durch einen außen angeschlossenen Kondensator noch verlängern.

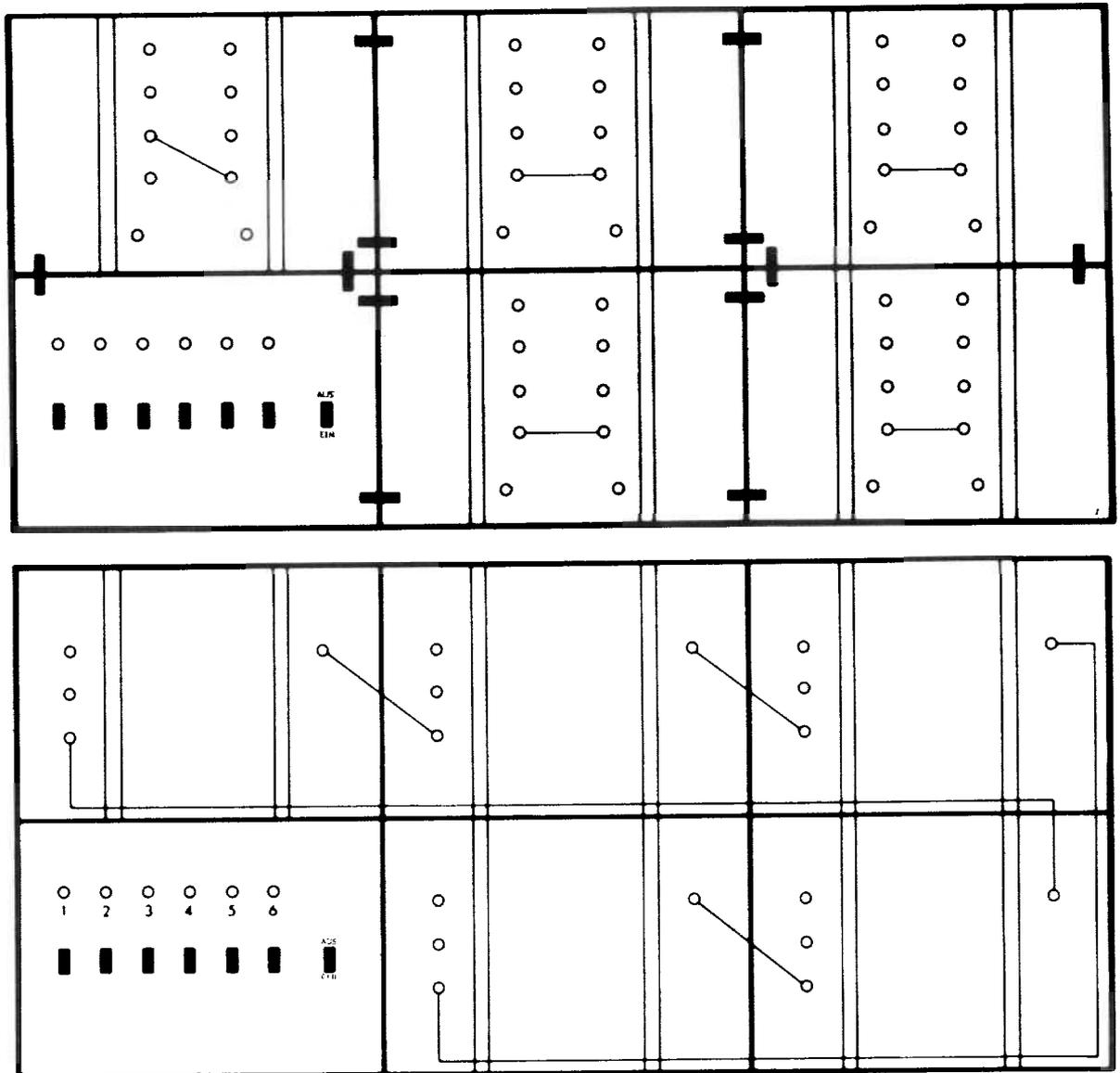


Abb.61

Je mehr Bausteine wir in die Rückkopplung des Blinkgenerators einbauen, um so größer wird die gesamte Laufzeit und um so langsamer blinkt der Generator. Die Anordnung mit 5 Bausteinen sieht dann so aus:

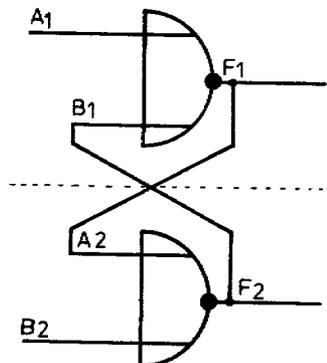
11. Flipflops

Um Informationen in einem Computer speichern zu können, bedient man sich außer dem Magnetbandspeicher häufig vollenlektronischer Speicherelemente, die im Gegensatz zu den Bandgeräten ohne mechanisch bewegliche Teile auskommen. Sie können nahezu die gleichen Arbeiten übernehmen und haben als wesentlichen Vorteil eine viel kürzere Zugriffszeit.

Ein solches Speicherelement ist das Flipflop, das die Eigenschaft besitzt, duale Informationen, die kurzzeitig an seinem Eingang liegen, über einen beliebigen Zeitraum speichern zu können. Das Fassungsvermögen des Einzelspeichers beträgt 1 bit, das bedeutet, daß nur die kleinste duale Einheit (0 oder 1) gespeichert werden kann.

11.1. Speicher-Flipflop (R-S-Flipflop)

Die einfachste Form aller Flipflops ist das sogenannte Speicher-Flipflop. Es läßt sich z.B. aus zwei kreuzweise rückgekoppelten NOR-Schaltungen aufbauen, wie das folgende Schaltbild zeigt.



Bauen Sie zunächst diese Schaltung aus zwei Logik-Bausteinen auf. Aus der Funktionstabelle für eine NOR-Schaltung ergibt sich nach dem Programmierschema für zwei Eingangsvariable (A und B) - Kapitel 7 - folgende Verdrahtung:

Funktionstabelle NOR:

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

→ f1 = 1
 → f2 = 0
 → f3 = 0
 → f4 = 0

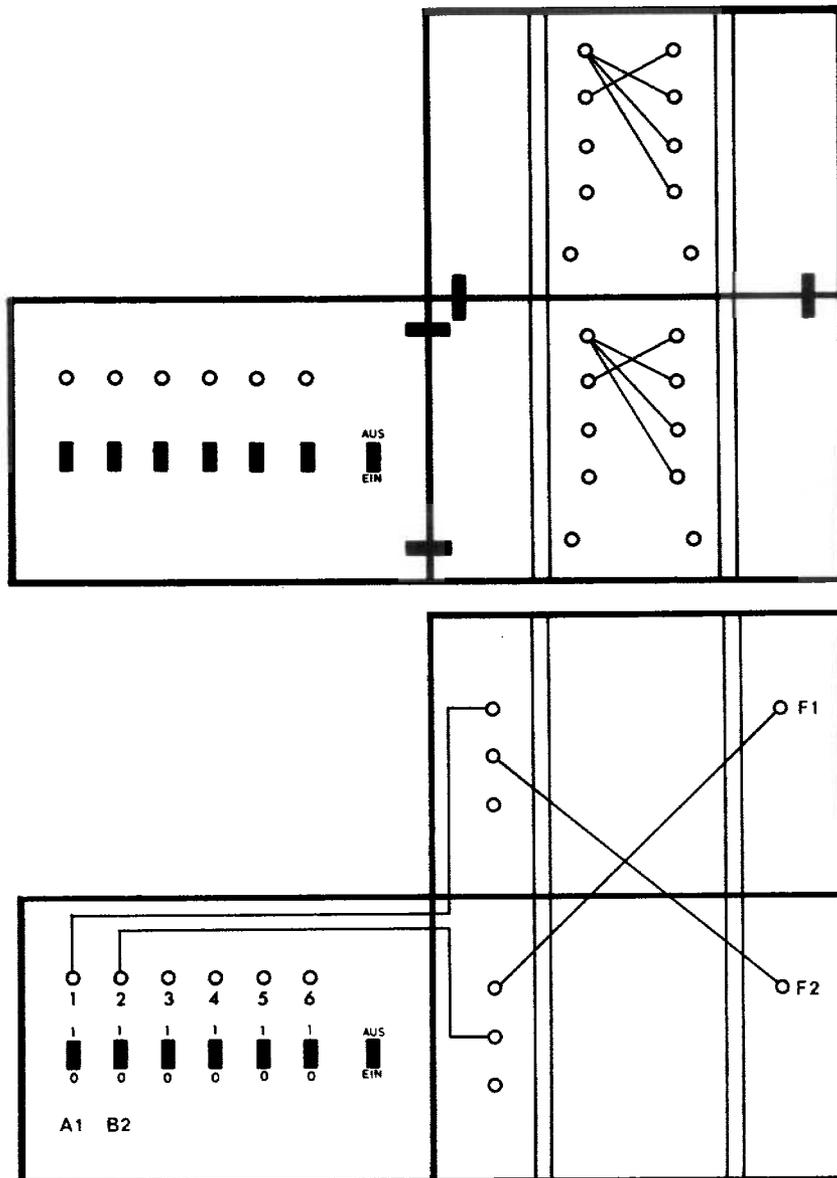


Abb.62

Sie können sich anhand der logischen Schaltsymbolik veranschaulichen, wie das Flipflop arbeitet. Die Funktionstabelle für beide kreuzweise gekoppelten NOR-Schaltungen unterstützt Sie bei diesen Überlegungen.

	Schalter/Eing.		Ausg.
	A1	B2	F1
1.	0	0	0
2.	0	1	1
3.	0	0	1
4.	1	0	0
5.	0	0	0
6.	1	1	0
7.	0	0	1
8.	0	0	0

1. NOR			2. NOR		
A1	$\bar{B1}$	F1	A2	$\bar{B2}$	F2
0	1	0	0	0	1
0	0	1	1	1	0
0	0	1	1	0	0
1	1	0	0	0	1
0	1	0	0	0	1
1	0	0	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1

Anmerkung zu 7.: 1. NOR ist schneller als 2. NOR.
 Anmerkung zu 8.: 2. NOR ist schneller als 1. NOR.

Gehen wir einmal von folgender Situation aus:

1.) Am Ausgang F1 liegt ein 0-Signal, und die Eingänge A1 und B2 sind ebenfalls auf 0. Aus dem rechten Teil der Tabelle können Sie entnehmen, daß der Ausgang der zweiten NOR-Schaltung auf 1 liegt, da durch die Rückkopplungsleitung von F1 auf A2 beide Eingänge 0 sind. 1 an F2 und aufgrund der Rückkopplung gleichzeitig an B1 bedeutet für F1 = 0. Dieser Zustand wird als stabil bezeichnet.

2.) Wird jetzt der Schalter B2 auf 1 gelegt, ist F2 = 0. Durch die direkte Drahtverbindung zwischen F2 und B1 wird F1 = 1, da beide Eingänge der 1. NOR-Schaltung 0 sind. Dieser Zustand des Flipflops wird ebenfalls als stabil bezeichnet. Der Ausgangswert F1 = 1 bleibt auch bestehen, wenn, wie in Position 3, der Schalter B2 wieder auf 0 gesetzt wird. Damit hat die Schaltung die Information 1 gespeichert.

Die Information können Sie nur löschen, indem Sie A1 auf 1 schalten (4.). F1 ist dann 0, dementsprechend auch A2 und B2, so daß F2 = 1 wird.

5.) Legen Sie nun den Eingang A1 wieder auf 0. Die Information F1 = 0 bleibt erhalten.

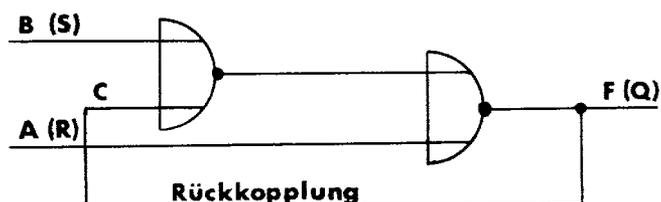
Werden A1 und B2 = 1 (Pos. 6.), so ist für diese Zeit F1 zwar 0, bei der nachfolgenden Informationseingabe (A1 = 0 und B2 = 0 in Pos. 7. und 8.) stellt sich jedoch der Ausgang F1 auf einen nicht voraussehbaren Wert ein, der nur durch die Schaltzeiten der NOR-Bausteine bedingt ist. Arbeitet z.B. die erste NOR-Schaltung schneller, dann tritt der Ausgangswert F1 = 1 ein (7.). Ist dagegen die zweite NOR-Schaltung schneller, so wird F1 = 0 (siehe Pos. 8.). Sie können die Funktionstabelle auf diese Eigenart hin überprüfen, wenn Sie durch den Anschluß des 220 μ F Elektrolytkondensators die Verzögerungszeit einer NOR-Schaltung künstlich verlängern. Sind beide Verzögerungszeiten gleich, dann blinkt die Schaltung im schnellen Rhythmus, sofern A1 und B2 gleichzeitig von 1 auf 0 geschaltet werden. Durch dieses undefinierte Verhalten bezeichnet man die Pos. 6. bis 8. als nicht stabil. Sie dürfen bei dem Betrieb eines Flipflops nicht auftreten.

Aus diesen Überlegungen läßt sich für das Speicher-Flipflop folgende stark vereinfachte Funktionstabelle ableiten:

A1	B2	F1
0	0	0
0	1	1
1	0	0
1	1	?

Zur Auswertung dieser Tabelle noch einige wichtige Anmerkungen:
 Bei allen Flipflops wird die Wertigkeit der Ausgangsvariablen F1 für den Zeitpunkt angegeben, wenn die einzustellenden Informationen nicht mehr an den Eingängen liegen und, wie in unserem Beispiel A1 und B2, wieder 0 sind. Daraus ergibt sich: Wenn $A1 = 0$ und $B2 = 1$, ist der Ausgang $F1 = 1$. Diese Information bleibt auch bestehen, wenn A1 und B2 wieder 0 werden. Ist $A1 = 1$ und $B2 = 0$, wird $F1 = 0$ und bleibt in diesem Zustand, auch wenn A1 und B2 wieder 0 werden. Aus diesem Grund finden Sie auch in der ersten Spalte der Funktionstabelle die zwei Möglichkeiten 0 und 1 für die Ausgangsvariable F1. In der letzten Zeile steht unter F1 ein Fragezeichen. Das bedeutet: Dieser Zustand ist nicht stabil, wenn A1 und B2 wieder auf 0 gehen.

In den vorangegangenen Überlegungen haben wir das Speicher-Flipflop mit zwei Logik-Bausteinen verwirklicht. Dadurch ergaben sich zwei Ausgänge, wobei jedoch nur der Zustand des Ausgangs F1 betrachtet wurde. Wenn das Logikschaltbild etwas umgezeichnet wird, erkennen Sie, daß bei nur einem Ausgang das Speicher-Flipflop sich mit einem Logik-Baustein realisieren läßt.



Um nach der Funktionstabelle die Programmierung finden zu können, trennen wir zunächst die Rückkopplungsleitung von F nach C auf.

A (R)	B (S)	C	F (Q)	
0	0	0	0	
0	0	1	1	→ f1 = C
0	1	0*	1	
0	1	1**	1	→ f2 = 1
1	0	0***	0	
1	0	1*	0	→ f3 = 0
1	1	0***	0	
1	1	1*	0	→ f4 = 0

Wenn Sie nun nur die Eingangsvariablen A und B betrachten, erkennen Sie, daß bei A und B = 0, F = 0 oder 1, bei A = 0 und B = 1 F nur 1 sein kann. Wird B wieder 0, bleibt die Information F = 1 erhalten. Bei A = 1 und B = 0 kann F nur 0 werden, und auch bei A und B = 1 wird F = 0. Dieser letzte Zustand ist jedoch nicht stabil. Durch die Rückkopplung von F auf C sind die in der Funktionstabelle mit einem Stern versehenen Zustände nicht möglich und gehen innerhalb der Zweiergruppe in die mit zwei Sternen versehenen Zustände über.

Aus der gefundenen Programmierung ergibt sich folgende Verdrahtung:

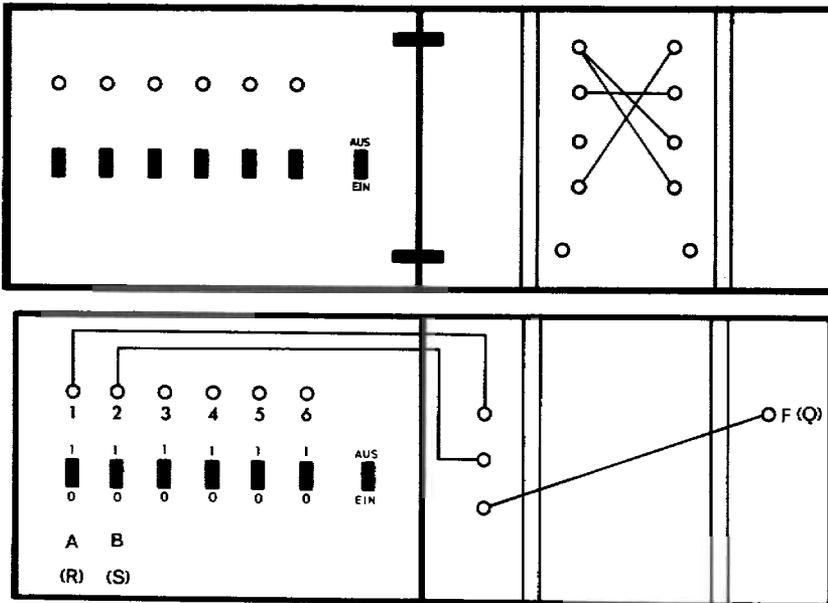


Abb.63

Zu Beginn des Experimentes sollen die beiden Schalter A und B auf 0 stehen. Je nach der Endsituation eines vorangegangenen Schaltexperimentes steht der Ausgang F auf 1 oder 0. Unabhängig davon hat der Ausgang F jedoch den Wert 1, wenn Sie den Schalter B auf 1 stellen. Schalten Sie den Schalter B dann wieder auf 0, werden Sie sehen, daß der Wert F = 1 erhalten bleibt; die 1 ist gespeichert. Auch wenn Sie den Schalter B wieder auf 1 legen, ändert sich der Speicherzustand nicht.

Betätigen Sie nun den Schalter A. Sobald Sie ihn auf 1 stellen, wird die 1 am Ausgang F wieder gelöscht; wir können auch sagen, der Ausgang F hat den Wert 0. Dieser Wert 0 bleibt auch erhalten, wenn Sie den Schalter A wieder auf 0 zurückstellen, mit anderen Worten, unser Speicher-Flipflop hat jetzt den Wert 0 gespeichert.

Die Schalterstellung A und B = 1 ergeben an F zwar 0, wenn aber beide Schalter wieder auf 0 gesetzt werden, stellt sich ein Zustand ein, der nicht berechenbar ist und dem Zufall überlassen werden muß.

Die Operation $B (S) = 1$ (Betätigung des Schalters B) wird auch mit "set" (setzen) und $A (R) = 1$ mit "reset" (rücksetzen) bezeichnet. Daher die zunächst in Klammern gesetzten Eingangsbezeichnungen R und S. Außerdem spricht man bei einem Flipflop von dem Ausgang Q, wobei der stabile Ruhezustand, der in unserem Beispiel bei $A (R) = B (S) = 0$, die Werte 0 oder 1 annehmen kann. Aus diesen neuen Gesichtspunkten kommen wir zu der Funktionstabelle. Die in Klammern gesetzten Ein- und Ausgangsbezeichnungen beziehen sich auf die Eingänge des Logik-Bausteins.

R (A)	S (B)	Q (F)
0	0	0 1
0	1	1
1	0	0
1	1	?

Die Funktionstabelle ist folgendermaßen zu lesen:

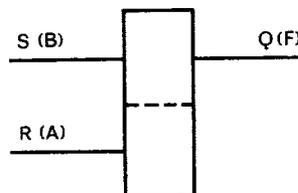
Bei $R = S = 0$ behält Q seinen alten Wert ($Q = 0$ oder 1).

$R = 0$ und $S = 1$ bewirkt $Q = 1$.

$R = 1$ und $S = 0$ bewirkt $Q = 0$.

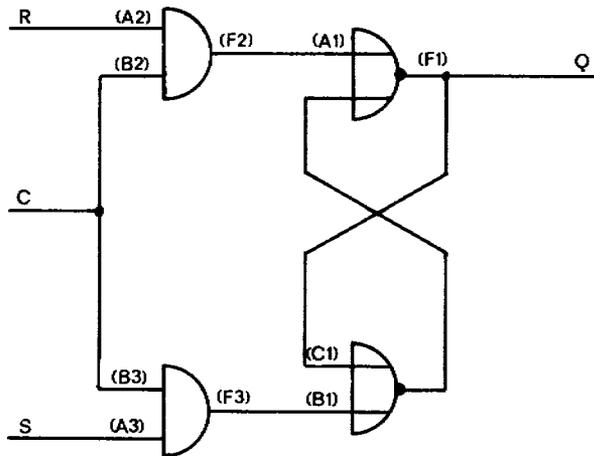
$R = 1$ und $S = 1$ bewirkt $Q = 0$, solange die Eingangssignale erhalten bleiben. Werden die Eingangssignale jedoch gleichzeitig wieder von 1 auf 0 gesetzt, so stellt sich ein nicht vorhersehbarer Zustand ein ($Q = ?$).

Für ein Speicher-Flipflop sieht das Logik-Symbol folgendermaßen aus:



11.2. Auffang-Flipflop

Ein abgewandeltes Speicher-Flipflop ist das sogenannte Auffang-Flipflop, das vielfach auch als getaktetes R-S-Flipflop bezeichnet wird. Es unterscheidet sich von dem Speicher-Flipflop nur durch die beiden vorgeschalteten UND-Schaltungen.



Beide UND-Schaltungen bewirken, daß die Signale R und S nur dann an den Eingängen des Flipflops wirksam werden können, wenn der Takteingang C (C von englisch "clock") auf 1 geht. Bei dieser Schaltungsart des Speicher-Flipflops kann man die Eingänge R und S als sogenannte Vorbereitungseingänge betrachten, die das logische Verhalten des Flipflops vorbestimmen, wobei der Eingang C den Zeitpunkt festlegt, wann die an den Vorbereitungseingängen liegende Information gespeichert werden soll. In Computern findet man diese Art des Flipflops überall dort, wo nur zu einem genau definierten Zeitpunkt Informationen gleichzeitig in bestimmte Speicher gegeben werden sollen, während andere Speichergruppen für diese Informationen gesperrt bleiben.

C	R	S	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	?

Und so sieht die Schaltung des Auffang-Flipflops mit 3 Logik-Bausteinen aus:

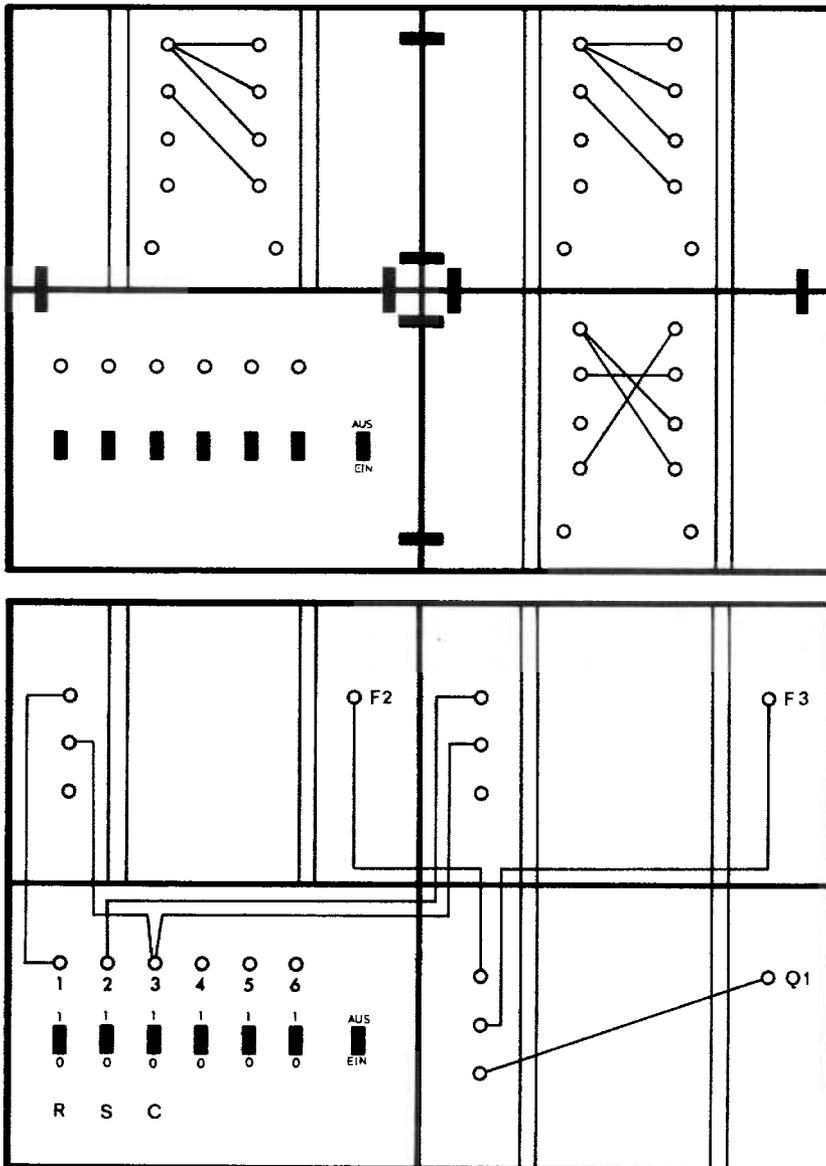
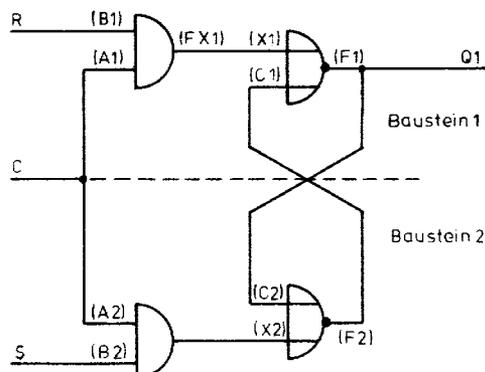


Abb.64

Wenn Sie das Auffang-Flipflop mit nur zwei Bausteinen realisieren wollen, müssen Sie sich die Gesamtschaltung als zwei symmetrisch aufgeteilte Schaltungshälften vorstellen, die aus in Reihe geschalteten UND- und NOR-Gattern bestehen.



Um die Funktionstabelle für die Reihenschaltung einer UND- und NOR-Schaltung aufstellen zu können, brauchen Sie zunächst nur das Logikschema für den ersten Baustein zu betrachten. Aus den Einzeltabellen für das UND und NOR mit zwei Eingängen ergibt sich folgende Funktionstabelle, aus der Sie leicht die Programmierung ableiten können:

A1	B1	FX1
0	0	0
0	1	0
1	0	0
1	1	1

X1	C1	F1
0	0	1
0	1	0
1	0	0
1	1	0

A1	B1	C1	F1
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = \bar{C}

→ f2 = \bar{C}

→ f3 = \bar{C}

→ f4 = 0

Da beide Schaltungshälften die gleiche Funktion aufweisen, erhalten Sie mit den zusätzlich anzubringenden Rückkopplungsleitungen F2 - C1 und F1 - C2 den Verdrahtungsplan eines Auffang-Flipflops mit zwei Logik-Bausteinen.

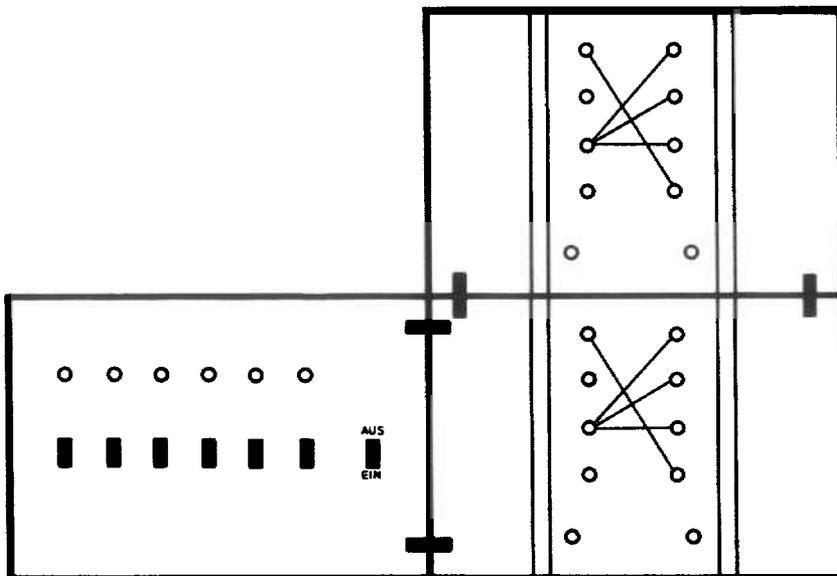


Abb. 65

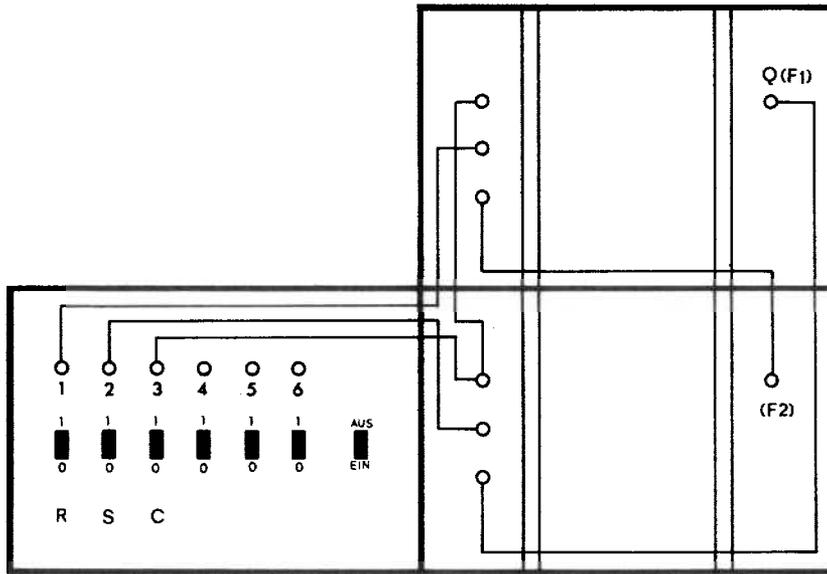
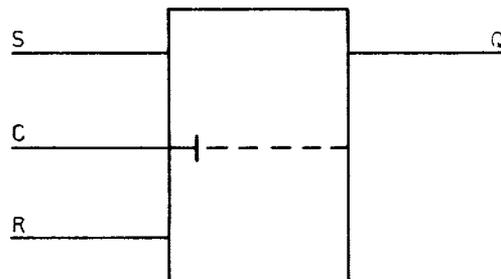


Abb. 65

Funktionstabelle des
Auffang-Flipflops

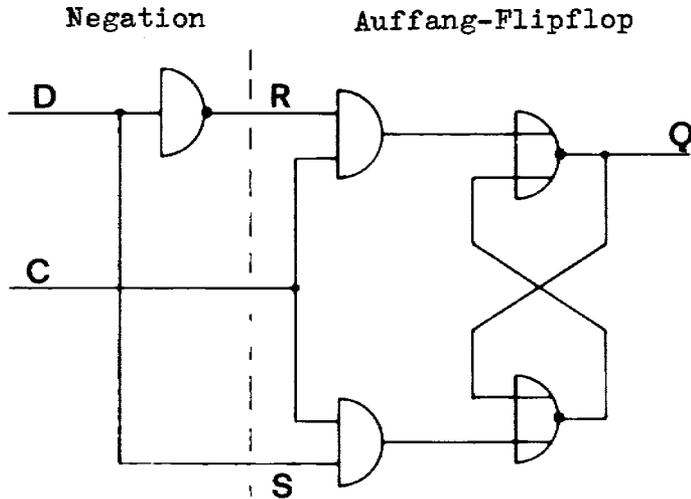
C	R	S	Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	?

Logiksymbol des
Auffang-Flipflops



D-Auffang-Flipflop 1

Ein anderes Auffang-Flipflop ist das sogenannte D-Auffang-Flipflop (Delay-Auffang-Flipflop, englisch delay = Verzögerung), das in vier verschiedenen Ausführungsformen möglich ist. Betrachten Sie zunächst den Logikschaltplan des D-Auffang-Flipflop 1 - kurz DFF1 genannt.



Wie Sie erkennen können, besteht es aus dem getakteten RS-Flipflop (Auffang-Flipflop) und einer im (R)-Eingang liegenden Negation. Außerdem ist der negierte (R)-Eingang direkt mit (S) verbunden und mit der neuen Bezeichnung D versehen. Die Speicherinformation am D-Eingang kann das DFF1 wie auch beim Auffang-Flipflop nur dann übernehmen, wenn der Takteingang $C = 1$ ist.

Liegt bei $C = 1$ z.B. D auf 0, ist auch $(S) = 0$, durch die Negation wird (R) jedoch 1. Der Ausgang Q ist wie beim Auffang-Flipflop 1. Geht der Takteingang auf 0 zurück, bleibt die Information $Q = 1$ erhalten, auch wenn an D ein anderes Signal vorhanden ist. Bei $C = 1$ und $D = 1$ ist $(S) = 1$ und $(R) = 0$, daraus resultiert der Ausgang $Q = 1$.

Prüfen Sie diesen logischen Zusammenhang anhand der Verdrahtung eines D-Auffang-Flipflops 1 mit drei Logik-Bausteinen. Falls Sie noch den Auffang-Flipflop aus 2 Logik-Bausteinen aufgebaut haben, brauchen Sie nur noch die Negation in die Leitung (R) einzufügen und den Eingang der Negation mit S zu verbinden.

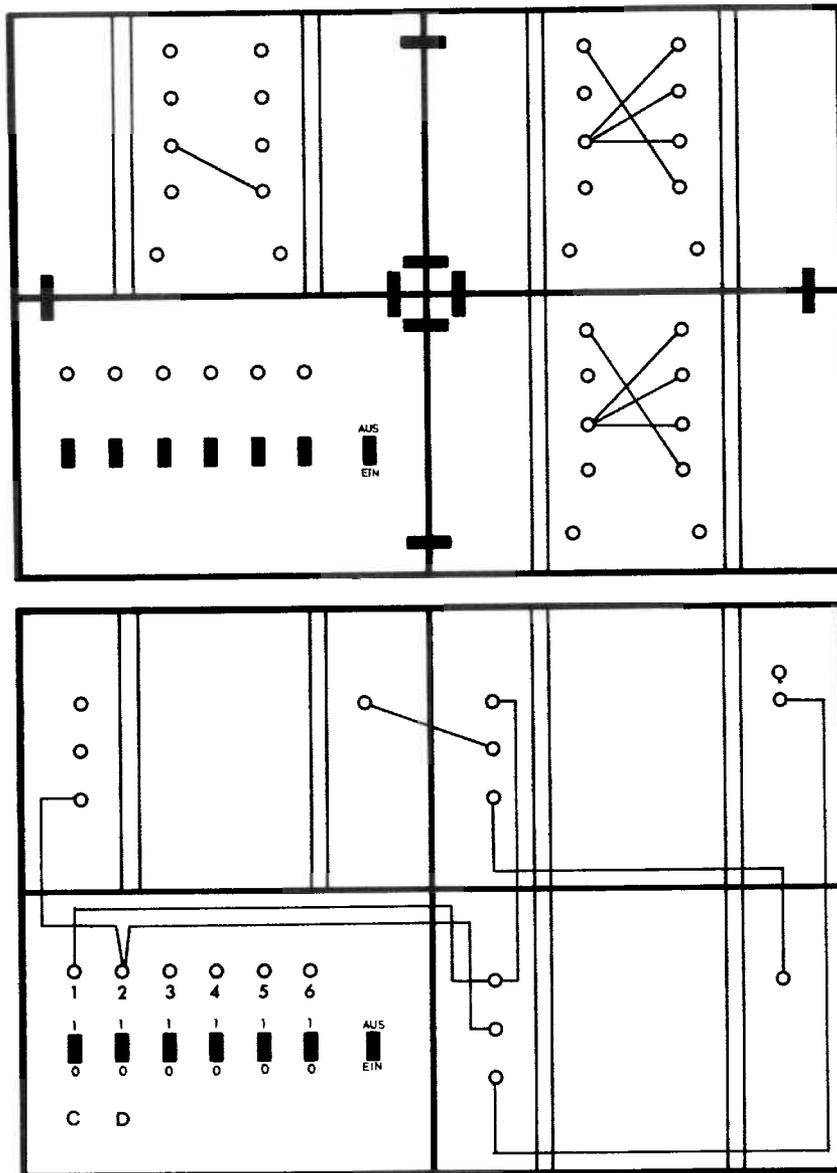
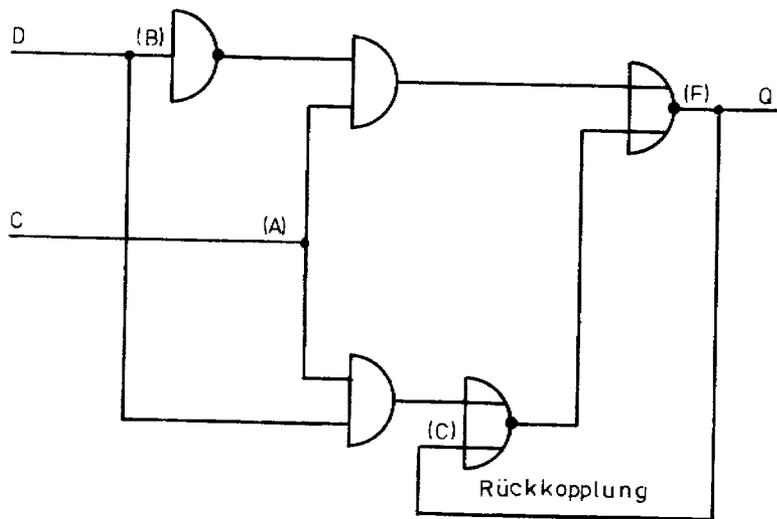


Abb.66

Sie erhalten dann eine Schaltung, die dieser Funktionstabelle entspricht:

C	D	Q
0	0	0
0	1	1
1	0	0
1	1	1

Versuchen Sie nun, diese Schaltung auf nur einen Logik-Baustein zu übertragen. Der Übersicht wegen zeichnen wir den Logikschaltplan so um, daß entsprechend dem zu programmierenden Logik-Baustein 3 Eingänge und ein Ausgang zu erkennen sind.



Alle in Klammern gesetzten Ein- und Ausgangsbezeichnungen beziehen sich auf die Ein- und Ausgangsbuchsen des Logik-Bausteins. Um die Programmierung finden zu können, müssen Sie sich zunächst die Rückkopplungsleitung von (F) nach (C) aufgetrennt denken. So können Sie aus der Funktionstabelle für das D-Auffang-Flipflop in bekannter Weise die Programmierung ableiten.

Funktionstabelle DFF1

C	D	Q
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	1	1

Programmierung

(A) C	(B) D	(C)	(F) Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0 ⁺⁺	0
1	0	1 ⁺	0
1	1	0 ⁺	1
1	1	1 ⁺⁺	1

→ f1 = C

→ f2 = C

→ f3 = 0

→ f4 = 1

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen, dann sind die mit + versehenen Zustände nicht möglich. Sie gehen in den mit ++ gekennzeichneten Zustand über.

Wenn Sie gemäß der gefundenen Programmierung den Baustein verdrahten und die Rückkopplungsleitung von (F) nach (C) anschließen, erhalten Sie das D-Auffang-Flipflop 1 mit nur einem Logik-Baustein.

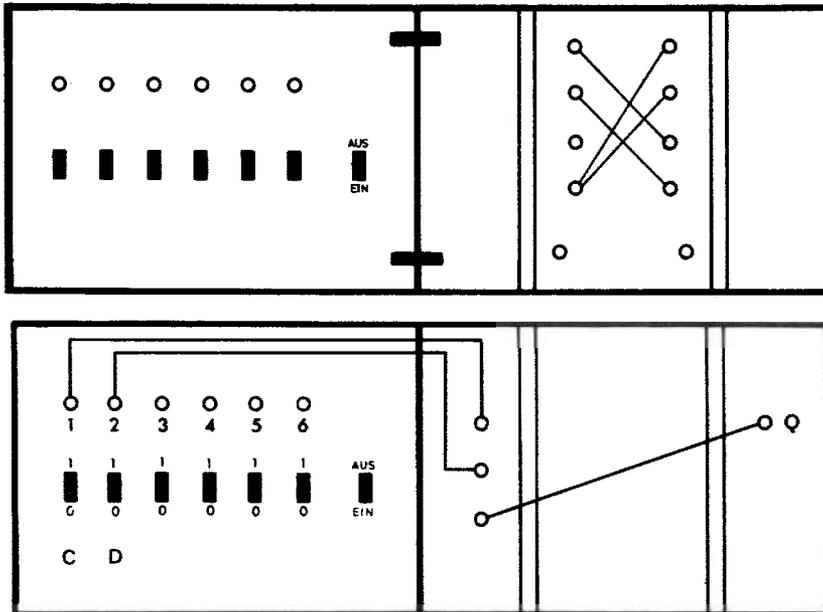
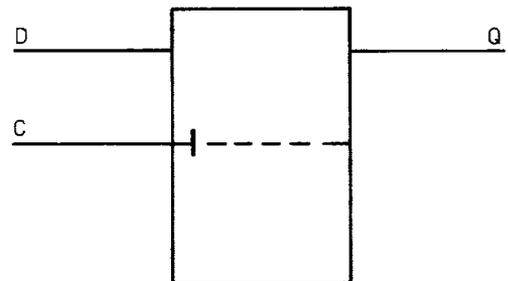


Abb.67

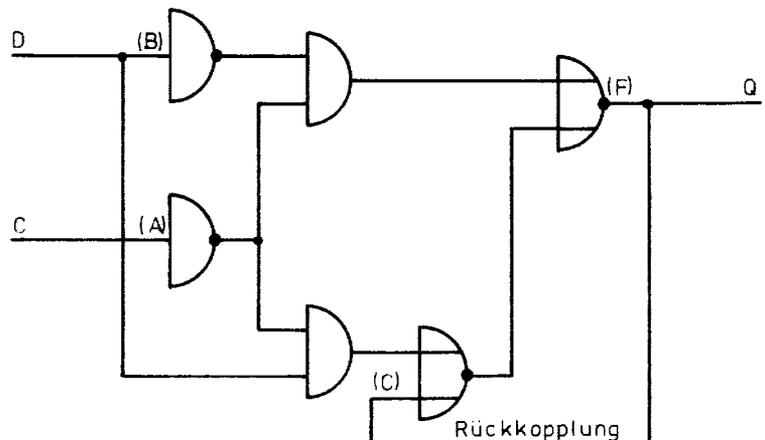
Logiksymbol
für das DFF1



D-Auffang-Flipflop 2 (DFF2)

Fast die gleichen Eigenschaften wie das DFF1 hat auch das D-Auffang-Flipflop 2. Es unterscheidet sich lediglich vom DFF1 in der Inversion des Takteingangs C. Nur wenn $C = 0$ ist, kann eine Information gespeichert werden. Bei $C = 1$ ist eine Informationseingabe nicht möglich, das Signal am Eingang D hat keinen Einfluß auf das Flipflop.

Logikschaltplan
des DFF2



Funktionstabelle DFF2

C	D	Q
0	0	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Programmierung

(A) C	(B) D	(C)	(F) Q
0	0	0 ⁺⁺	0
0	0	1 ⁺	0
0	1	0 ⁺	1
0	1	1 ⁺⁺	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

→ f1 = 0

→ f2 = 1

→ f3 = C

→ f4 = C

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen, dann sind die mit + versehenen Zustände nicht möglich. Sie gehen in den mit ++ gekennzeichneten Zustand über.

Und hier die Verdrahtung des DFF2

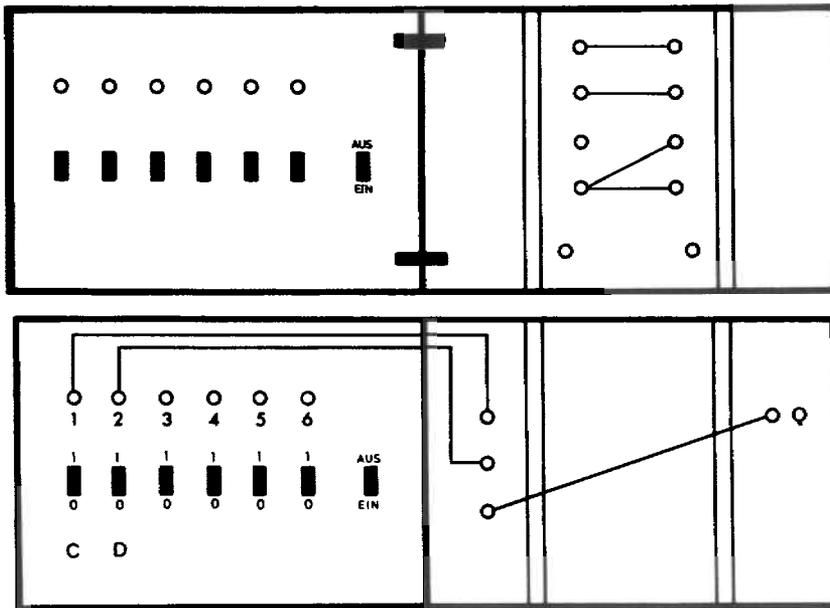
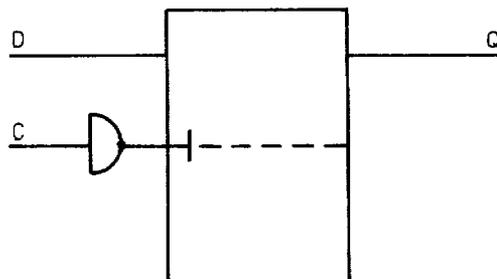


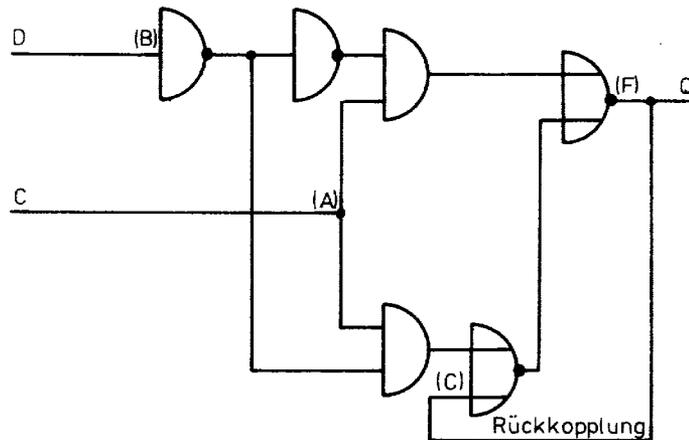
Abb. 68

Logiksymbol



D-Auffang-Flipflop 3 (DFF3)

Soll bei einem geöffneten D-Eingang (Takteingang C = 1) die Eingangs-
information 0 am Ausgang Q eine 1 hervorrufen und umgekehrt, muß vor
dem D-Eingang des DFF1 eine Negation geschaltet werden. Wir erhalten
dann das DFF3 mit folgendem Logikschaltplan:



Daraus ergibt sich die Funktionstabelle und die Programmierung für
einen Logik-Baustein.

Funktionstabelle DFF3

C	D	Q
0	0	0
0	1	0
1	0	1
1	1	0

Programmierung

(A)	(B)	(C)	(F)
C	D		Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0 ⁺	1
1	0	1 ⁺⁺	1
1	1	0 ⁺⁺	0
1	1	1 ⁺	0

- f1-C
- f2-C
- f3-1
- f4-0

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen,
dann sind die mit ⁺ versehenen Zustände nicht möglich. Sie
gehen in den mit ⁺⁺ gekennzeichneten Zustand über.

Und so sieht die Verdrahtung des DFF3 aus:

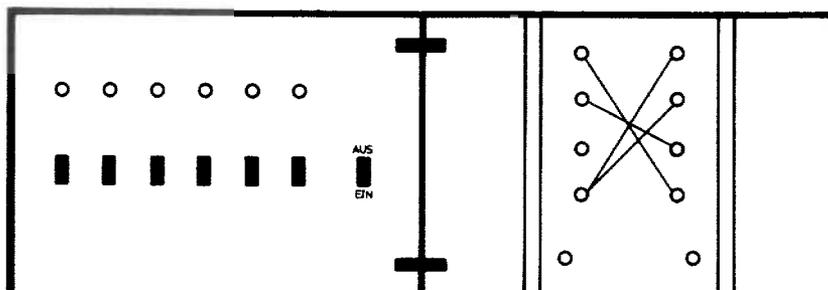


Abb. 69

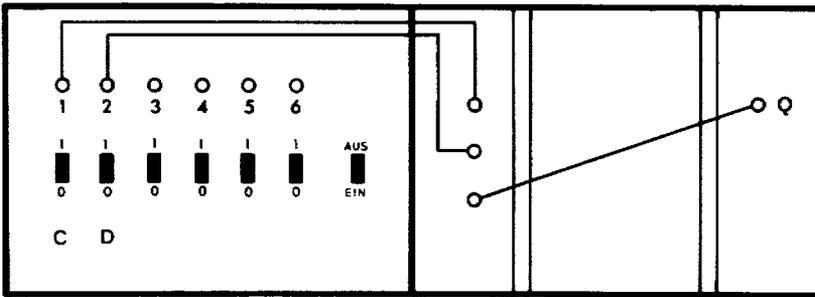
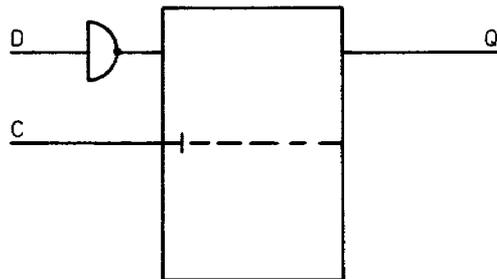


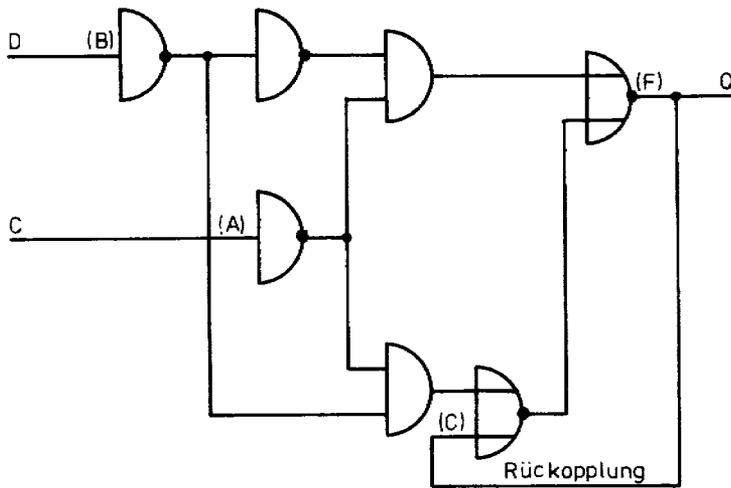
Abb. 69

Logiksymbol



D-Auffang-Flipflop 4 (DFF4)

Werden der D- und der C-Eingang des DFF1 invertiert, dann erhalten wir folgenden Logikschaltplan mit der dazugehörigen Funktionstabelle:



Funktionstabelle DFF4

C	D	Q
0	0	1
0	1	0
1	0	0
1	0	1

Programmierung

(A) C	(B) D	(C)	(F) Q	
0	0	0 ⁺	1	→ f1 = 1
0	0	1 ⁺⁺	1	
0	1	0 ⁺⁺	0	→ f2 = 0
0	1	1 ⁺	0	
1	0	0	0	→ f3 = C
1	0	1	1	
1	1	0	0	→ f4 = C
1	1	1	1	

Anmerkung: Ist die Rückkopplungsleitung von (F) nach (C) angeschlossen, sind die mit ⁺ versehenen Zustände nicht möglich. Sie gehen in den mit ⁺⁺ gekennzeichneten Zustand über.

Die Verdrahtung des so programmierten DFF4 sieht so aus:

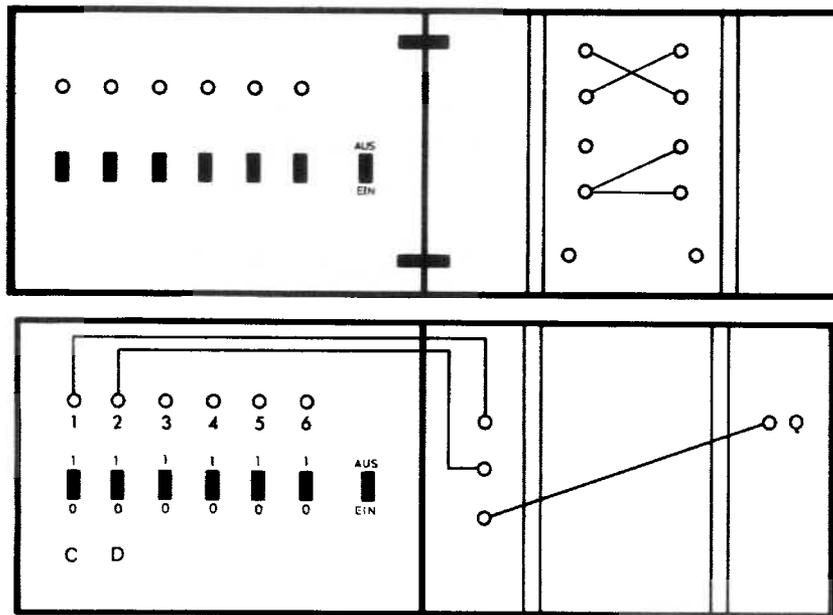
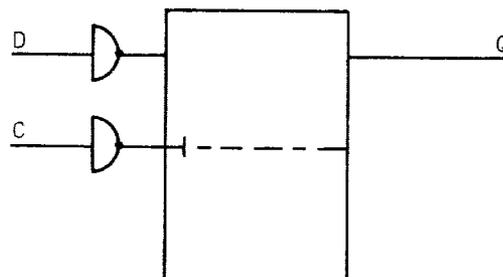


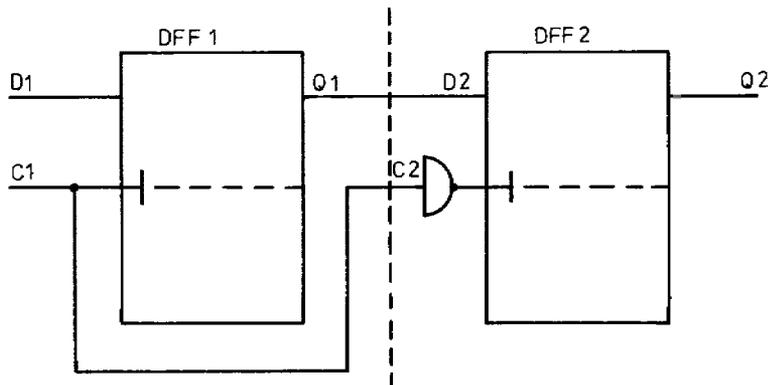
Abb. 70

Logiksymbol



11.3. 2-Speicher-Flipflop (Binärzähler)

In den Kapiteln 11.1. und 11.2. haben Sie das Speicher-Flipflop (RS-Flipflop) und das Auffang-Flipflop in verschiedenen Variationen kennengelernt. Es gibt jedoch noch eine weitere Flipflop-Schaltung, die ein sehr interessantes Verhalten zeigt: das sogenannte 2-Speicher-Flipflop (andere Bezeichnungen sind Zähl-Flipflop, Binäruntersetzter). Dieses Speicher-Flipflop besteht in seiner einfachsten Ausführung aus zwei unterschiedlichen D-Auffang-Flipflops, die, wie im Logikschema angegeben, miteinander verbunden sind.



Mit Hilfe der Funktionstabelle für das DFF1 und DFF2 können Sie leicht die Abhängigkeit des Ausgangssignals Q_2 von den Eingangsinformationen C_1 und D_1 erkennen.

Funktionstabelle DFF1

C_1	D_1	Q_1
0	0	0
0	1	1
1	0	0
1	1	1

Funktionstabelle DFF2

C_2	D_2	Q_2
0	0	0
0	1	1
1	0	0
1	1	1

Für den Fall, daß der Takteingang C_1/C_2 von 0 auf 1 geht, wird die an D_1 liegende Information zunächst vom DFF1 übernommen. Ein Beispiel: Ist $C_1/C_2 = 1$ und D_1 auch 1, dann ist auch $Q_1 = 1$. Dieser Wert kann vom DFF2 erst dann übernommen werden, wenn der Taktimpuls an C wieder von 1 auf 0 zurückgeht, da die Inversion des C_2 -Eingangs einen Einfluß des an D_2 liegenden Eingangssignals bei $C_2 = 1$ auf den Ausgang Q_2 verhindert. Untersuchen Sie diesen Sachverhalt anhand der nachfolgenden Schaltung.

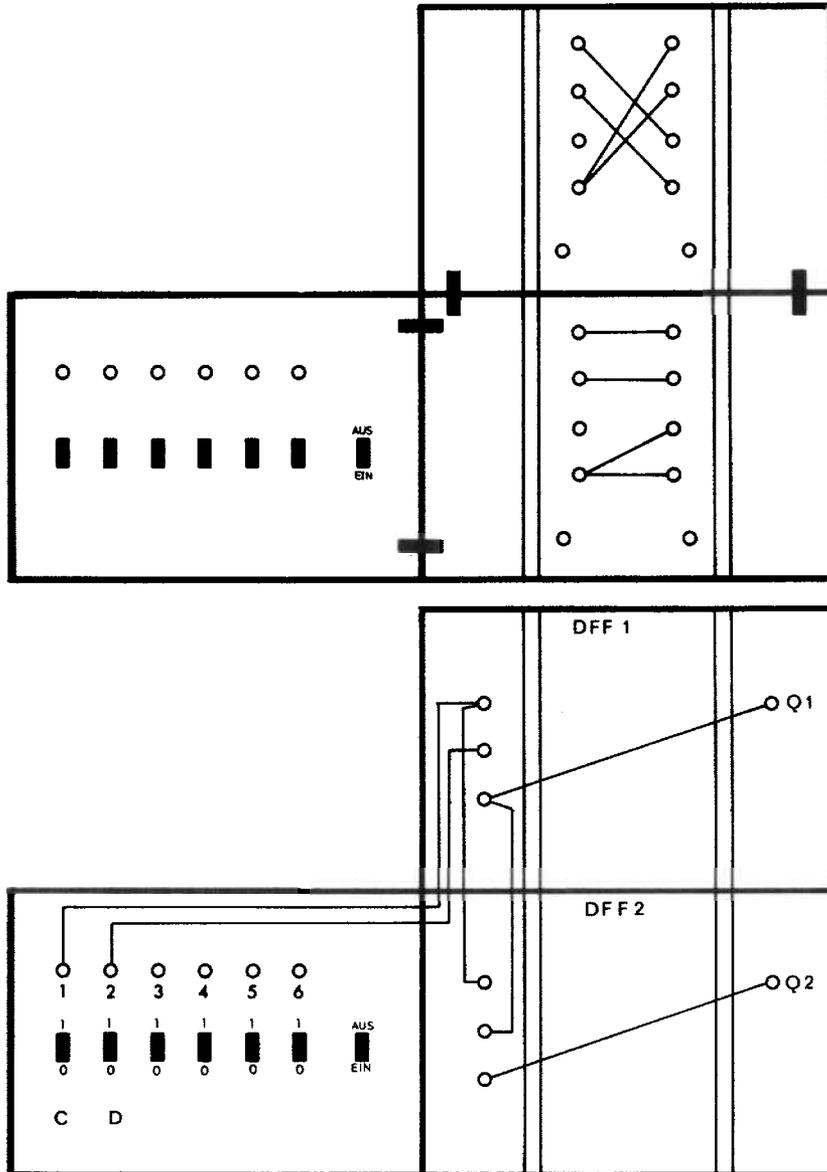
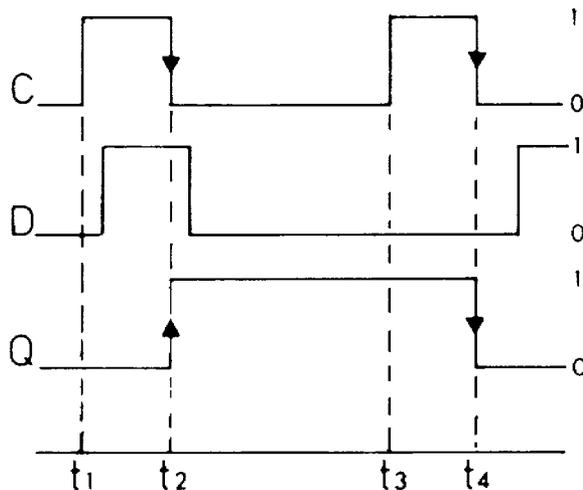


Abb.71

Da zunächst bei $C = 1$ das an D liegende Signal in den ersten Speicher (Master, Zwischenspeicher) gelangt und erst bei $C = 0$ vom zweiten D-Auffang-Flipflop (Slave, Hauptspeicher) übernommen wird, spricht man von einer Zwischenspeicherung des Signals. In einer Funktionstabelle ausgedrückt, sieht das Verhalten des 2-Speicher-Flipflops so aus:

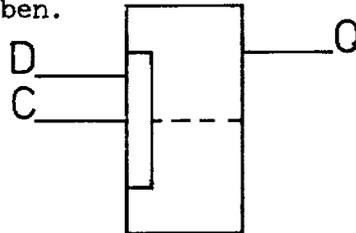
D	Q2
0	0
1	1

Bedingung: Nur in dem Augenblick, wenn $C = 1$ ist, beeinflusst die Information an D das Verhalten des Flipflops. Die Übergabe an den Ausgang Q2 erfolgt nur dann, wenn C wieder auf 0 zurückgeht. Dieser Sachverhalt läßt sich auch zeichnerisch darstellen:

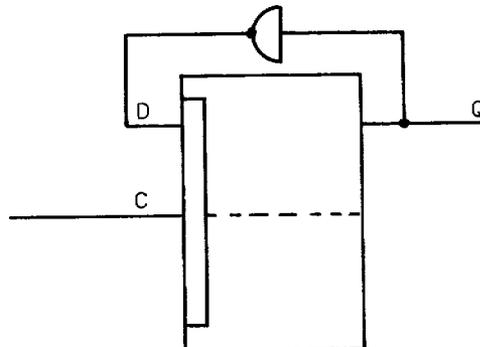


Wenn z.B. zum Zeitpunkt t_1 das an C liegende Signal auf 1 geht, wird der erste Speicher geöffnet. Nur die letzte im Zeitraum von t_1 bis t_2 am Eingang D liegende Information wird zwischengespeichert und zum Zeitpunkt t_2 an den Ausgang Q weitergegeben. Sie bleibt solange erhalten, bis bei t_3 der Taktimpuls wieder auf 1 geht und den ersten Speicher öffnet. Liegt eine Informationsänderung vor, so wird diese zur Zeit t_4 an den Ausgang Q weitergegeben.

Logiksymbol des
2-Speicher-
Flipflops



Die Zwischenspeicherung eines Signals wird überall dort notwendig, wo zur gleichen Zeit eine Information gespeichert und eine andere abgegeben werden soll. Bei der Realisierung einer Zählhaltung z.B. muß der Binäruntersetzter aus einem 2-Speicher-Flipflop aufgebaut sein. Dieser Baustein, der nur noch den Takteingang C besitzt, hat die Eigenschaft, daß bei jedem Taktimpuls das Ausgangssignal in den entgegengesetzten Zustand springt. Einen solchen Binäruntersetzter erhalten Sie, wenn Sie das Ausgangssignal des 2-Speicher-Flipflops invertieren und auf den D-Eingang zurückführen.



Durch die Rückführung des invertierten Q-Signals liegt immer eine dem Ausgangssignal entgegengesetzte Information an D, die beim nächsten Taktimpuls von der 0 auf 1 gehenden Taktflanke in den Zwischenspeicher gegeben und mit der aktiven Flanke (C geht von 1 auf 0) vom Hauptspeicher (Q) übernommen wird.

Sie können diese Vorgänge leichter verstehen, wenn Sie den nachfolgenden Binäruntersetzer aus 3 Logik-Bausteinen aufbauen.

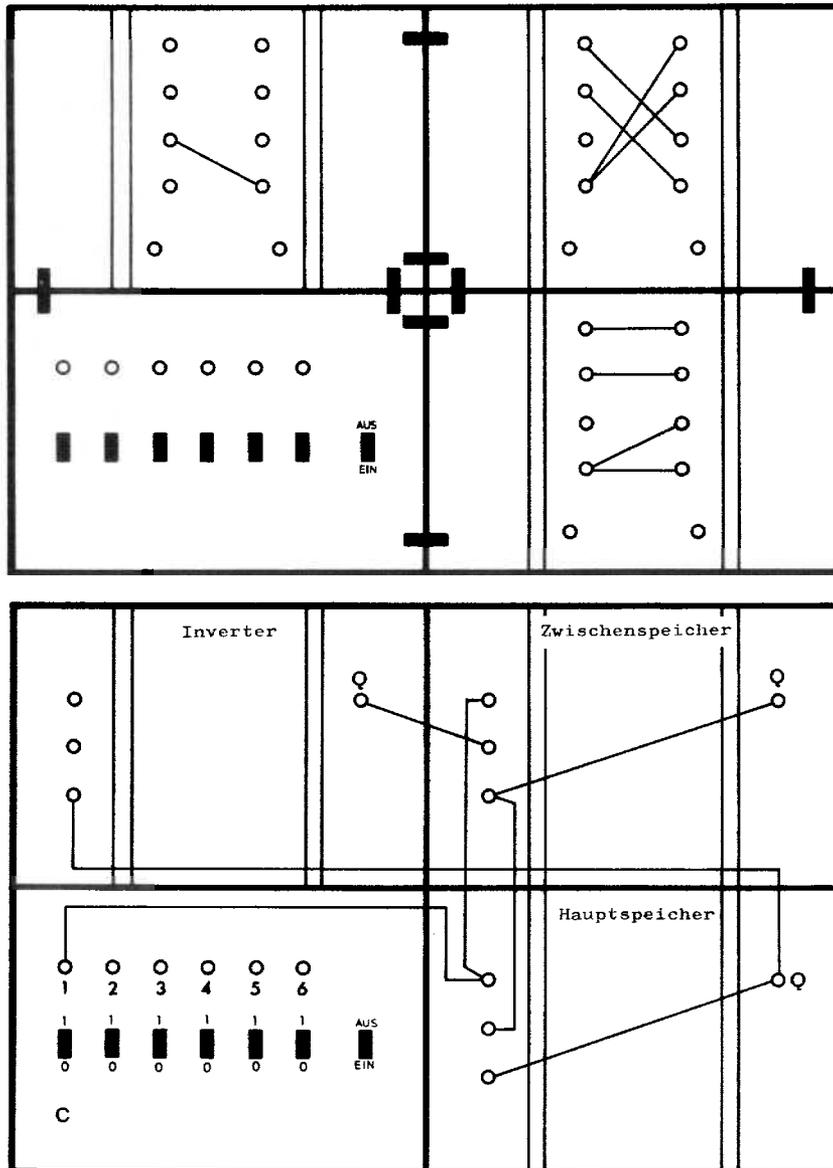
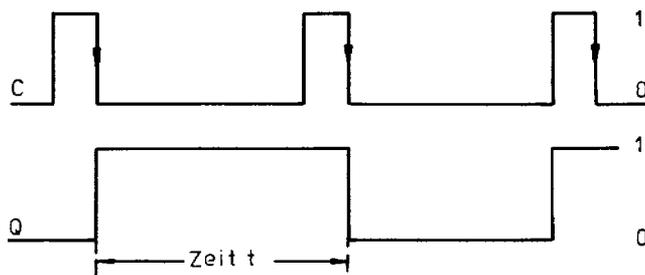


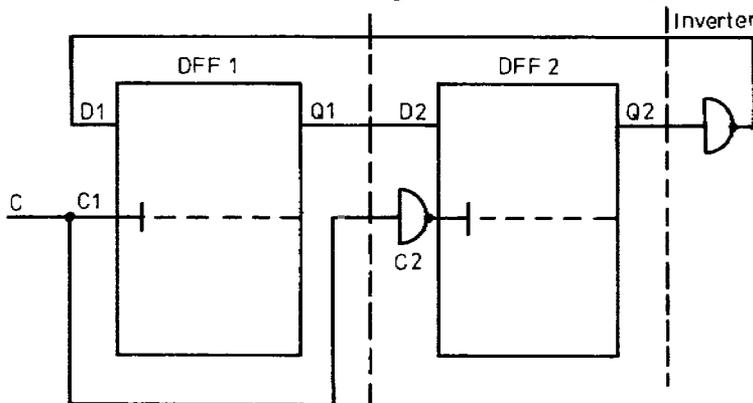
Abb.72

Schalten Sie den Taktschalter C von 0 auf 1. Sie erkennen, daß die Information am Ausgang des Inverters vom Zwischenspeicher übernommen wird und erst dann in den Hauptspeicher gelangt, wenn Sie C wieder auf 0 zurücksetzen. Wenn Sie nur die Anzeigelampe des Hauptspeichers betrachten, sehen Sie, daß der Ausgangszustand des Hauptspeichers (Ausgang Q) nur umkippt, wenn der Taktimpuls C von 1 auf 0 geht.



Aus der Eingangsinformation 0 und 1 (2 Zustände in der Zeit t) ist entweder ein 0- oder ein 1-Signal (1 Zustand in der Zeit t) geworden. 2 Taktimpulse ergeben also einen Ausgangsimpuls. Das Signal ist also durch den Faktor 2 geteilt, oder anders ausgedrückt, um den Faktor 2 unter-
setzt worden.

Dieser Zähl-Flipflop soll nun mit 2 Logik-Bausteinen realisiert werden. Betrachten Sie dazu nochmals den logischen Schaltungsaufbau.



Der zusätzlich in die Leitung $D1 - Q2$ geschaltete Inverter kann entfallen, wenn Sie das DFF1 durch ein DFF3 ersetzen, da dieses Flipflop einen invertierten D-Eingang besitzt. Sie erhalten dann einen Binär-
untersetzer, der nur aus 2 Bausteinen besteht.

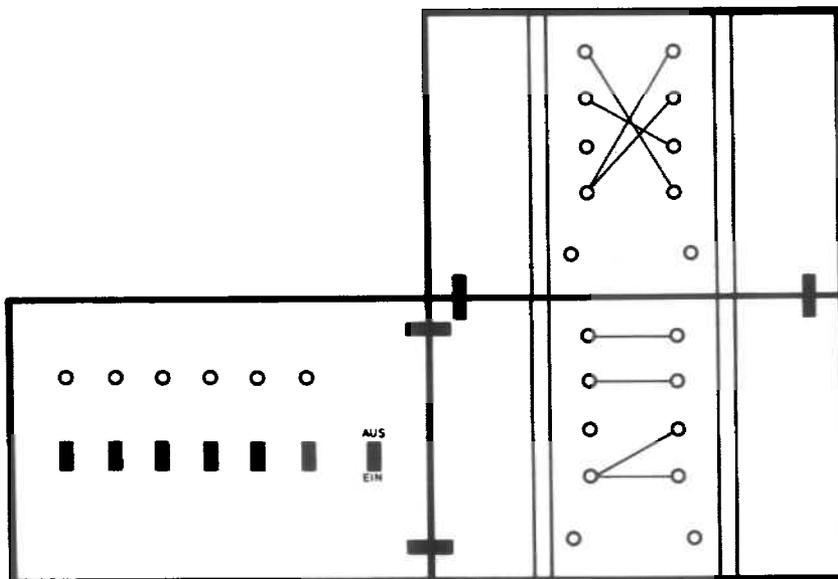


Abb.73

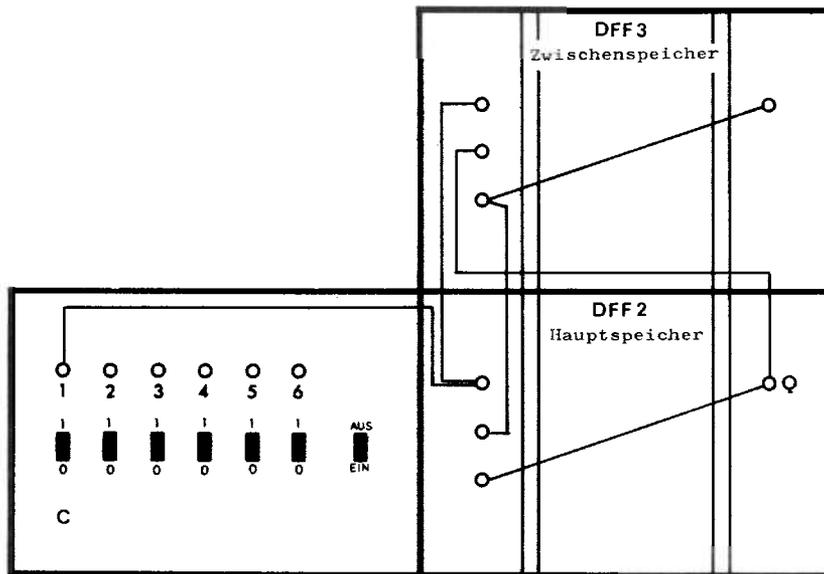
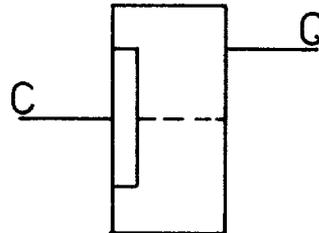


Abb. 73

Logiksymbol des
Binäruntersetzers

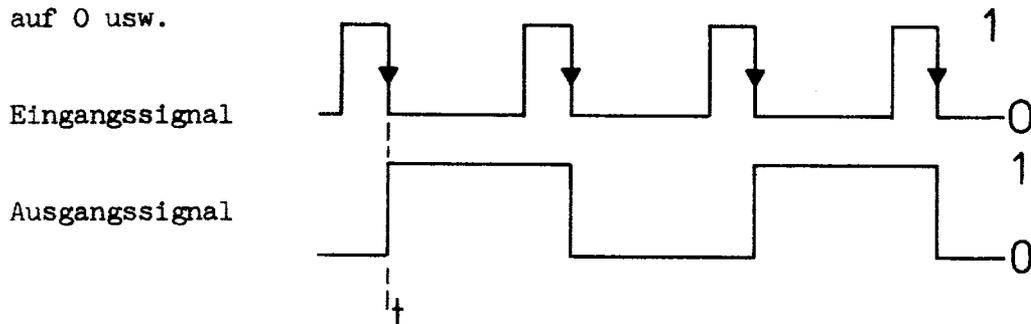


Im folgenden Beispiel finden Sie eine andere Schaltungsvariante für einen Binärzähler.

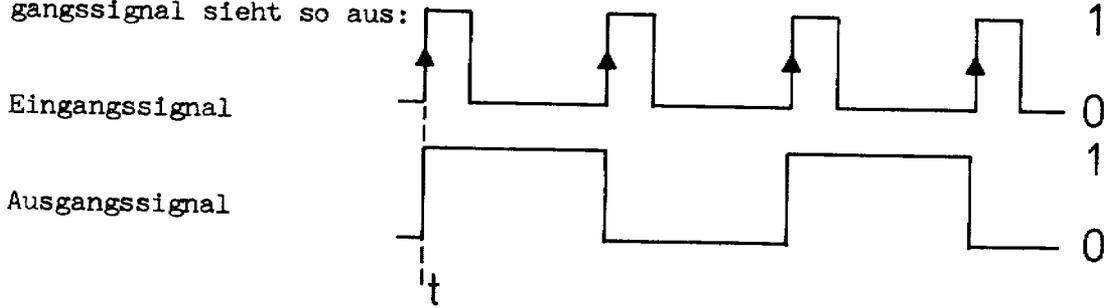
Generator mit Frequenzuntersetzter

Die Frequenz des Generators (Kapitel 10) läßt sich durch einen außen anzubringenden Kondensator verändern. Je größer die Kapazität wird, desto kleiner wird die Frequenz. Diese Frequenz kann mit einem Binärzähler noch zusätzlich verringert werden, und zwar im Verhältnis 1:2. Der Binärzähler wirkt also wie ein Frequenzuntersetzter.

Nach einem Eingangsimpuls (vom Generator) liegt der Ausgang des Binärzählers auf 0. Nach zwei Eingangsimpulsen liegt er auf 1, nach 3 wieder auf 0 usw.



Eine andere zusätzliche Zuordnung zwischen dem Eingangs- und dem Ausgangssignal sieht so aus:



Bei der ersten Möglichkeit trifft der Aufstieg des Ausgangssignals im Zeitpunkt t mit dem Abfall des Eingangssignals zeitlich zusammen. Bei dem zweiten Beispiel treffen zum Zeitpunkt t₁ der Anstieg des Eingangs- und des Ausgangssignals zeitlich aufeinander.

Die Programmierung der beiden Logikbausteine zum Binärzähler wurde für die erste Möglichkeit gewählt.

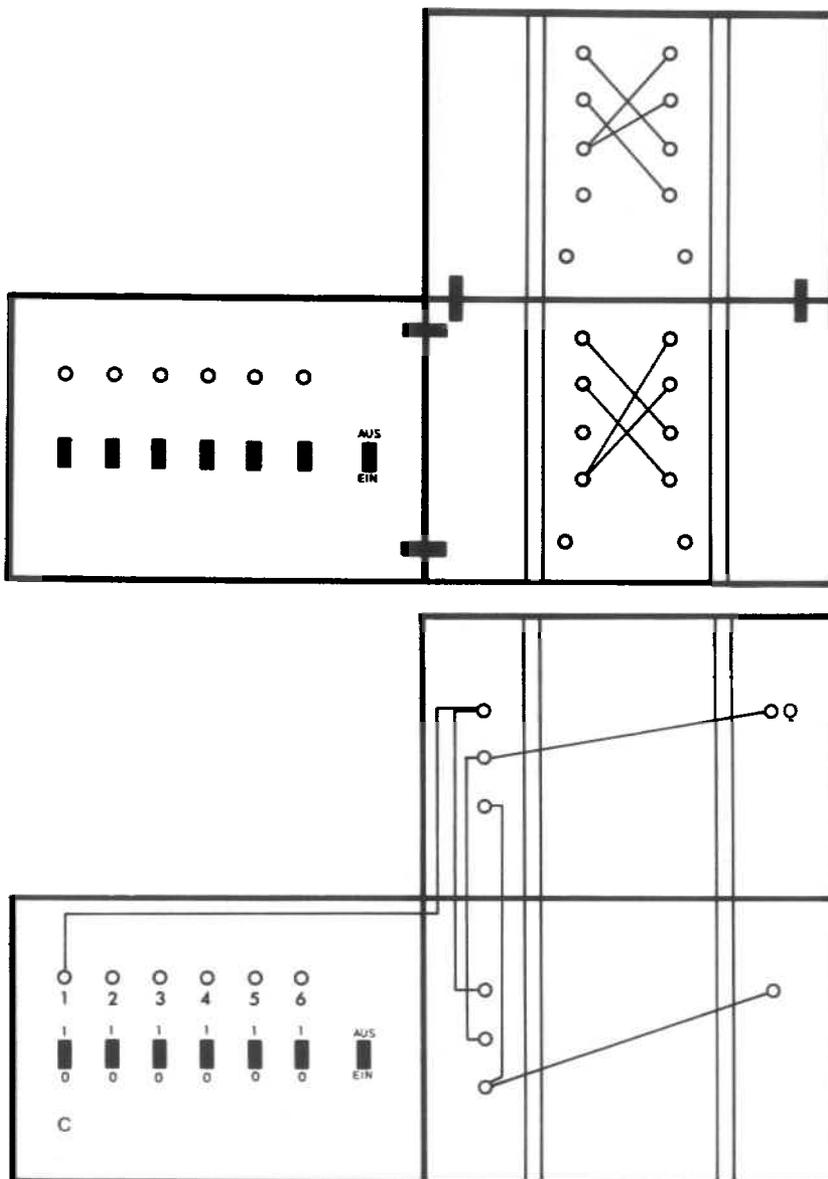


Abb. 74

Die Funktion des Binärzählers wird aus den Funktionstabellen der beiden Logik-Bausteine deutlich.

A	B	C	F1	F2
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

Aus der Verdrahtung der Logik-Bausteine können Sie erkennen, daß

der Ausgang F1 mit B und

der Ausgang F2 mit C verbunden sind.

Die Eingangsvariable A entspricht dem Eingang des Binärzählers.

Dann können folgende Zustände auftreten:

Wenn sein Eingang 0, der Ausgang F1 = 1 und B = 0 sind, so ist dieser Zustand nicht stabil. Denn

$$F1 \neq B$$

$$F2 = C$$

Wird B = F1 = 1, so wird der Zustand des Binärzählers stabil. Denn jetzt ist

$$F1 = B \text{ und}$$

$$F2 = C.$$

Wird der Eingang dann 1, so gilt

A	B	C	F1	F2
1	1	0	1	1

Aus diesem nicht stabilen Zustand - $F2 \neq C$ - wird der stabile Zustand

A	B	C	F1	F2
1	1	1	1	1

Man kann so das Ausgangssignal F1 als Funktion des Eingangssignals ermitteln.

Verbinden Sie nun den Generator nach der folgenden Abbildung mit dem Binärzähler.

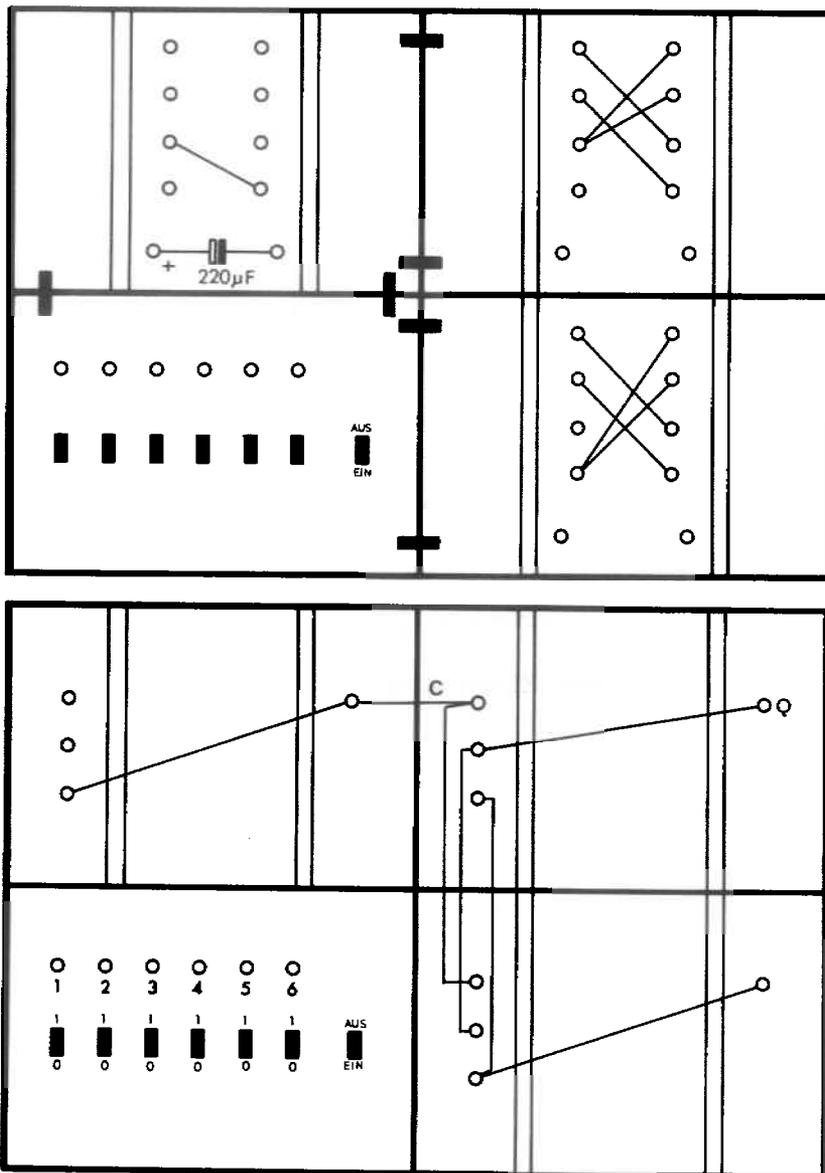


Abb.75

Sie erkennen, daß die Ausgangsfrequenz halb so groß ist wie die Frequenz des Generators.

Wenn Sie den Ausgang des Binäruntersetzers mit dem Eingang eines weiteren Zähl-Flipflops verbinden, können Sie das Ausgangssignal ein weiteres Mal untersetzen.

12. Das Dualsystem

12.1. Einführung

Computer haben merkwürdige Rechengewohnheiten - wenigstens kommt es uns so vor, wenn wir sie neu kennenlernen. So rechnen Computer z.B. mit dem einfachsten Zahlensystem, das es gibt, mit dem Dualsystem (Zweiersystem; duo (lateinisch): zwei; dualis: zweifach). Das Dualsystem kennt nur 2 Ziffern, nämlich die Ziffern

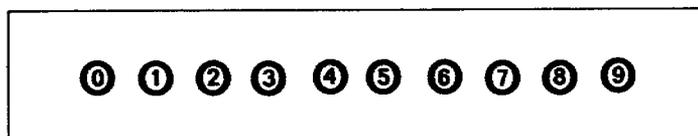


Ein ganzes Zahlensystem aus nur zwei Ziffern? - Ja, tatsächlich! Und Computer rechnen damit.

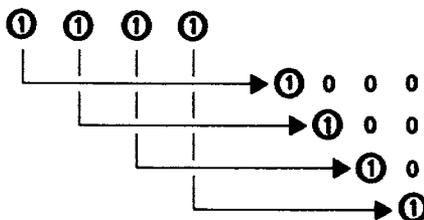
Wenn Sie diese Broschüre nicht nur überfliegen, können auch Sie im Dualsystem rechnen, so sicher wie mit dem Ihnen vertrauten Dezimalsystem. (Fast so sicher.)

Halten wir uns zunächst noch einmal die Prinzipien vor Augen, nach denen das Dezimalsystem aufgebaut ist. Wir verstehen dann das Dualsystem besser, denn es hat die gleichen Aufbauprinzipien als Grundlage.

Das Dezimalsystem (Zehnersystem; decem (lateinisch): zehn; decimalis: zehnfach) kennt 10 verschiedene Ziffern:



Der Wert einer Dezimalziffer hängt nicht nur von ihr selbst ab, sondern auch von der Position, die sie innerhalb einer Zahl einnimmt. Hier ein Beispiel, die vierstellige (also aus 4 Ziffern bestehende) Zahl 1111:



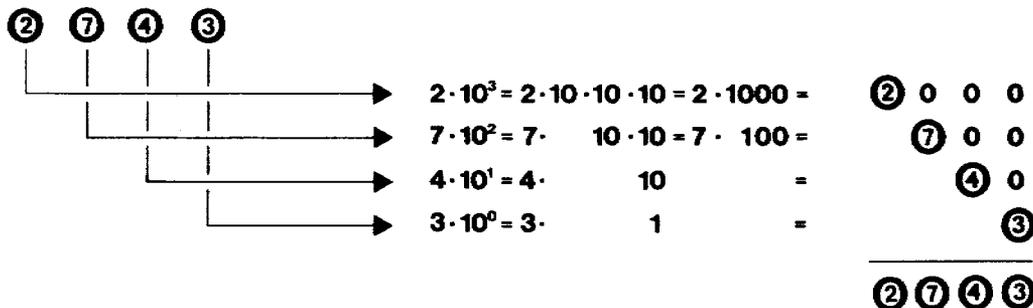
Von links beginnend hat also die erste 1 einen Wert von 1000, die zweite 1 einen Wert von 100, die dritte 1 hat einen Wert von 10 und die vierte 1 schließlich ist 1 wert. Jede 1 ist also 10-mal soviel wert wie die rechts neben ihr stehende 1. Mit anderen Worten: Das Dezimalsystem hat die Basis 10.

Das Aufbauprinzip des Dezimalsystems können wir uns an einer Zehnerpotenztafel übersichtlich veranschaulichen.

Für den Fall, daß es Ihnen nicht mehr geläufig ist: z.B. $10^3 = 3$. Potenz von 10; Basis: 10; Hochzahl (= Exponent): 3. Und noch etwas: Jede Zahl hoch 0 ist gleich 1; also auch $10^0 = 1$; $2^0 = 1$.

Ziehen wir gleich noch ein konkretes Beispiel hinzu, die Zahl 2743, und "errechnen" wir anhand der Zehnerpotenztafel den Gesamtwert dieser Zahl:

10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1



Die Gesamtzahl ist also die Summe der Werte der einzelnen Ziffernstellen (Ziffer mal Stellenwert).

Soweit die kleine Rekapitulation des vertrauten Dezimalsystems.

Nun zur Zahlenwelt des Computers, zum Dualsystem. - Warum rechnen Computer mit Dualzahlen? - Ganz einfach deshalb, weil es technisch am einfachsten ist. Man unterscheidet bei elektromechanischen und elektronischen Bauteilen, die in einem Computer verwendet werden, nur zwischen zwei verschiedenen Zuständen: Durch einen Draht fließt entweder Strom oder es fließt kein Strom; ein Schalter (z.B. ein Transistor) ist entweder offen oder geschlossen; ein Magnetring ist entweder rechts herum magnetisiert (Richtung des magnetischen Nordpols) oder links herum usw. Wollten wir Computer im Dezimalsystem rechnen lassen, so müßten wir 10 verschiedene Zustände ermöglichen (beispielsweise 0, 1, 2, ... 9 Volt), was zwar technisch möglich, aber erheblich komplizierter wäre.

Von den beiden verschiedenen Zuständen nennen wir den einen einfach 0 und den anderen 1. Damit haben wir die beiden Ziffern des Dualsystems. Wenn wir uns im folgenden mit dem Dualsystem befassen, wollen wir dabei das Dezimalsystem immer zum Vergleich und zur klaren Orientierung hinzuziehen, da es nun einmal das uns vertrauteste Zahlensystem ist.

Lösung

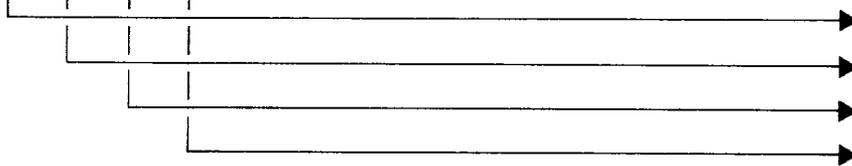
2^3	2^2	2^1	2^0
8	4	2	1

				0	①
			1		①
		1	0		②
		1	1		③
	1	0	0		④
	1	0	1		⑤
	1	1	0		⑥
	1	1	1		⑦
1	0	0	0		⑧
1	0	0	1		⑨

Welcher Dezimalzahl entspricht die Dualzahl 1101? - Errechnen Sie die Dezimalzahl:

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

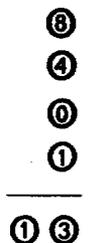
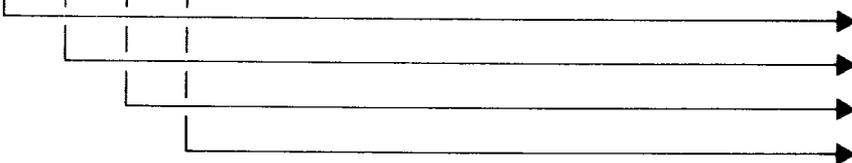
1 **1** 0 **1**



Und hier die Lösung:

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

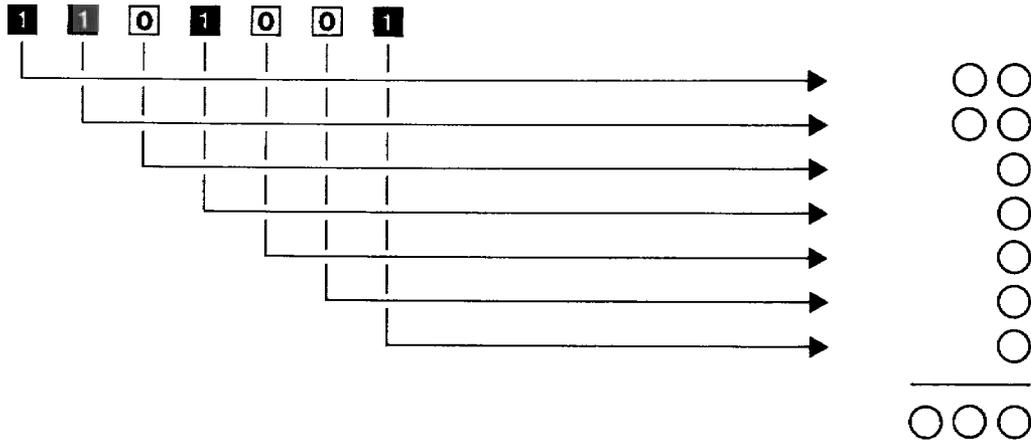
1 **1** 0 **1**



Anders ausgedrückt: $1101_2 = 13_{10}$

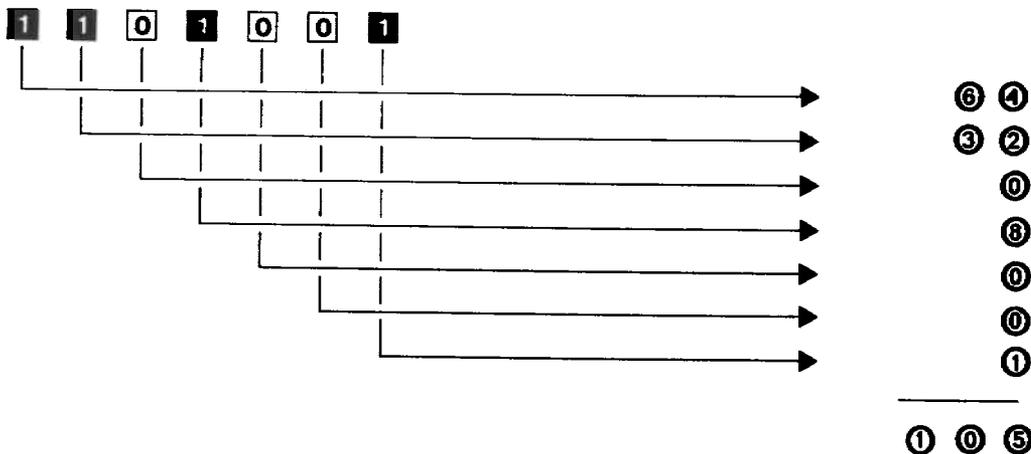
Wie heißt die Dezimalzahl, die der Dualzahl 1 1 0 1 0 0 1 entspricht?

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



So ist es richtig:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



Also: $1101001_2 = 105_{10}$

Und wie wird nun umgekehrt zu einer Dezimalzahl die entsprechende Dualzahl gefunden? Z.B :

$$86_{10} = ?_2$$

Sollten Sie sich die Umrechnung schon zutrauen, obwohl wir sie noch nicht behandelt haben, dann versuchen Sie es doch einmal. Benutzen Sie dazu die Zweierpotenzentabelle:

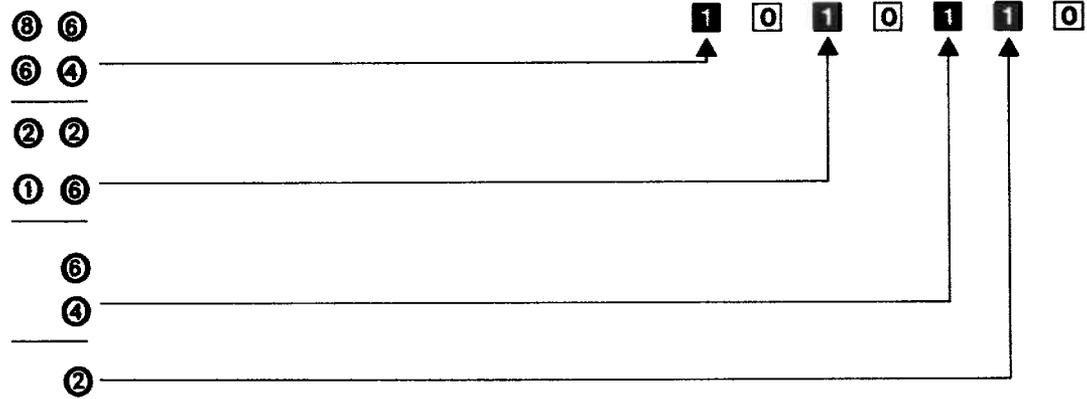
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Das Ergebnis: $86_{10} = 1010110_2$

Haben Sie das Ergebnis selbst gefunden? - Wenn nein, macht es nichts.

Wenn ja, um so beachtlicher!

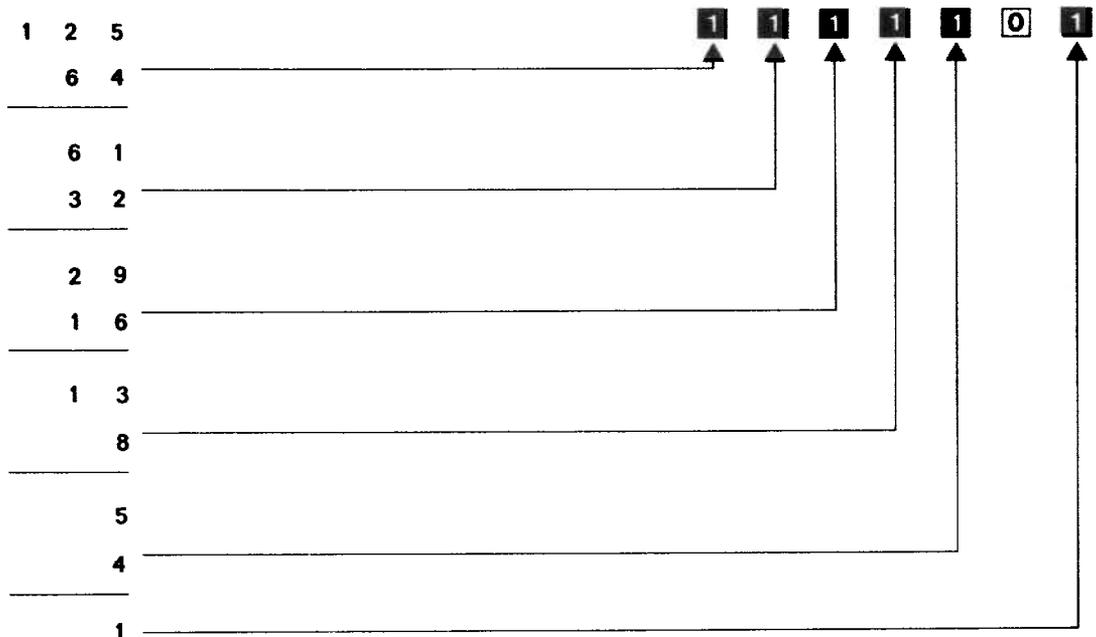
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



Zunächst wird die größte Zweierpotenz festgestellt, die in 86 enthalten ist, das ist 64. Also wird an der Wertstelle 64 eine 1 notiert. Bleibt ein Rest von 22; 32 ist nicht in 22 enthalten; deshalb wird eine 0 für die Wertstelle 32 notiert. Aber 16 ist in 22 enthalten; also eine 1 an die Wertstelle 16. $22 - 16 = 6$; 6 enthält nullmal die 8, aber einmal die 4 und einmal die 2, womit der Rest untergebracht wäre und für die verbleibende Wertstelle 1 nur noch eine 0 zu notieren bliebe.

Ermitteln Sie nun bitte die Dualzahl zur Dezimalzahl 125:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1



Damit wollen wir die Umwandlung von Dezimalzahlen in Dualzahlen und von Dualzahlen in Dezimalzahlen abschließen.

Sie kennen jetzt den Aufbau des Dualsystems und staunen nachträglich vielleicht, wie klar dieses System im Grunde ist.

Abschließend noch ein wichtiger Begriff:

Eine duale Ziffernstelle - egal, ob sie die Ziffer 0 oder 1 enthält - wird auch Bit genannt. Bit ist eine Abkürzung des englischen Ausdrucks binary diget, was binäre Ziffer - duale Ziffer bzw. Ziffernstelle heißt. Ein Bit ist auch jedes technische Element, das die Zustände 0 oder 1 annehmen kann. So ist z.B. das Speicher-Flipflop ein Speicherelement, das ein Bit speichern kann, und dieses Bit kann entweder 0 oder 1 sein.

Wird Bit als Maßeinheit verwendet, so wird es klein geschrieben: bit. Die Zahl 1001 beispielsweise ist 4 bit groß. Das ist wiederum gleichbedeutend mit: Die Zahl 1001 besteht aus 4 Bits.

12.2. Zahlenraten

Fertigen Sie sich fünf Karten an, die die Zahlen 1 - 31 wie folgt enthalten:

Karte 1			
3	17	25	13
7	29	23	9
11	21	31	5
15	27	19	1

Karte 2			
3	14	7	27
18	31	23	15
6	26	11	19
10	22	30	2

Karte 3			
5	14	7	20
28	22	30	12
6	31	23	29
13	15	21	4

Karte 4			
9	28	12	24
15	25	31	10
11	29	30	14
26	13	27	8

Karte 5			
17	25	19	24
22	29	27	21
28	26	31	18
20	30	23	16

Bauen Sie nun aus Ihren Logik-Bausteinen fünf Speicher-Flipflops und verbinden Sie diese Speicher-Flipflops mit der Eingabeeinheit gemäß Abbildung.

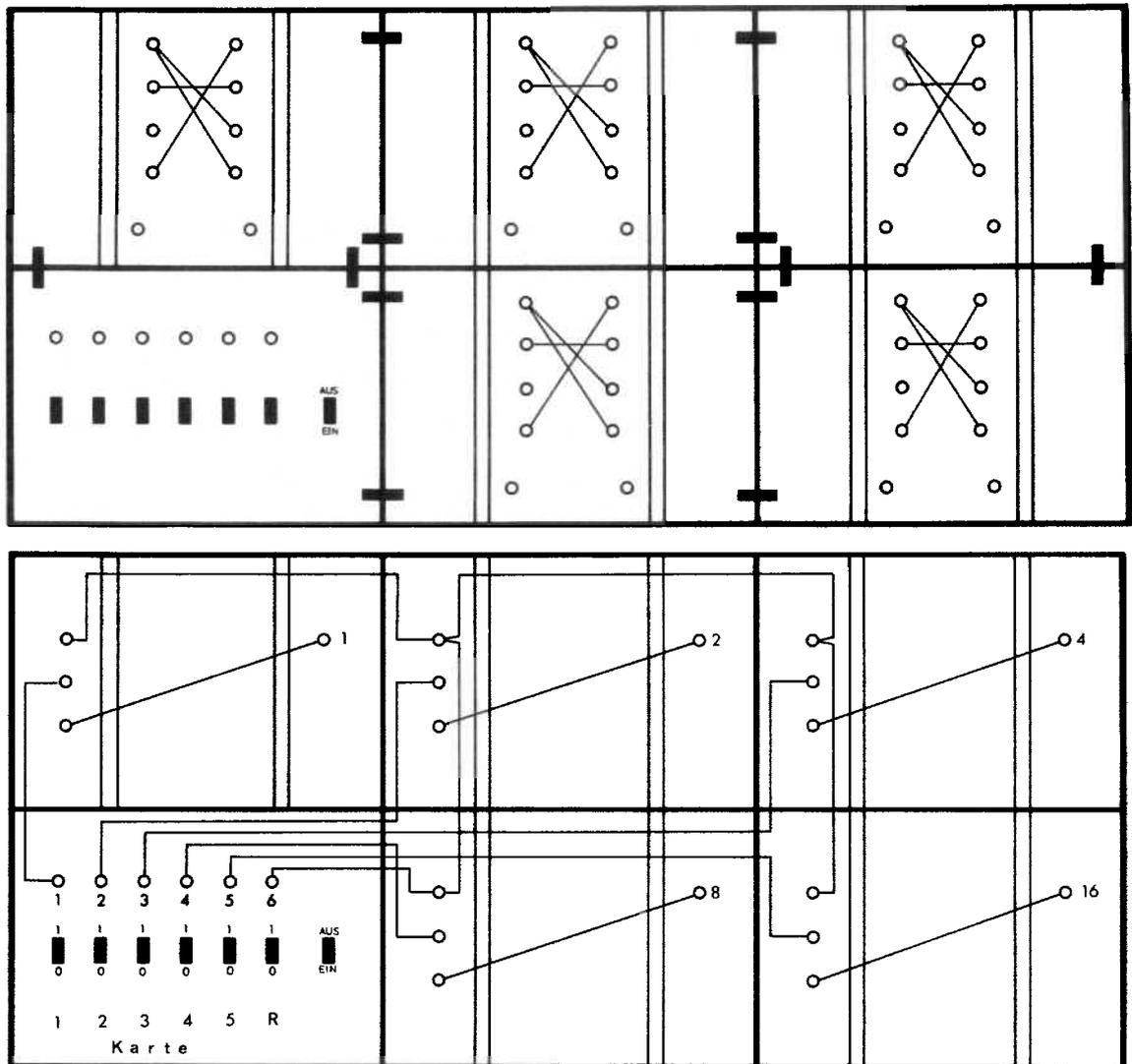


Abb.76

Jedes Speicher-Flipflop hat einen eigenen Set-Schalter, alle fünf Speicher-Flipflops haben zusammen einen Reset-Schalter. Den fünf Speicher-Flipflops ordnen wir die Wertigkeiten 1, 2, 4, 8 und 16 zu, entsprechend den Stellenwertigkeiten im Dualsystem. Dabei spielt es keine Rolle, daß wir die Stellenwertigkeiten von oben nach unten und nicht, wie im Kapitel über das Dualsystem, von rechts nach links aufgeführt haben.

Bitten Sie nun einen Mitspieler - am besten jemanden, der weder Ihren Computer noch das Dualsystem kennt - sich eine Zahl zwischen 1 und 31 zu denken. Legen Sie dem Mitspieler dann der Reihe nach die Karten 1 bis 5 vor und fragen ihn jeweils, ob die gezeigte Karte die gedachte Zahl enthält. Antwortet der gefragte Mitspieler mit JA, so wird kurz der Schalter, der der entsprechenden Karte zugeordnet ist, auf 1 gesetzt (set). Nach Abfrage aller Karten wird die gedachte Zahl von dem Computer dual angezeigt. Wie Sie die angezeigte Zahl ins Dezimalsystem übertragen, haben

Sie bereits kennengelernt. Das Ergebnis ist jedenfalls die vom Mitspieler gedachte Zahl.

Vielleicht führen Sie das Experiment zunächst einmal mit sich allein durch. Sicher werden auch Sie erstaunt sein, wieso Sie durch dieses Verfahren auf die Lösung kommen können. Und dies ist das Geheimnis der Zahlenkarten:

Auf den Karten sind die Zahlen so geordnet, daß alle diejenigen, die in der dualen Darstellung in der ersten Stelle eine 1 haben, auf Karte 1 abgebildet sind. Diejenigen Zahlen, die in der dualen Darstellung eine 1 in der zweiten Stelle haben, sind auf Karte 2 enthalten. Alle Zahlen mit einer 1 an der dritten dualen Stelle sind auf Karte 3 usw. Wenn nun bekannt ist, auf welchen Karten die gedachte Zahl vorhanden ist, so ist die duale Darstellung bekannt und kann ins Dezimalsystem umgerechnet werden.

dezimal	dual	
1	1	
2	1 0	
3	1 1	
4	1 0 0	
5	1 0 1	
6	1 1 0	
7	1 1 1	
8	1 0 0 0	
9	1 0 0 1	
10	1 0 1 0	
11	1 0 1 1	
12	1 1 0 0	
13	1 1 0 1	
14	1 1 1 0	
15	1 1 1 1	
16	1 0 0 0 0	
17	1 0 0 0 1	
18	1 0 0 1 0	
19	1 0 0 1 1	
20	1 0 1 0 0	
21	1 0 1 0 1	
22	1 0 1 1 0	
23	1 0 1 1 1	
24	1 1 0 0 0	
25	1 1 0 0 1	
26	1 1 0 1 0	
27	1 1 0 1 1	
28	1 1 1 0 0	
29	1 1 1 0 1	
30	1 1 1 1 0	
31	1 1 1 1 1	
	5 4 3 2 1	Stelle

12.3. Rechnen mit Dualzahlen ist denkbar einfach, denn es gibt ja nur die beiden Ziffern 0 und 1. Nur weil es Ihnen ungewohnt ist, werden Sie vielleicht anfangs etwas mehr Mühe haben als beim Rechnen mit Dezimalzahlen (nebenbei: denken Sie immer an die Lesart der Dualzahlen, also z.B.: $1 + 1 = 10$; Eins + Eins = Eins-Null).

Addition

Bevor wir Dualzahlen addieren, erinnern wir uns zuerst an die Dezimaladdition. Zwei oder mehr Summanden werden zu einer Summe addiert. Wenn die Summenbildung auf einer Dezimalstelle den Wert 9 überschreitet, entsteht ein Übertrag auf die linke Nachbarstelle; bei dieser Nachbarstelle wird der Übertrag mitaddiert, ein Beispiel:

$$\begin{array}{r}
 1 5 7 \text{Summand A} \\
 + 5 4 3 \text{Summand B} \\
 \hline
 1 1 \text{Überträge} \\
 \hline
 7 (1) 0 (1) 0 \text{Summe} \\
 \hline
 \hline
 \end{array}$$

Bei der Addition von Dualzahlen wird entsprechend verfahren. Ein Übertrag auf die linke Nachbarstelle entsteht allerdings bereits, wenn die Summenbildung auf einer Dualstelle den Wert 1 überschreitet.

16	8	4	2	1		
----	---	---	---	---	--	--

$$\begin{array}{r}
 1 1 1 0 \text{Summand A (14)} \\
 + 1 1 0 1 \text{Summand B (13)} \\
 \hline
 1 1 \text{Überträge} \\
 \hline
 = 1 (1) 1 (1) 0 1 1 \text{Summe (27)} \\
 \hline
 \hline
 \end{array}$$

Nun versuchen Sie doch einmal, die folgende Aufgabe selbst zu lösen. Sehen Sie sich bitte nicht gleich die nachfolgende Lösung an, sondern probieren Sie erst einmal selbst. Zunächst können Sie Ihre eigene Lösung dadurch kontrollieren, indem Sie zu den Dualzahlen die Dezimalzahlen errechnen und die Summen vergleichen:

32	16	8	4	2	1	
----	----	---	---	---	---	--

$$\begin{array}{r}
 1 1 1 1 \text{Summand A} \\
 + 1 1 0 1 0 \text{Summand B} \\
 \hline
 \text{Überträge} \\
 \hline
 = \text{Summe} \\
 \hline
 \hline
 \end{array}$$

12.4. Addition

Im Kapitel 12.3. haben Sie das Rechnen mit Dualzahlen gelernt. Da das Dualsystem nur die Werte 0 und 1 kennt, lassen sich auch Ihre elektronischen Logik-Bausteine zu einem kleinen Rechner programmieren. Sollen z.B. 2 einstellige Dualzahlen mit Logik-Bausteinen addiert werden, so müssen Sie sich, um die Programmierung zu finden, zunächst noch einmal die Rechenregeln der Addition ansehen.

Im Dualsystem ergibt

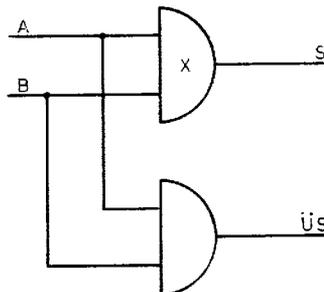
1. Dualzahl	+	2. Dualzahl	=	Summe	und Übertrag
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

Wird die erste Dualzahl mit A und die zweite mit B bezeichnet, läßt sich aus diesem logischen Zusammenhang eine Funktionstabelle für die Summe S und den Übertrag ÜS aufstellen.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

A	B	ÜS
0	0	0
0	1	0
1	0	0
1	1	1

Sie erkennen, daß bei der Addition die Summe S mit der Exklusiv-ODER-Funktion und der Übertrag ÜS mit der UND-Funktion gebildet werden können.



Diese einfachste Art eines Rechners nennt man Halbaddierer. Die Programmierung der zwei Logik-Bausteine läßt sich nach dem Ihnen bekannten Schema (Kapitel 7) leicht finden.

Programmierung Baustein S

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Programmierung Baustein ÜS

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Schaltung des Halbaddierers

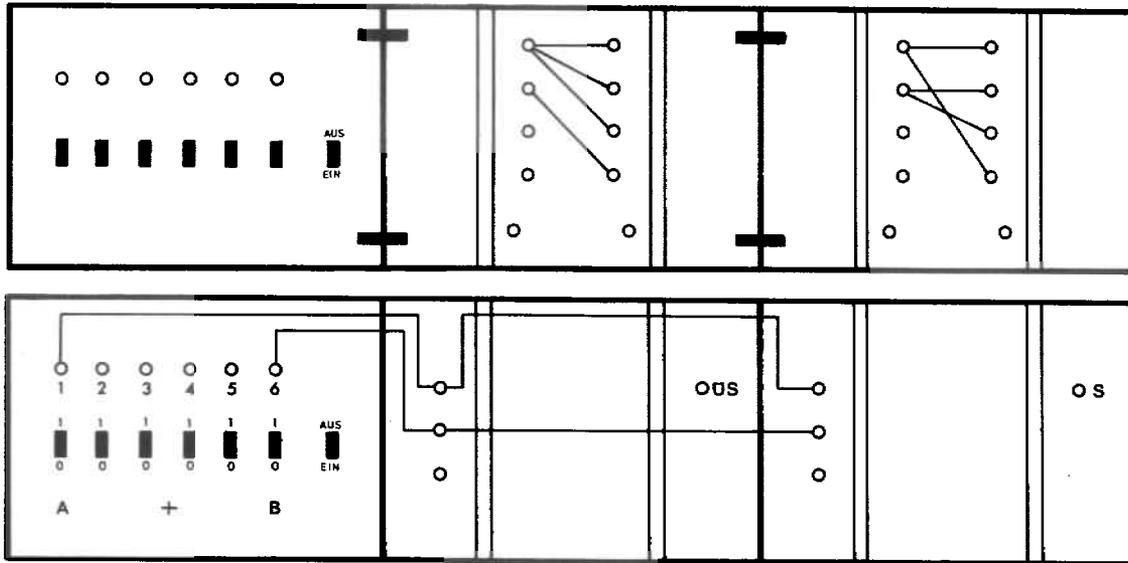


Abb.77

Beispiel:

Addieren Sie mit der Schaltung die Dualzahlen $0 + 1$, $1 + 1$, $1 + 0$, $0 + 0$ und überprüfen Sie das Ergebnis.

Wie Sie festgestellt haben, lassen sich mit dieser Schaltung jedoch nur einstellige Dualzahlen addieren.

Wenn beispielsweise neben den Dualziffern A und B auch noch der Übertrag \ddot{U} aus dem vorangegangenen Rechengang berücksichtigt werden soll, muß der Halbaddierer zu einem Volladdierer ausgebaut werden. Mit Hilfe der Funktionstabellen für die Summe S und den Übertrag $\ddot{U}S$ läßt sich die Programmierung für den Volladdierer finden.

Funktionstabelle für Summe S

1. Dualziffer	+	2. Dualziffer	+	Übertrag	Summe
A		B		\ddot{U}	S
A		B		C	F
0		0		0	0
0		0		1	1
0		1		0	1
0		1		1	0
1		0		0	1
1		0		1	0
1		1		0	0
1		1		1	1

→ $f1 = C$

→ $f2 = \bar{C}$

→ $f3 = \bar{C}$

→ $f4 = C$

Funktionstabelle für den Übertrag ÜS

1. Dualziffer + 2. Dualziffer + Übertrag			Übertrag
A	B	Ü	ÜS
A	B	C	F
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

- f1 = 0
- f2 = C
- f3 = C
- f4 = 1

Und so sieht die Verdrahtung des Volladdierers aus:

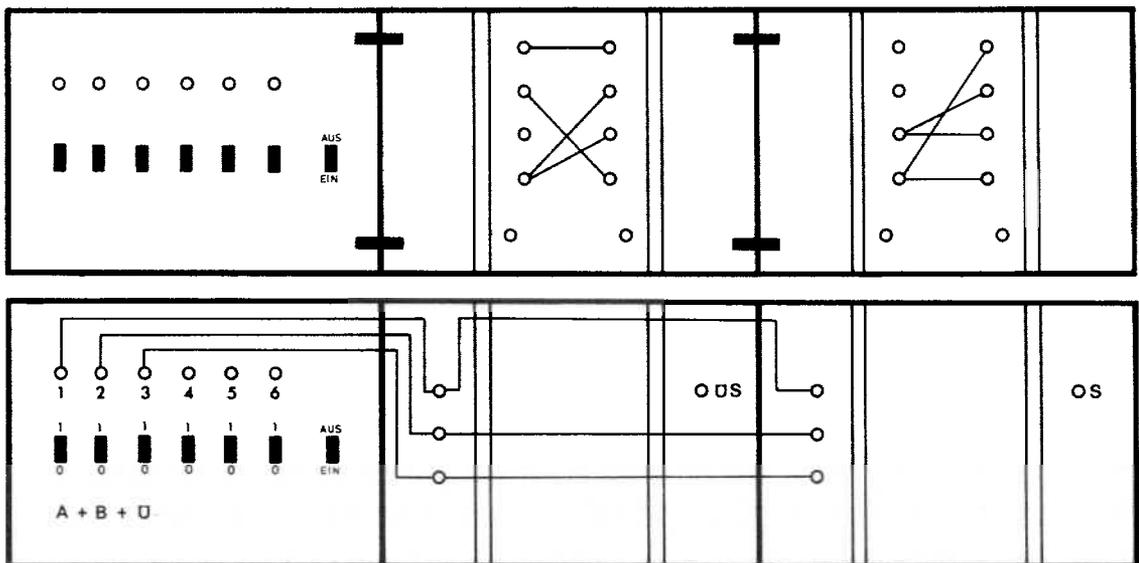
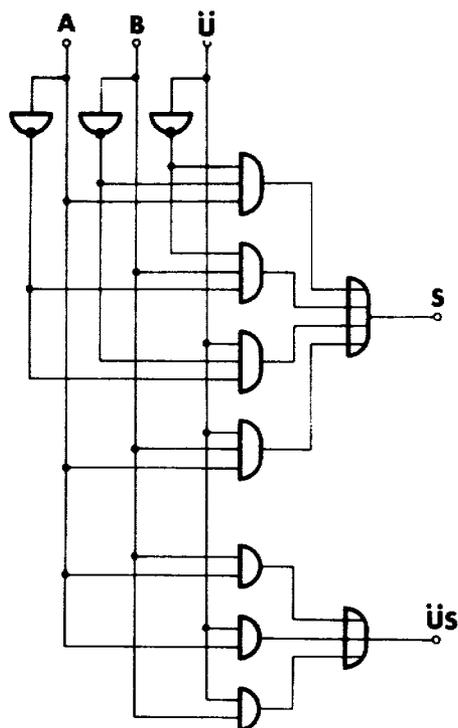


Abb.78

Logikplan des Volladdierers



Ein Beispiel:

Es sollen die Dualzahlen 1011 und 111 addiert werden (11 + 7 dezimal). Wir beginnen mit der rechten Dualstelle, der Stelle mit der geringsten Wertigkeit. Dabei bedeutet A die jeweilige Stelle der ersten Dualzahl, B bedeutet die jeweilige Stelle der zweiten Dualzahl. ÜS bedeutet ein Übertrag, der bei der Summenbildung entsteht, Ü ein Übertrag, der am Eingang mit zu berücksichtigen ist, der also dem Übertrag der vorangegangenen Addition entspricht. S ist die Summe der beiden addierten Dualziffern.

Erster Schritt:	A = 1,	B = 1,	Ü = 0	-	Ergebnis:	S = 0,	ÜS = 1
zweiter "	: A = 1,	B = 1,	Ü = 1	-	"	: S = 1,	ÜS = 1
dritter "	: A = 0,	B = 1,	Ü = 1	-	"	: S = 0,	ÜS = 1
vierter "	: A = 1,	B = 0,	Ü = 1	-	"	: S = 0,	ÜS = 1
fünfter "	: A = 0,	B = 0,	Ü = 1	-	"	: S = 1,	ÜS = 0

Gesamtergebnis: 10010
(18 dezimal)

Bei dem hier gezeigten Additions-Rechenwerk müssen Sie dem Computer etwas unter die Arme greifen. Die Ergebnisse und die jeweiligen Überträge jeder Ziffernstelle werden extern zwischengespeichert; d.h. sie müssen auf einem Zettel notiert werden. Die Summenziffern notieren Sie, um so schrittweise das Endergebnis zu erhalten. (Selbstverständlich von rechts nach links, entsprechend der Stellenwertigkeit der Ziffern.) Die Ausgangsüberträge ÜS notieren Sie, um sie bei der Addition der nächsthöheren Ziffernstelle mit zu berücksichtigen.

Einen Addierer, bei dem die Dualzahlen parallel eingegeben werden und der die Summe aus 2 zweistelligen Dualzahlen direkt anzeigt, erhalten Sie, wenn ein Halbaddierer und ein Volladdierer nach der folgenden Abbildung in Reihe geschaltet werden.

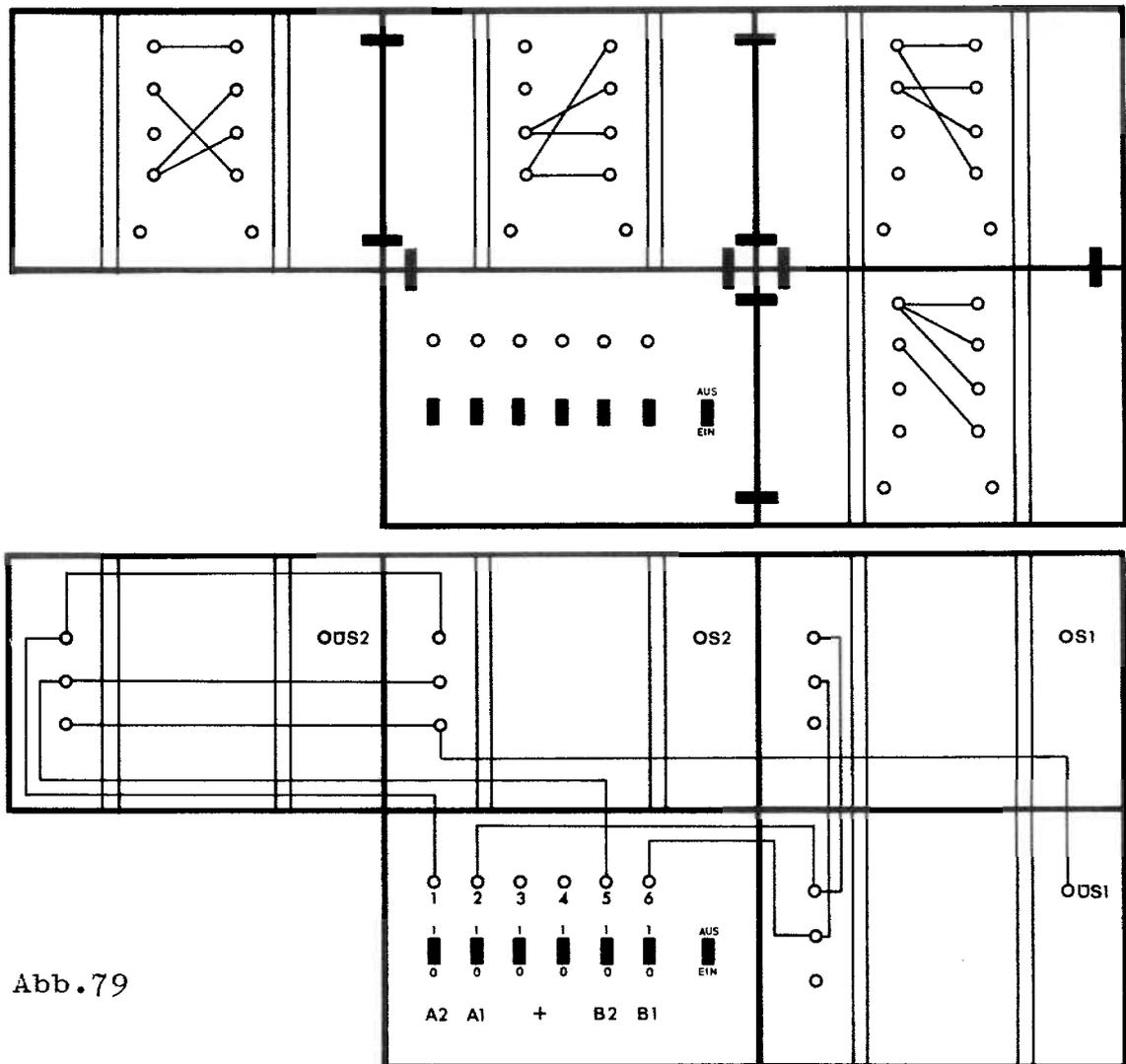


Abb.79

Wenn Sie das Ergebnis der Addition ablesen wollen, dürfen Sie den Übertrag $\bar{U}S1$ nicht mit berücksichtigen, da er bei der Summenbildung des Volladdierers ($S2$, $\bar{U}S2$) mit in die Anzeige eingeht.

Ein kleines Rechenbeispiel:

Mit $A1$ und $A2$ geben Sie die erste Dualzahl - z.B. 10 - ein und mit $B1$ und $B2$ die zweite - angenommen sie sei 11. Im Dezimalsystem entspricht das der Rechnung $2 + 3 = 5$. Die Zahl 5 im Dezimalsystem entspricht 101 im Dualsystem. Sie stellen fest, daß die Anzeigelampe $S1 = 1$, $S2 = 0$ und $\bar{U}S1 = 1$ sind und können damit das Ergebnis direkt ablesen. Die Umrechnung vom Dual- in das Dezimalsystem wird erleichtert, wenn Sie den Ausgängen $S1$ den Wert 1 (dezimal), $S2$ den Wert 2 und $\bar{U}S3$ den Wert 4 zuordnen. Da in diesem Beispiel $S1 = 1$ (dual) und $\bar{U}S3 = 1$ (dual) sind, ist das Ergebnis $1 + 4 = 5$. Dieser Addierer in Parallelbetrieb kann durch Reihenschaltung von weiteren Volladdierern in seiner Kapazität vergrößert werden.

12.5. Subtraktion

Kurz zusammengefaßt hier noch einmal die Rechenregeln für das Subtrahieren von Dualzahlen:

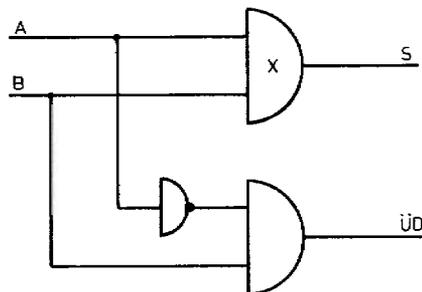
1. Dualzahl	-	2. Dualzahl	=	Differenz	Übertrag
0	-	0	=	0	0
0	-	1	=	1	1
1	-	0	=	1	0
1	-	1	=	0	0

Daraus lassen sich die Funktionstabellen für die Differenz D und für den Übertrag ÜD ableiten, wobei in den Tabellen die erste Dualzahl mit A und die zweite mit B bezeichnet werden.

A	B	D
0	0	0
0	1	1
1	0	1
1	1	0

A	B	ÜD
0	0	0
0	1	1
1	0	0
1	1	0

Die Differenz kann wie bei der Addition mit der Exklusiv-ODER-Funktion gebildet werden, der Übertrag ÜD dagegen aus einem UND mit negiertem A-Eingang. So sieht der Logikplan des Halbsubtrahierers aus:



Soll dieser Logikplan mit zwei Logik-Bausteinen verwirklicht werden, ergibt sich folgende Programmierung:

Programmierung Baustein D

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

→ f1 = 0

→ f2 = 1

→ f3 = 1

→ f4 = 0

Programmierung Baustein ÜD

A	B	F
0	0	0
0	1	1
1	0	0
1	1	0

→ f1 = 0

→ f2 = 1

→ f3 = 0

→ f4 = 0

Verdrahtungsplan des Halbsubtrahierers

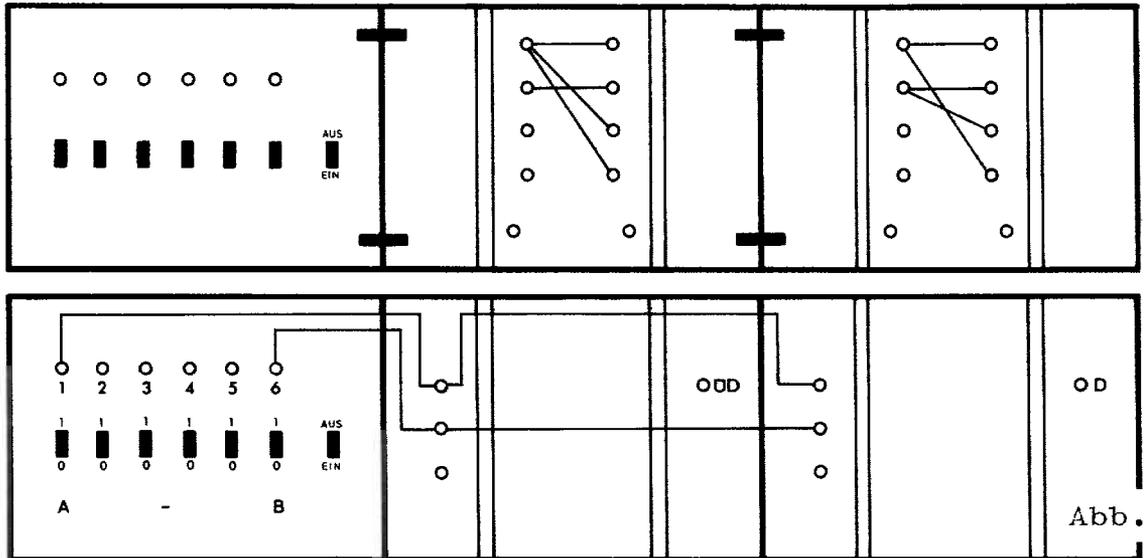


Abb. 80

Kontrollieren Sie die Schaltung mit Hilfe der Rechenregeln für die Subtraktion. Sollen neben der eigentlichen Subtraktion A minus B noch der Übertrag \bar{U} aus dem vorangegangenen Rechenvorgang berücksichtigt werden, muß die Programmierung für einen Vollsubtrahierer gefunden werden.

Funktionstabelle und Programmierung für die Differenz D

1. Dualziffer		2. Dualziffer		Übertrag	Differenz
A	B	A	B	\bar{U}	D
A	B	A	B	C	F
0	0	0	0	0	0
0	0	0	1	1	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	1	0
1	1	1	0	0	0
1	1	1	1	1	1

- $f1 = C$
- $f2 = \bar{C}$
- $f3 = \bar{C}$
- $f4 = C$

Funktionstabelle und Programmierung für den Übertrag $\bar{U}D$

1. Dualziffer		2. Dualziffer		Übertrag	Übertrag
A	B	A	B	\bar{U}	$\bar{U}D$
A	B	A	B	C	F
0	0	0	0	0	0
0	0	0	1	1	1
0	1	1	0	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	0
1	1	1	0	0	0
1	1	1	1	1	1

- $f1 = C$
- $f2 = 1$
- $f3 = 0$
- $f4 = C$

Die Schaltung des Vollsubtrahierers aus 2 Logik-Bausteinen:

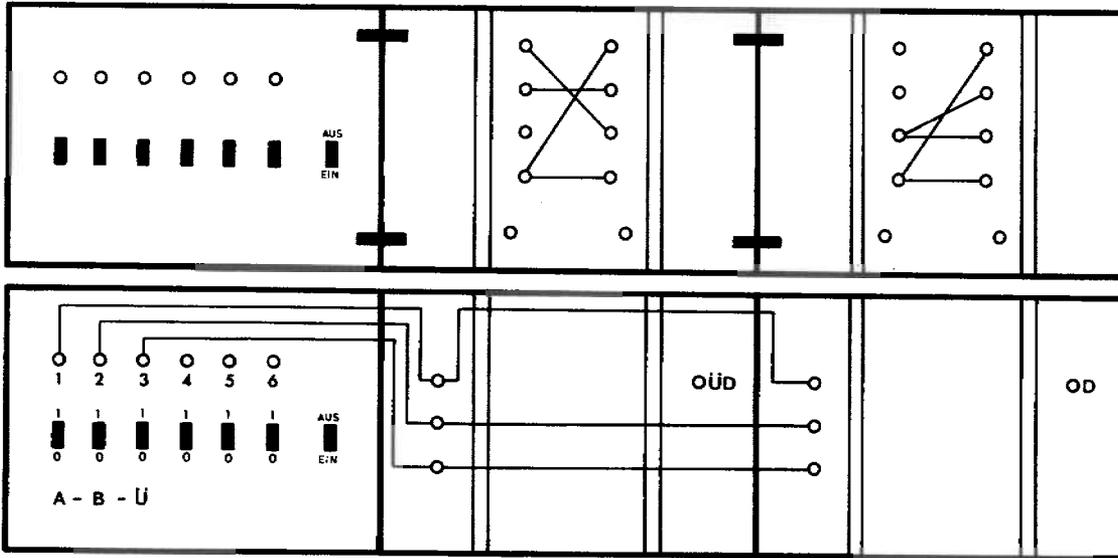


Abb.81

Beispiel: Es soll die Dualzahl 0111 (7) von der Dualzahl 1011 (11) abgezogen werden.

Erster Schritt:	A = 1,	B = 1,	Ü = 0	-	Ergebnis:	D = 0,	ÜD = 0
zweiter "	A = 1,	B = 1,	Ü = 0	-	"	D = 0,	ÜD = 0
dritter "	A = 0,	B = 1,	Ü = 0	-	"	D = 1,	ÜD = 1
vierter "	A = 1,	B = 0,	Ü = 1	-	"	D = 0,	ÜD = 0

Endergebnis: 0100 (4)

D bedeutet hier Differenz, ÜD ist der Übertrag, der bei der Differenzbildung entsteht; alle übrigen Bezeichnungen entsprechen denen der Addition.

Sie können sich denken, daß bei großen Computern die Ergebnisse der einzelnen Ziffernstellen sowie die Überträge im Computer selbst gespeichert werden. Daß Sie bei den Experimenten "externe Speicherung" mit hinzuziehen, tut dem Prinzip jedoch keinen Abbruch.

Die direkte Subtraktion von 2 zweistelligen Dualzahlen kann mit der Reihenschaltung aus einem Halb- und einem Vollsubtrahierer verwirklicht werden. Wie bei jeder Subtraktion von Dualzahlen muß der Minuend A größer als der Subtrahend B sein, da sonst das Ergebnis negativ wird und nicht ohne weiteres wieder in das Dezimalsystem umgewandelt werden kann. Der Parallelsubtrahierer in der folgenden Schaltung zeigt einen negativen Wert durch das Aufleuchten der Lampe ÜD2 an.

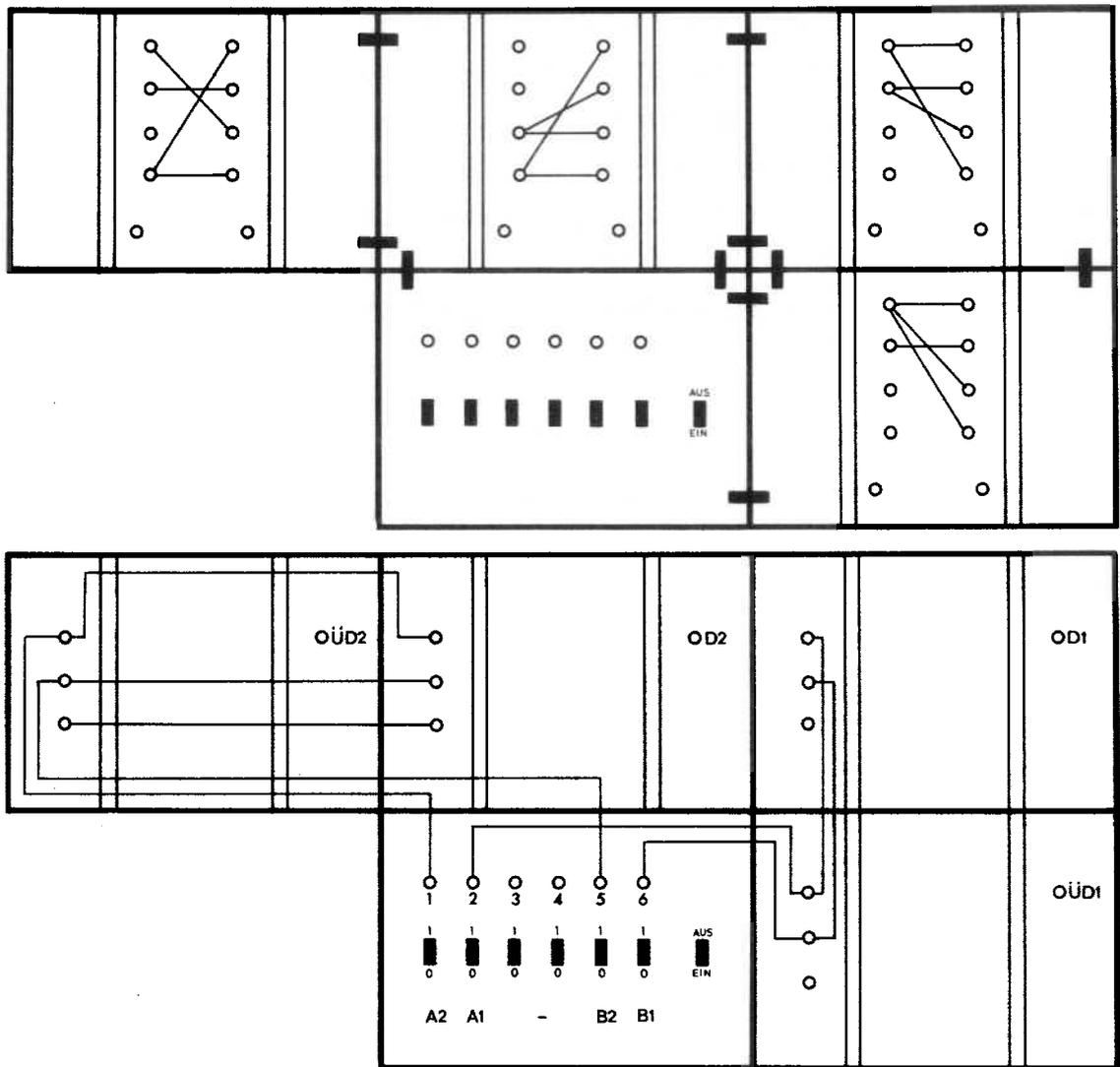


Abb.82

Beispiel:	<u>dual</u>	<u>dezimal</u>	<u>üD2</u>	<u>D2</u>	<u>D1</u>
11 - 10 =	001	3 - 2 =	- 0 + 0 + 1 =	+ 1	
01 - 01 =	000	1 - 1 =	- 0 + 0 + 0 =	0	
10 - 01 =	001	2 - 1 =	- 0 + 0 + 1 =	+ 1	
11 - 01 =	010	3 - 1 =	- 0 + 2 + 0 =	+ 2	
11 - 00 =	011	3 - 0 =	- 0 + 2 + 1 =	+ 3	

Sie können das Ergebnis leichter in das Dezimalsystem übertragen, wenn Sie dem Ausgang D1 die dezimale Ziffer +1 und D2 +2 zuordnen. Zeigt der Ausgang üD2 die duale Ziffer 1 an, ist das Ergebnis negativ, dabei ist die Wertigkeit dieser Anzeige (dezimal) -4.

Hier einige Beispiele, bei denen ein negativer Differenzbetrag auftritt:

	<u>dual</u>	<u>dezimal</u>	<u>üD2</u>	<u>D2</u>	<u>D1</u>
00 - 11 =	101	0 - 3 =	- 4 + 0 + 1 =	- 3	
01 - 11 =	110	1 - 3 =	- 4 + 2 + 0 =	- 2	
01 - 10 =	111	1 - 2 =	- 4 + 2 + 1 =	- 1	

Wie auch beim Addierer im Parallelbetrieb läßt sich diese Subtrahierschaltung durch weitere Vollsubtrahierer vergrößern. Da für jede Dualstelle 2 Logik-Bausteine gebraucht werden, können Sie sich vorstellen, daß für den Rechenvorgang größerer Werte der Aufwand an Rechenwerken nicht mehr zu vertreten ist. Aus diesem Grund finden Sie in großen Computern fast nur sequentielle Addier- und Subtrahierschaltungen, die den Rechenvorgang schrittweise ausführen und die Teilergebnisse zwischenspeichern.

12.6. Sequentielle Addier- und Subtrahierschaltung

Um auch mit geringem Schaltungsaufwand große Zahlen addieren und subtrahieren zu können, bedient man sich sogenannter sequentieller Rechenwerke. Die Dualzahlen werden hier nicht wie bei dem Paralleladdierer und -subtrahierer in ihrem gesamten Wert in das Rechenwerk gegeben, sondern die einzelnen Dualziffern der Zahlen A und B werden zeitlich nacheinander addiert bzw. subtrahiert, wobei das jeweilige Ergebnis von einem Speicher übernommen wird.

Der nachfolgende Verdrahtungsplan zeigt eine Addierschaltung, bei der die Überträge ÜS in einen Zwischenspeicher gelangen und durch die Rückführung auf den Volladdierer im nachfolgenden Rechenvorgang automatisch berücksichtigt werden.

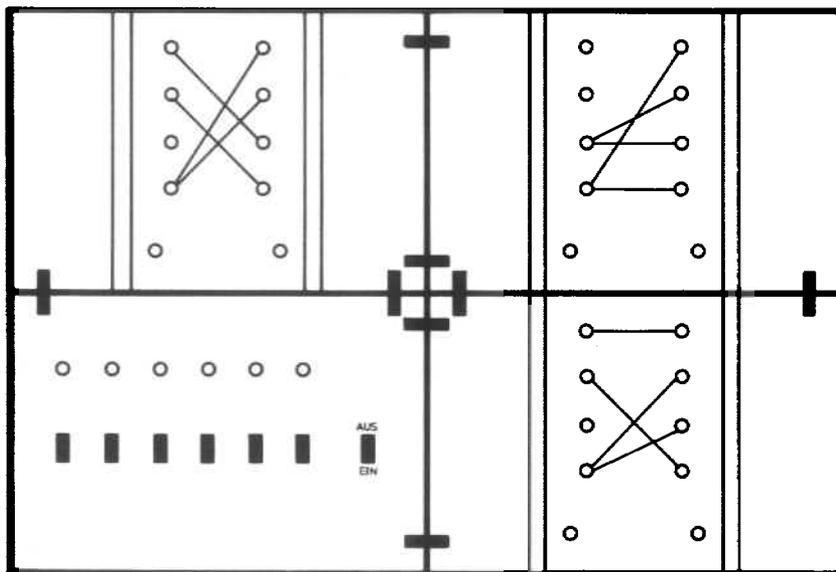


Abb.83

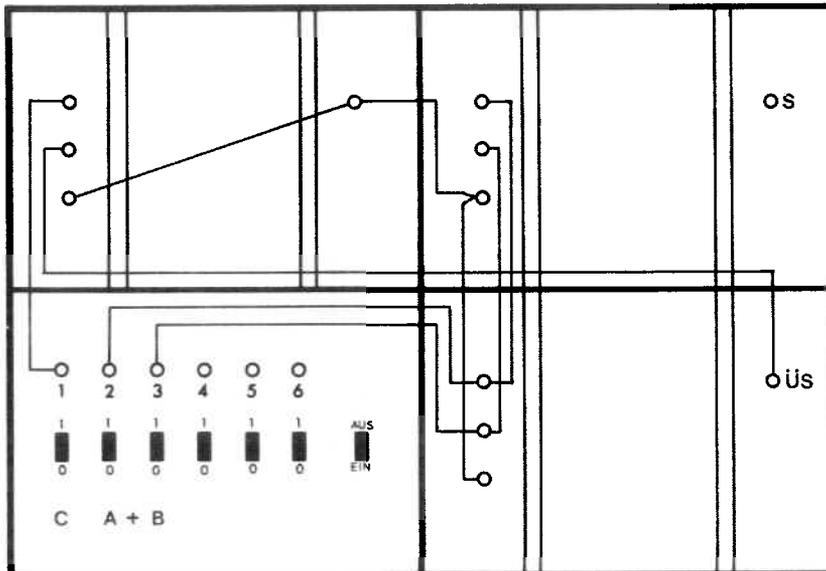


Abb. 83

Die jeweilige Speicherung des Übertrages ÜS ermöglicht das D-Auffang-Flipflop 1 (DFF1). In der Versuchsanordnung ist der linke Logik-Baustein als Flipflop programmiert, der rechte Schaltungsteil stellt den bekannten Volladdierer mit seinen Ausgängen S und ÜS dar.

Die Arbeitsweise ist folgende: Über die Schalter A und B geben Sie die ersten beiden Dualziffern (beachten Sie, daß hiermit die letzte Stelle der zu addierenden Dualzahlen gemeint ist) auf den Volladdierer, und es erscheint als erstes Teilergebnis die Summe S und der Übertrag ÜS. Als nächstes müssen Sie den Zwischenspeicher öffnen, indem Sie den Taktschalter C von 0 auf 1 setzen. Der Übertrag ÜS kann jetzt vom DFF1 übernommen werden. Wenn Sie nun den Schalter C wieder von 1 auf 0 zurückschalten, hat das D-Auffang-Flipflop den Übertrag gespeichert. Da der Ausgang des Speichers direkt mit dem Übertragseingang des Volladdierers verbunden ist, wird bei der nächsten Zifferneingabe der Übertrag automatisch berücksichtigt. Die jeweilige Summe S müssen Sie allerdings auf einem Zettel notieren.

Rechnen Sie folgendes Additionsbeispiel durch: $1011 + 0111$
(dezimal 11 + 7)

Hier die einzelnen Rechenschritte:

Schalter A		Schalter B	=	Summe S	ÜS	Schalter C
1	+	1	=	0	1	0 - 1 - 0
1	+	1	=	1	1	1 - 0
0	+	1	=	0	1	1 - 0
1	+	0	=	0	1	

Das Ergebnis ist also 10010 (dezimal 18).

Die sequentielle Subtrahierschaltung arbeitet nach dem gleichen Prinzip.
 Auch hier ein Rechenbeispiel: 1011 - 0111 (dezimal 11 - 7)

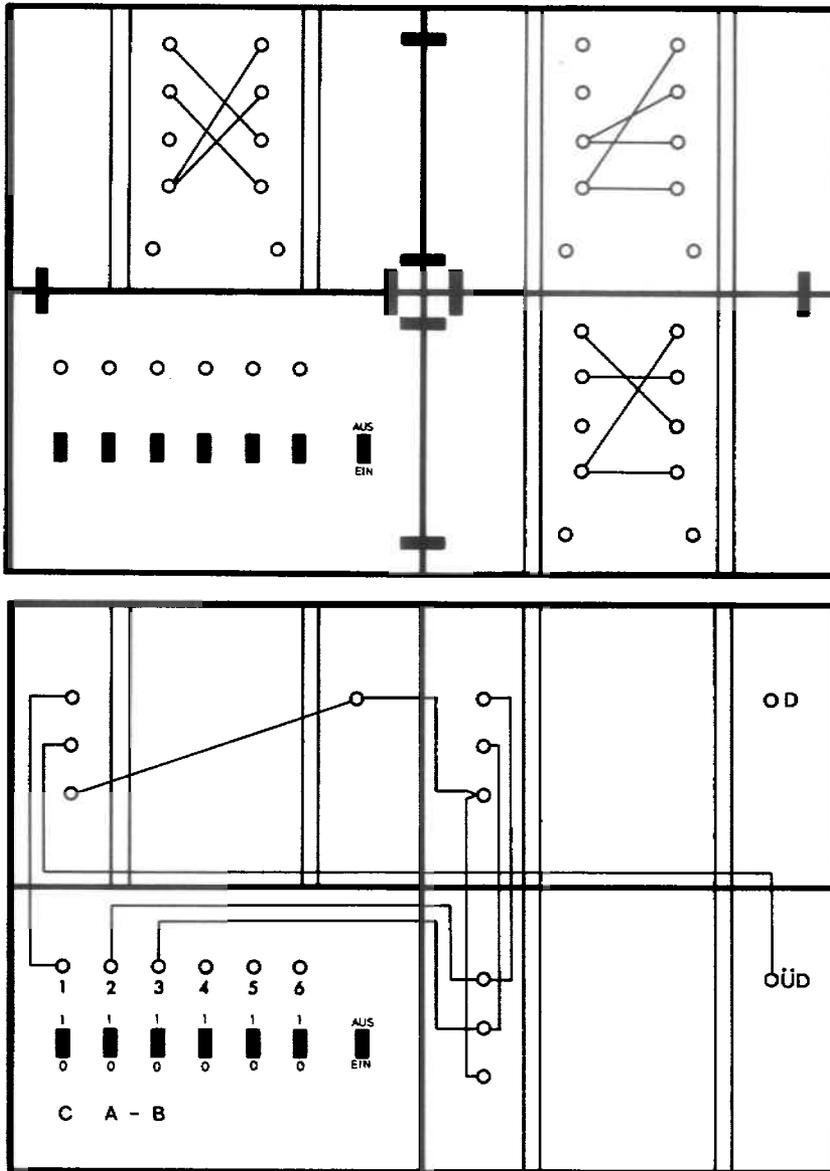


Abb. 84

Schalter A	Schalter B	Differenz D	ÜD	Schalter C
1	-	1	=	0
1	-	1	=	0
0	-	1	=	1
1	-	0	=	0

Ergebnis: 00100 (dezimal 4)

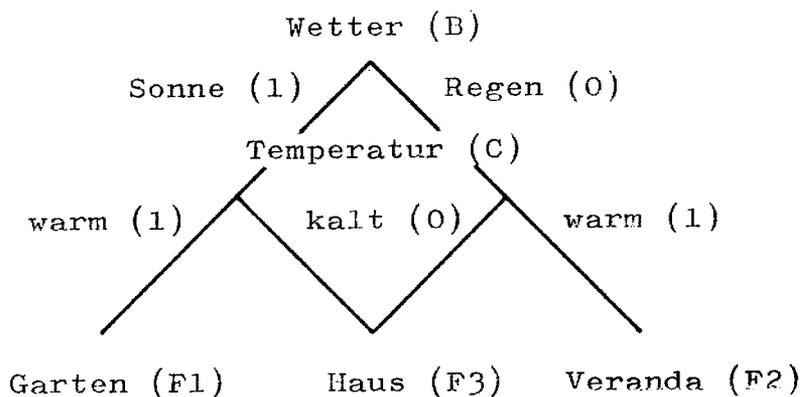
13. Anwendungsmöglichkeiten des CL 1601

13.1. Ein Party-Problem

Wir wollen eine Gartenparty feiern. Das ungewisse Wetter stellt uns vor folgendes Entscheidungsproblem:

1. Wenn die Sonne scheint und wenn es warm ist, dann feiern wir im Garten.
2. Wenn es regnet und wenn es warm ist, dann feiern wir in der Veranda.
3. Wenn es kalt ist, dann feiern wir in jedem Fall im Haus.

Wir können uns dieses Entscheidungsproblem übersichtlich als "Entscheidungsbaum" aufzeichnen.



Programmieren Sie Ihren Computer folgendermaßen:

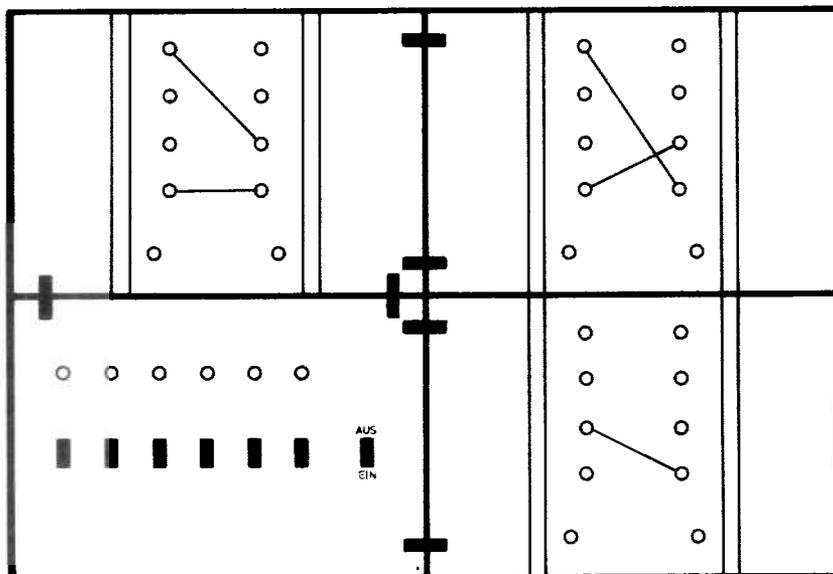


Abb.85

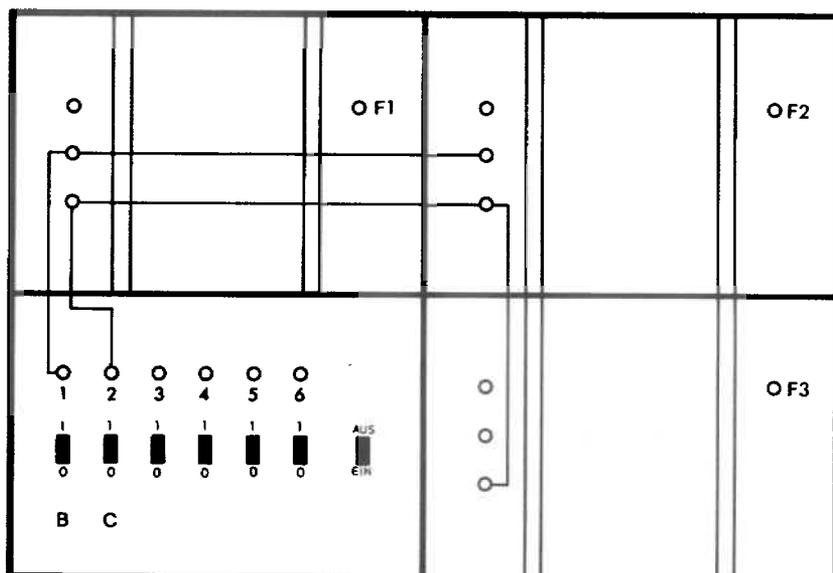


Abb.85

Und nun schalten Sie alle Möglichkeiten durch und ermitteln Sie, in welchen Fällen Sie Ihre Party im Garten, in der Veranda oder im Haus feiern werden.

Wetter B	Temperatur C	Garten F1	Veranda F2	Haus F3
0	0			
0	1			
1	0			
1	1			

Das ist also das Ergebnis:

B	C	F1	F2	F3
0	0	0	0	1
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

Wir können diese Funktionstabelle auch als Entscheidungstabelle bezeichnen, denn sie gibt uns an, wie wir uns je nach den Bedingungen bei unserem Problem entscheiden müssen. Bei dieser Gelegenheit sei auch noch ein weiterer Begriff für Funktionstabelle erwähnt: Wahrheitstabelle.

13.2. Wettervorhersage

Das Wetter läßt sich mit einiger Wahrscheinlichkeit richtig vorhersagen, wenn mehrere Bedingungen bekannt sind. Wir wissen:

Im Sommer wird bei steigendem Luftdruck und bei steigender Temperatur das Wetter schön; bei fallendem Luftdruck und fallenden Temperaturen wird das Wetter schlecht. Im Winter dagegen wird bei steigendem Luftdruck und fallenden Temperaturen das Wetter schön; bei fallendem Luftdruck und steigenden Temperaturen wird das Wetter schlecht.

Nun werden Sie mit Ihrem Computer mit Sicherheit nicht die Wettervorhersage der professionellen "Wetterpropheten" in den Schatten stellen können. Zudem wollen wir der Einfachheit halber vereinbaren, daß eine Wettervorhersage nicht möglich ist, wenn

im Sommer der Luftdruck steigt und die Temperatur fällt - oder umgekehrt - und wenn im Winter der Luftdruck steigt und die Temperatur steigt - oder umgekehrt.

Unsere Vorhersagetabelle sieht demnach so aus:

Jahreszeit A	Luftdruck B	Temperatur C	Wetter wird
Sommer 1	steigt 1	steigt 1	schön F1 = 1
	fällt 0	fällt 0	schlecht F2 = 1
Winter 0	steigt 1	fällt 0	schön F1 = 1
	fällt 0	steigt 1	schlecht F2 = 1

Programmieren Sie Ihren Computer wie in der Abbildung dargestellt.

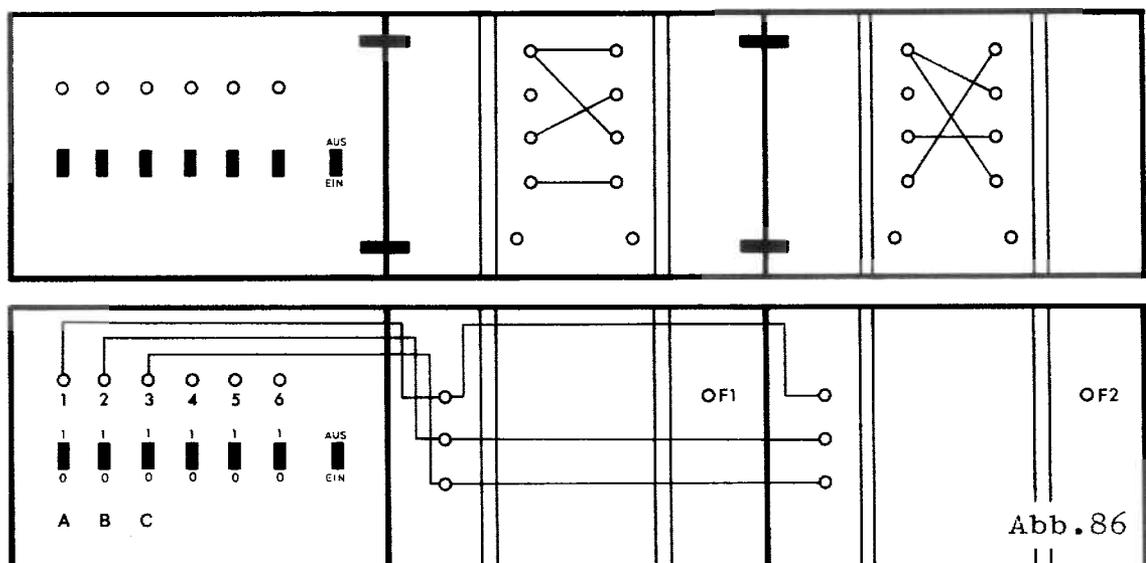


Abb. 86

Und jetzt schalten Sie nach folgender Vereinbarung die verschiedenen Bedingungen durch:

A = 0 : Winter
 A = 1 : Sommer
 B = 0 : Luftdruck fällt
 B = 1 : Luftdruck steigt
 C = 0 : Temperatur fällt
 C = 1 : Temperatur steigt

Tragen Sie die verschiedenen Wettervorhersagen in die Tabelle ein:

Jahreszeit	Luftdruck	Temperatur	Wetter schön	Wetter schlecht
A	B	C	F1	F2
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Die Lösungen:

Jahreszeit	Luftdruck	Temperatur	Wetter schön	Wetter schlecht
A	B	C	F1	F2
+ 0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
+ 0	1	1	0	0
1	0	0	0	1
+ 1	0	1	0	0
+ 1	1	0	0	0
1	1	1	1	0

Anmerkung: + keine Vorhersage möglich

13.3. Kreditprüfung

Ein Problem, das in allen Banken und Sparkassen täglich auftritt und eine Entscheidung verlangt:

Ein Kunde möchte von seinem Konto Geld abheben, auf dem Konto aber befindet sich kein Guthaben mehr. Wegen seiner Häufigkeit wird dieses Problem gern als Musterbeispiel zur Erläuterung von Entscheidungstabellen (entspricht unseren Funktionstabellen) herangezogen. Die verschiedenen Fragestellungen sind diese:

1. Ist das Kreditlimit noch nicht überschritten, d.h. ist ein Kredit ohne weiteres möglich (Kredit möglich)?

2. Wenn nein: Ist die Zahlungsweise des Kunden aus Erfahrung gut, so daß auch bei Überschreiten der Kreditgrenze kein Risiko besteht (Zahlungsweise gut)?
3. Wenn nein: Erteilt der Leiter der Kreditabteilung in diesem außergewöhnlichen Fall eine Sondergenehmigung (Sondergenehmigung)?

Müssen alle drei Fragen mit NEIN beantwortet werden, ist der Kreditantrag abzulehnen. Wenn jedoch bereits eine Frage mit JA beantwortet wird, so kann der Kredit überwiesen bzw. ausgezahlt werden.

Programmieren Sie nun Ihren Computer.

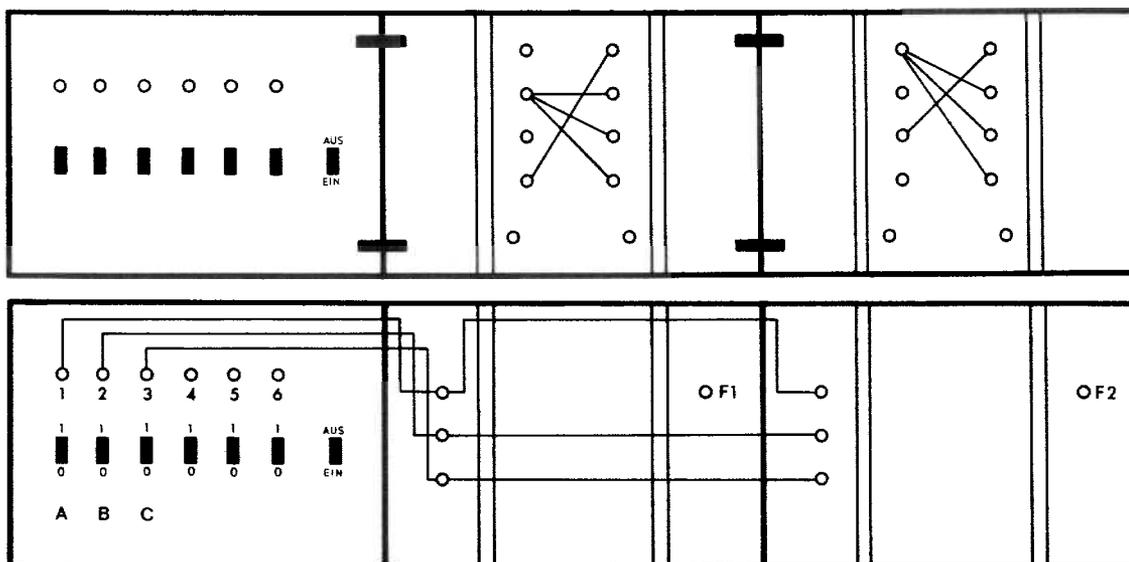


Abb.87

Wenn Sie nun die verschiedenen Möglichkeiten "durchschalten", erkennen Sie leicht, daß das Kreditproblem folgender Entscheidungstabelle entspricht:

A	Kredit möglich?	J	N	N	N
B	Zahlungsweise gut?		J	N	N
C	Sondergenehmigung erteilt?			J	N
=====					
F1	Kredit freigegeben	X	X	X	
F2	Kredit abgelehnt				X

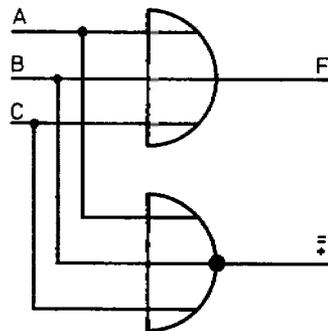
JA (J) = 1

NEIN (N) = 0

Entscheidung (X) = Lampe brennt (F = 1)

Sie erkennen aus der Tabelle, daß die Kreditgewährung eine ODER-Bedingung darstellt, denn nur eine der drei Bedingungen A, B und C muß erfüllt sein, damit der Kredit gewährt wird. Da bereits eine Bedingung ausreicht, um einen Kredit zu bekommen, haben wir die Fälle, bei denen mehrere Bedingungen erfüllt sind (einschließendes ODER), der Einfachheit halber hier weggelassen. Die drei Bedingungen zusammen mit dem Ereignis "Auftrag abgelehnt" stellen eine NOR-Funktion dar: Wenn ein Kredit nicht möglich ist und wenn die Zahlungsweise nicht gut ist und wenn die Sondergenehmigung nicht erteilt wird, dann wird der Auftrag abgelehnt.

Den Zusammenhang können wir durch folgendes Schaltbild darstellen:

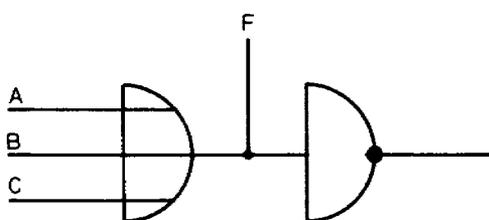


Versuchen Sie nun bitte, die beiden Funktionstabellen (ODER, NOR) gemäß der Entscheidungstabelle aufzustellen.

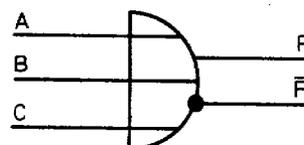
A	B	C	F1

A	B	C	F2

Wenn Sie sich jetzt die Ergebnisse noch einmal ansehen und feststellen, daß die beiden Rubriken "Auftrag angenommen" bzw. "Auftrag abgelehnt" immer einander entgegengesetzte Werte haben, so läßt sich leicht einsehen, daß auch folgende Darstellungsweise für die Schaltsymbolik möglich ist.



oder vereinfacht



Wir haben es hier mit einer logischen Funktion mit drei Eingängen und zwei Ausgängen zu tun. Die Ausgänge sind einander entgegengesetzt, man spricht deshalb auch von komplementären Ausgängen. Demnach können wir auch bei unserem Computer den zweiten Ausgang einfach als komplementären Ausgang programmieren. Wir verdrahten ihn als Inverter:

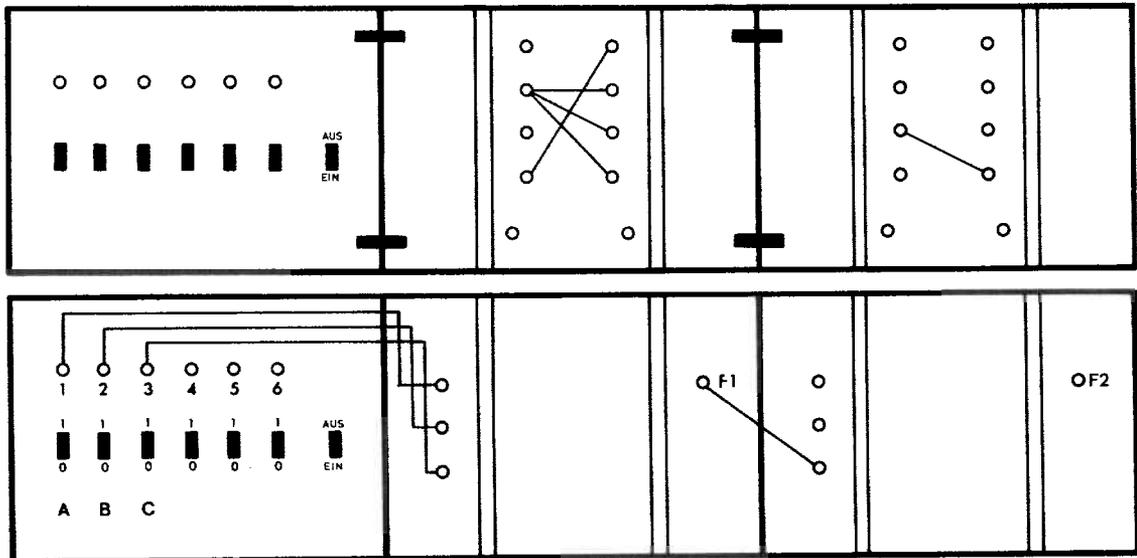


Abb.88

13.4. Platzreservierung

Computer werden heutzutage auch bei der Platzreservierung im Eisenbahnverkehr eingesetzt. Das wollen wir auch tun, allerdings verständlicherweise in etwas vereinfachter Form.

Nehmen wir an, Sie wünschen eine Platzreservierung für eine bestimmte Fahrt in der 2. Klasse. Das Personal, das für die Platzreservierung zuständig ist, hat folgende Anweisungen:

Wenn in der 2. Klasse noch ein Platz frei ist, dann für den Reisenden reservieren.

Ist in der 2. Klasse kein Platz frei, dann fragen, ob eine Reservierung in der 1. Klasse akzeptiert wird.

Wenn JA: Ist dort noch ein Platz frei?

Wenn JA: In der 1. Klasse reservieren.

Wenn in der 1. Klasse kein Platz mehr frei ist, fragen, ob für einen anderen Zug gebucht werden soll.

Wenn JA, wird eine Reservierung für diesen Ausweichzug vorgenommen.

Wird der Ersatzzug nicht gewünscht, so ist dem Reisenden zu empfehlen, ohne Platzkarte rechtzeitig beim Eintreffen des Zuges auf dem Bahnsteig zu sein.

Stellen wir die Zusammenhänge in einer Entscheidungstabelle dar, ergeben sich folgende Verhältnisse:

A	2. Klasse verfügbar?	J	N	N	N
B	1. Klasse akzeptiert?		J	N	N
C	Ersatzzug akzeptiert?			J	N
=====					
F1	Buchen 2. Klasse	X			
F2	Buchen 1. Klasse		X		
F3	Buchen Ersatzzug			X	
F4	Reisen ohne Platzkarte				X

JA (J) = 1
 NEIN (N) = 0
 Entscheidung (X) = Lampe brennt (F = 1)

Zu den gleichen Ergebnissen kommen Sie, wenn Sie 4 Logik-Bausteine wie folgt programmieren.

A	B	C	F1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = 0

→ f2 = 0

→ f3 = \bar{C}

→ f4 = 0

A	B	C	F2
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = 0

→ f2 = \bar{C}

→ f3 = 0

→ f4 = 0

A	B	C	F3
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = C

→ f2 = 0

→ f3 = 0

→ f4 = 0

A	B	C	F4
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

→ f1 = \bar{C}

→ f2 = 0

→ f3 = 0

→ f4 = 0

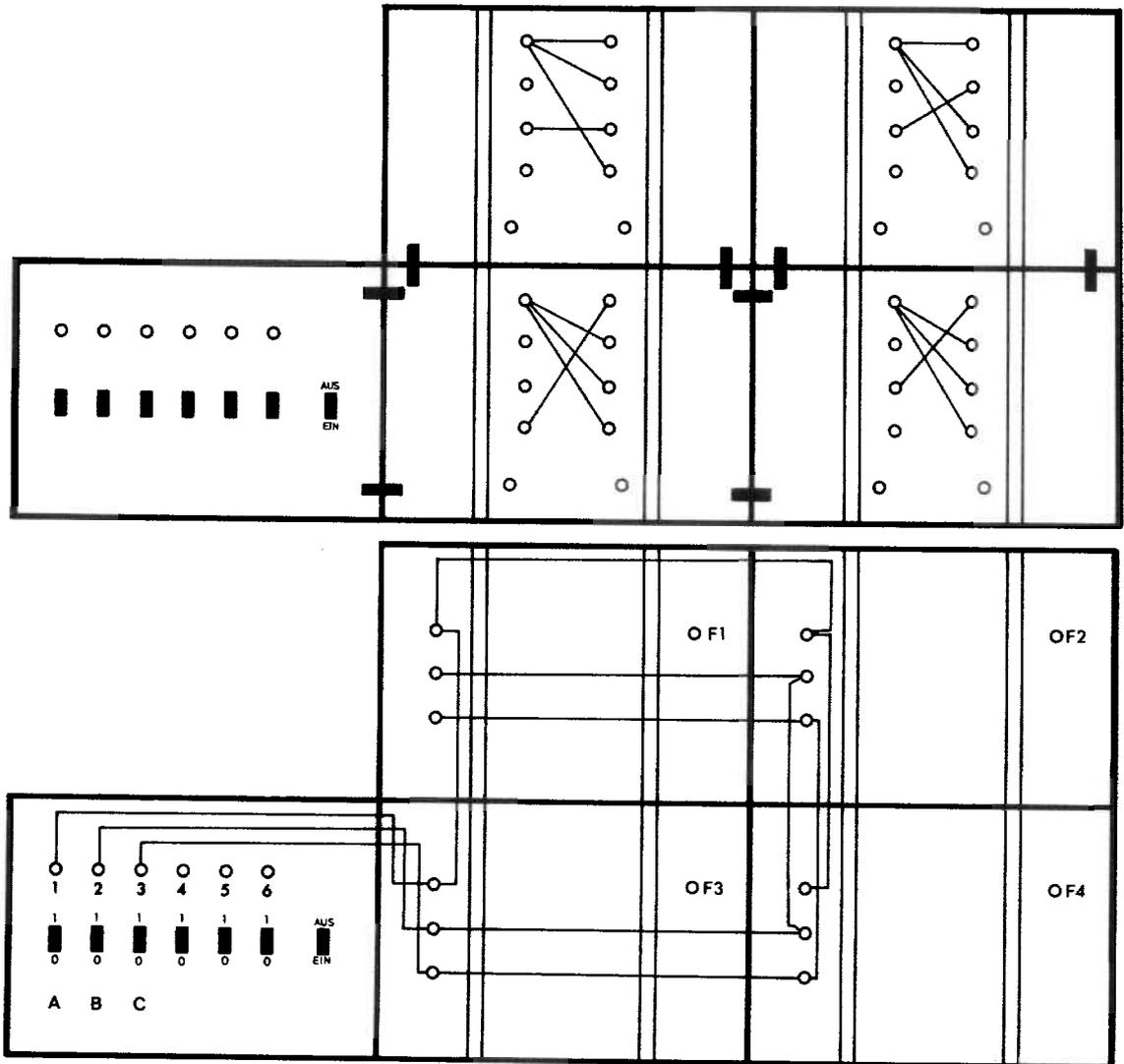


Abb.89

13.5. Ein Tresor wird geknackt

Ein sequentielles Kombinationsschloß aus Flipflops.

Programmieren Sie die Computer-Bausteine wie auf der Abbildung dargestellt als sogenannte D-Auffang-Flipflops 1. Setzen Sie nun alle Schalter auf 0 und bringen Sie die Schalter 1 - 5 der Eingabeeinheit einmal auf 1 und dann wieder auf 0. Damit sind alle Ausgänge der Versuchsanordnung auf 0 gesetzt. Wenn Sie jetzt den Schalter 6 auf 1 setzen und nacheinander die Schalter 1 - 5 kurzzeitig auf 1 und anschließend wieder auf 0 bringen, können Sie beobachten, daß die eingegebene Information - das Signal 1 also - von Logik-Baustein zu Logik-Baustein bzw. von Flipflop zu Flipflop weitergegeben wird, bis es beim 5. Flipflop angelangt ist und alle Lampen leuchten.

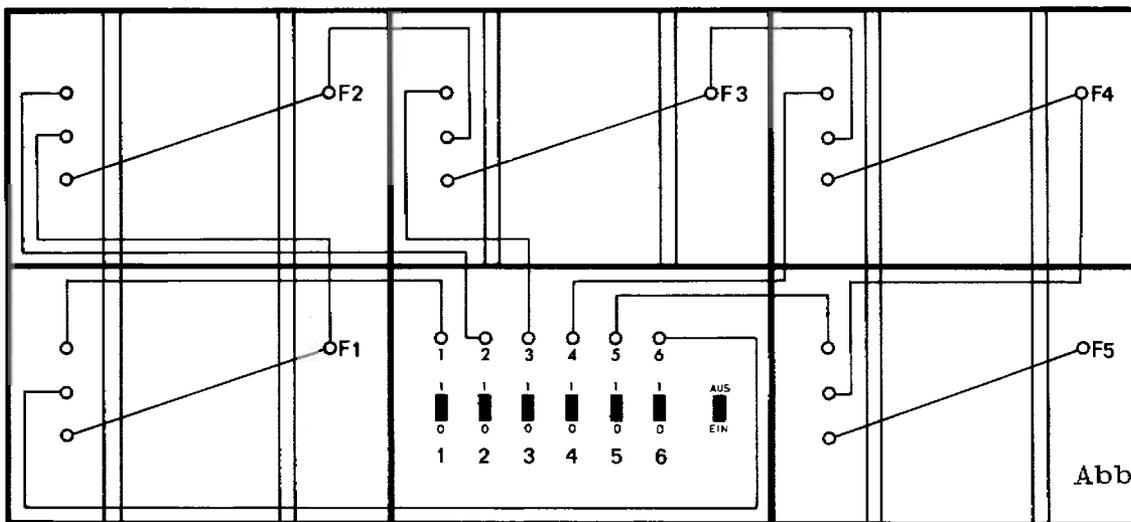
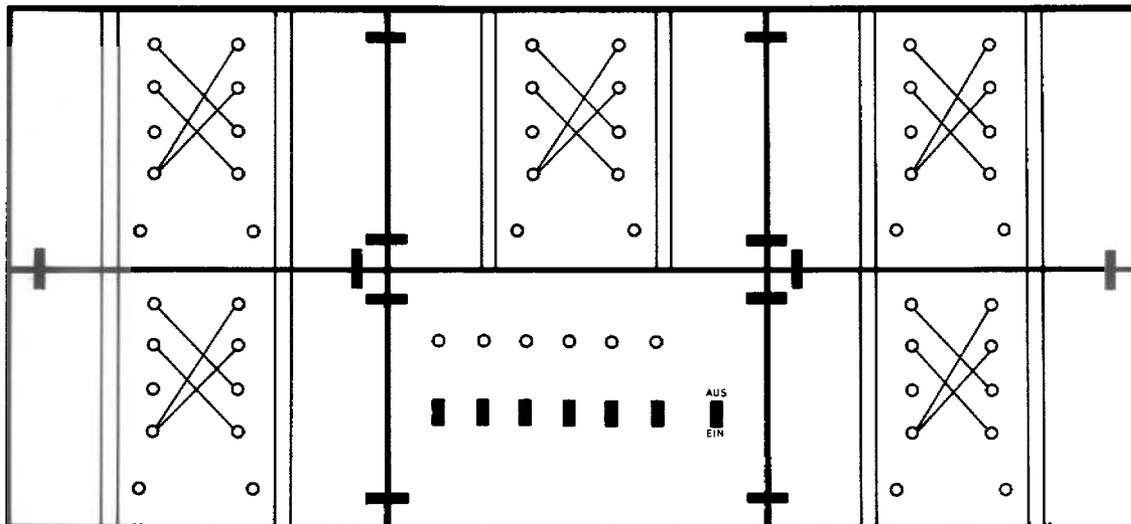


Abb. 90

Betätigen Sie die Schalter in anderer Reihenfolge, so wird das Signal nicht weitergegeben. Wir bauen unsere Versuchsanordnung jetzt zu einem Kombinationsschloß um, das nur derjenige betätigen kann, der die Kombination, d.h. die Reihenfolge, in der die Schalter zu bedienen sind, kennt.

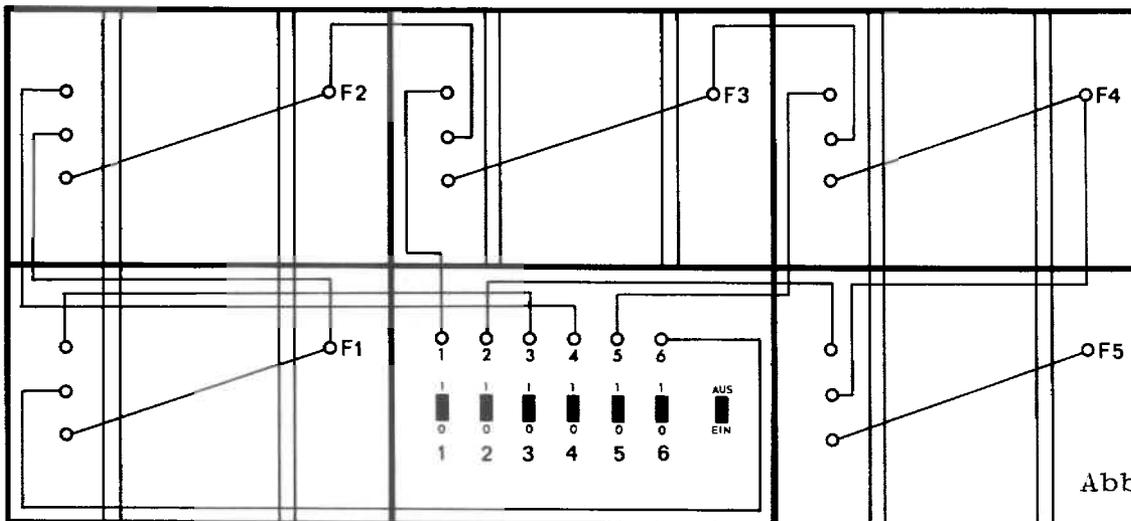


Abb. 91

Die Schaltungsanordnung soll das Schloß zu einem Tresor sein. Lassen Sie einen Freund versuchen, ob er diesen Tresor knacken kann. Das Kombinationsschloß gibt den Zugang zum Tresor frei, wenn alle Lampen der Logik-Bausteine brennen. Solange Ihr Freund die Verdrahtung nicht durchschaut, ist es reine Glückssache, zum Ergebnis zu gelangen, denn es gibt insgesamt 120 Möglichkeiten für die Reihenfolge, in der die Schalter durchgeschaltet werden können. Sie selbst kennen natürlich die richtige Kombination - im abgebildeten Beispiel 3 4 1 5 2 - und können somit den Tresor jederzeit öffnen.

Diese Kombination können Sie beliebig ändern, indem Sie die Eingänge A der Logik-Bausteine 1 - 5 immer anderen Schaltern zuordnen. Der Schalter, der mit Logik-Baustein 1 verbunden ist, muß stets zuerst betätigt werden, dann der mit Logik-Baustein 2 usw.

13.6. Knobeln

Haben Sie schon einmal geknobelt? Sicher kennen Sie dieses Spiel und wissen, daß man sich dabei verschiedener Hilfsmittel wie Streichhölzer, Würfel usw. bedienen kann. Oder Sie nehmen eine Münze. Ihr Mitspieler setzt auf "Adler" oder "Zahl", und Sie werfen das Geldstück in die Luft. Ist beispielsweise die Zahl sichtbar und Ihr Partner hat auch auf "Zahl" gesetzt, so ist er der Gewinner des Spiels.

Diese Art des Knobeln lässt sich auch elektronisch verwirklichen. Betrachten Sie dazu den nachfolgenden Verdrahtungsplan.

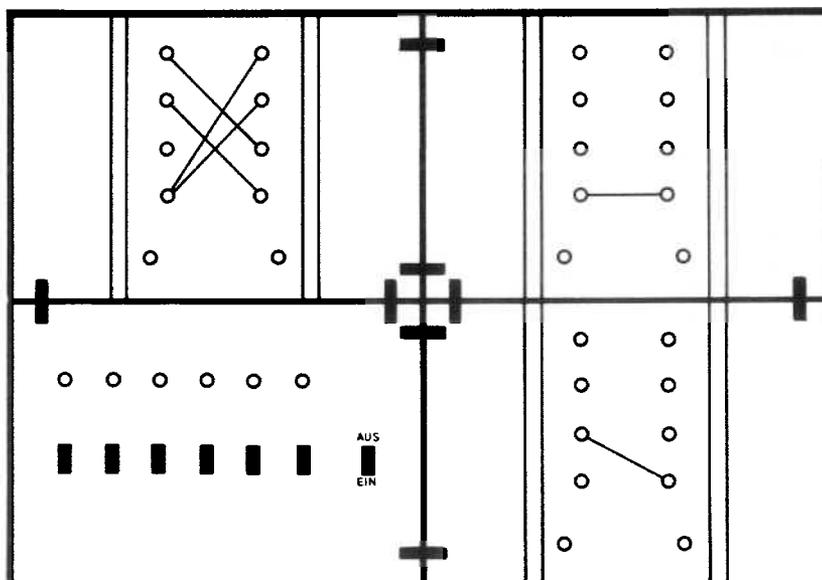


Abb. 92

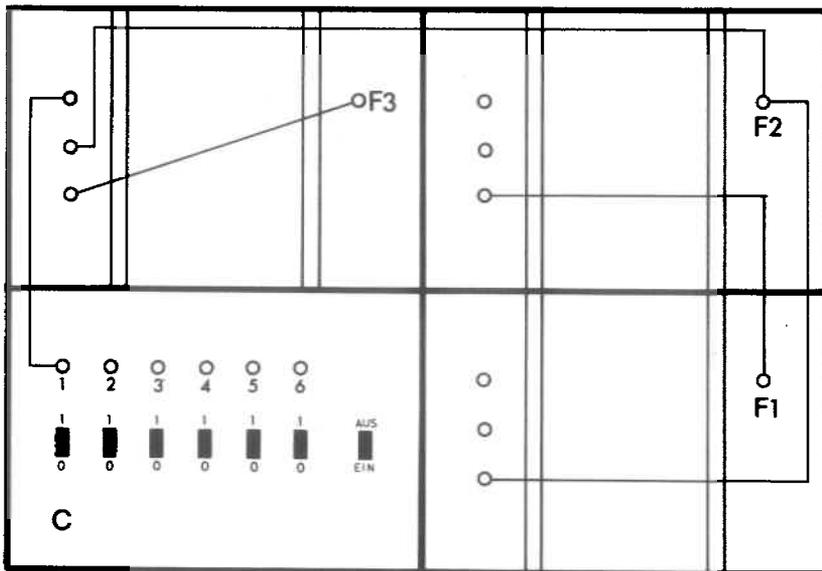


Abb.92

Die Logik-Bausteine 1 und 2 sind hier als Generator geschaltet, wobei der Ausgang F2 mit dem Eingang D eines D-Auffang-Flipflops 1 verbunden ist. Wird der Schalter C auf 1 gestellt, beeinflusst die Ausgangsspannung des Generators das Verhalten des DFF1. Wenn Sie jetzt den Schalter C wieder in die 0-Stellung bringen, speichert das Flipflop den Wert, der zuletzt am D-Eingang vorhanden war. Da die Generatorfrequenz sehr hoch ist und Sie den Schaltzustand nicht mit dem Auge verfolgen können, ist es ganz dem Zufall überlassen, ob die Lampe (F3) brennt oder ob sie dunkel bleibt.

13.7. Würfeln

Würfelspiele gibt es in zahlreichen Variationen. Doch sicherlich haben Sie noch nie elektronisch gewürfelt, wie Sie es in diesem Abschnitt tun werden.

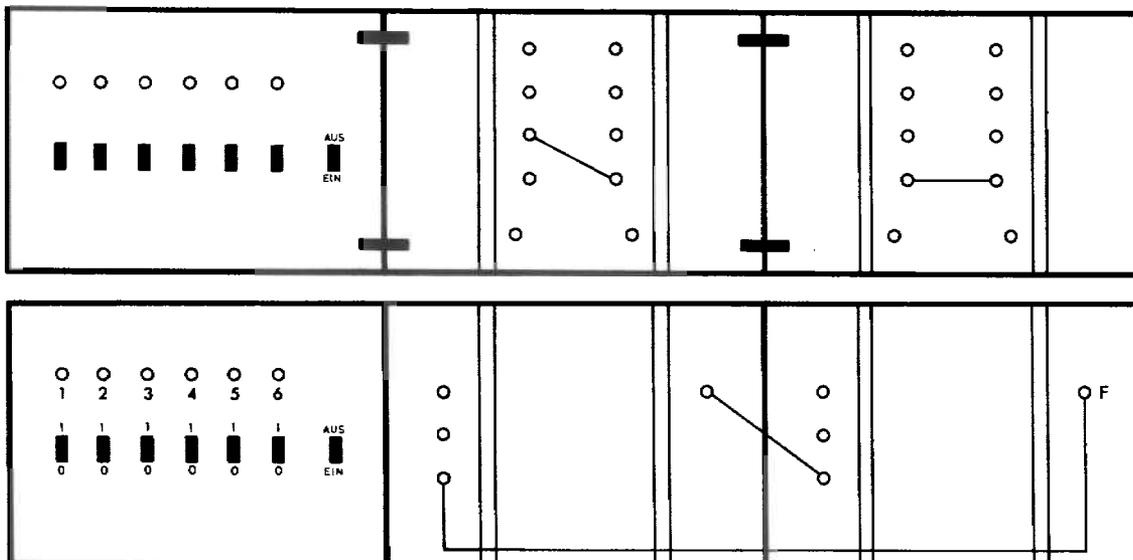


Abb.93

Grundlage dieses Würfelspiels ist ein Generator, der aus zwei Logik-Bausteinen (Inversions- und Identitätsschaltung) besteht.

Die Generatorfrequenz wird durch die Laufzeit der Logik-Bausteine bestimmt. Der Ausgang des Generators ist mit den Eingängen der drei Logik-Bausteine angeschlossen, die als D-Auffang-Flipflops geschaltet sind. Ein DFF1, das haben Sie inzwischen erfahren, speichert am Eingang B die Information, sobald der Eingang A den Wert 0 annimmt. Das Ergebnis des Würfels hängt also ausschließlich davon ab, ob Sie bei der Betätigung der Würfelschalter den Wert 0 am Ausgang des Generators oder den Wert 1 treffen. Da das bei der Generatorfrequenz nicht mehr mit dem Auge erfaßt werden kann, ist es wie beim Spiel mit richtigen Würfeln immer ein Zufall, welche Zahl Sie treffen.

Doch programmieren Sie zunächst einmal die gesamte "Würfelmaschine", wie in der folgenden Abbildung dargestellt.

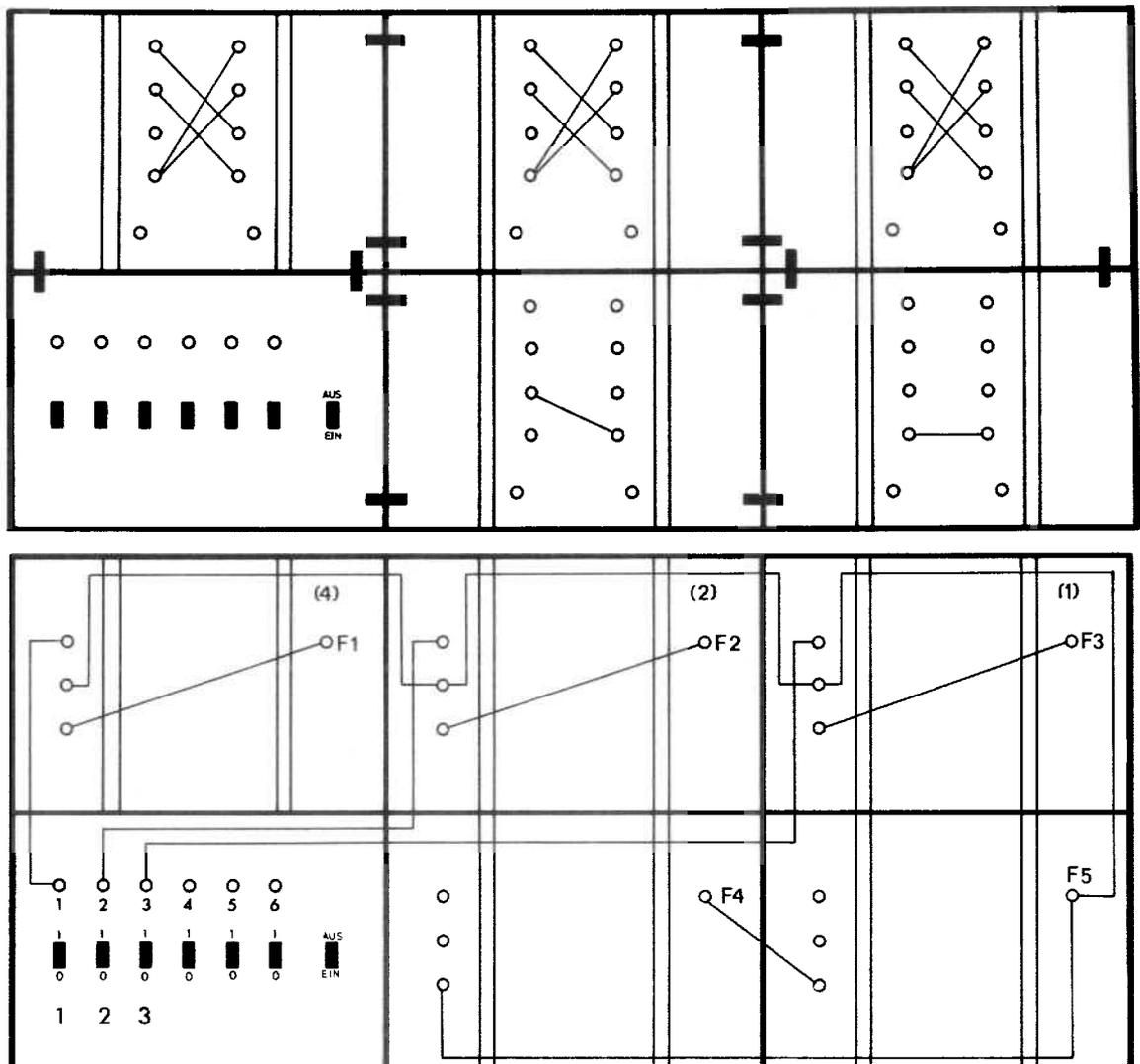


Abb. 94

Nach dem Betätigen des Hauptschalters blinkt der Ausgang des Generators, und die Ausgänge der drei DFF1 leuchten. Durch die Schalter 1 bis 3 können Sie nun die "Becher schütteln", d.h. das Würfeln beginnt. Nacheinander oder gleichzeitig müssen dann die Schalter auf STOP (0) gestellt werden, und das Ergebnis kann an den Ausgängen der D-Auffang-Flipflops abgelesen werden. F1 hat den Wert 4, F2 besitzt den Wert 2 und F3 hat den Wert 1. Wenn Sie die Werte der Ausgänge F1 bis F3 addieren, erhalten Sie das Ergebnis. Dabei stellt sich heraus, daß Ihr "Würfel" nicht, wie sonst üblich, von 1 bis 6 zählt, sondern von 0 bis 7. Wenn Sie sich schwer daran gewöhnen können, sollten Sie festlegen, daß z.B. die 0 einem ungültigen Wurf entspricht und bei 7 noch einmal wiederholt werden darf. Sie können natürlich auch bei 0 wiederholen lassen.

Ein Spielvorschlag für mehrere Personen: Jeder Teilnehmer erhält die gleiche Zahl von Zündhölzern und wer die höchste Zahl in einer Runde würfelt, darf ein Streichholz fortlegen. Wer zuerst alle Hölzer ablegen kann, hat gewonnen.

13.8. Bauer, Ziege, Wolf und Kohlkopf

In einer schon fast klassischen Denksportaufgabe will ein Bauer eine Ziege, einen Wolf und einen Kohlkopf über einen Fluß transportieren. Es steht ihm jedoch nur ein kleines Boot zur Verfügung, in dem er jeweils nur ein Objekt mitnehmen kann. Es ergeben sich daraus Probleme. Denn wenn der Bauer z.B. den Wolf hinübrudert, frißt die Ziege inzwischen den Kohlkopf. Wenn der Wolf und die Ziege zurückgelassen werden, frißt sie der Wolf auf. Der Bauer muß also so geschickt das Übersetzen ausführen, daß keine Gefahr auftreten kann. Eine Gefahr soll auch immer dann ausgeschlossen werden, wenn der Bauer zugegen ist.

Die Stellung 0 der Schalter 1 bis 4 (für Bauer, Ziege, Wolf und Kohlkopf) bedeutet, daß sich das Objekt am linken Ufer befindet; 1 heißt, daß es sich am rechten Ufer aufhält. Droht am linken Ufer oder am rechten Gefahr, so leuchtet dafür die Lampe des entsprechenden Logik-Bausteins auf. Während der Überfahrt besteht keine Gefahr, da der Bauer immer anwesend ist. Für die Anwesenheit des Bauern am linken Ufer sieht die Funktionstabelle folgendermaßen aus:

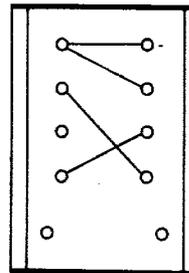
Vorbedingung: Bauer links (0) Teilbedingungen

Ziege A	Wolf B	Kohl C	Gefahr links F2	Gefahr rechts F1
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1	0 +)	0
1	1	1		0
1	0	1		1
1	1	0		1
1	1	1		1

+) Es droht keine Gefahr, solange der Bauer dort anwesend ist.

Aus dieser Funktionstabelle kann die Programmierung für den Logik-Bau-
stein entnommen werden, der auf die Gefahr am rechten Ufer hinweist,
wenn sich der Bauer am linken befindet.

Gefahr rechts F1	
0	
0	f1 = 0
0	
0	f2 = 0
0	
1	f3 = C
1	
1	f4 = 1
1	



Für die Anwesenheit des Bauern am rechten Ufer und die dabei entstehen-
den Gefahren gilt eine entsprechende Funktionstabelle.

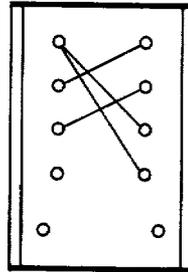
Vorbedingung: Bauer rechts (1) Teilbedingungen

Ziege A	Wolf B	Kohl C	Gefahr links F2	Gefahr rechts F1
0	0	0	1	
0	0	1	1	
0	1	0	1	
0	1	1	0	0 +)
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	0	

+) Wie auf dem linken Ufer droht auch hier keine Gefahr, wenn der
Bauer anwesend ist.

Dieser Logik-Baustein wird nach den Ergebnissen der Funktionstabelle programmiert und enthält somit die Teilbedingungen für die Gefahr am linken Ufer, wenn der Bauer rechts ist.

Gefahr links F2	
1	f1 = 1
1	
1	f2 = \bar{C}
0	
0	f3 = 0
0	
0	f4 = 0
0	



Die beiden Logik-Bausteine für die Teilbedingung "Gefahr am linken Ufer" und "Gefahr am rechten Ufer" müssen nun noch mit zwei weiteren verbunden werden, die die Verknüpfung der Gefahren an beiden Ufern mit dem Aufenthalt des Bauern herstellen.

	Bauer A	Teilbedingung Gefahr rechts B		Gefahr rechts F3	
Bauer links	0	0		0	f1 = 0
	0	1		1	f2 = 1
Bauer rechts	1	0	+	0	f3 = 0
	1	1		0	f4 = 0

+) Keine Gefahr, da der Bauer ebenfalls rechts ist.

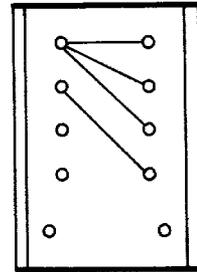
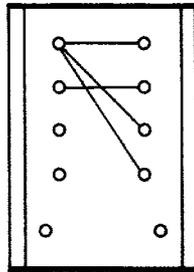
Funktionstabelle für den Aufenthalt des Bauern und die Teilbedingung für Gefahr links:

Bauer A	Teilbedingung Gefahr links B		Gefahr links F4	
0	0		0	f1 = 0
0	1	+	0	f2 = 0
1	0		0	f3 = 0
1	1		1	f4 = 1

+) Keine Gefahr, da der Bauer ebenfalls links ist.

Programm Logik-Baustein 3

Programm Logik-Baustein 4



Wenn Sie die vier Logik-Bausteine bereits programmiert haben, brauchen Sie diese nur noch nach der folgenden Abbildung mit der Eingabeeinheit zu verbinden.

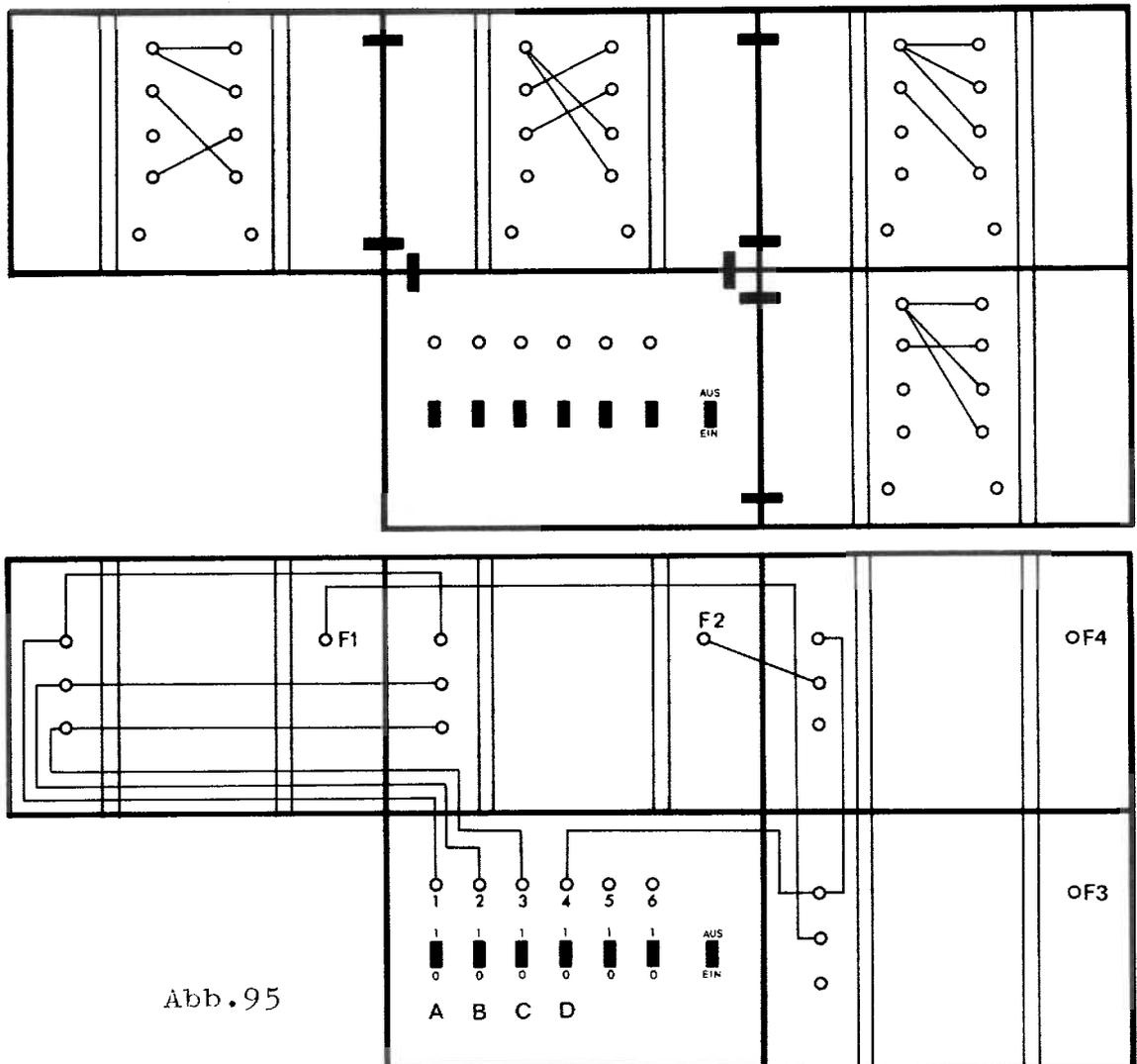


Abb. 95

Mit dieser Schaltung können Sie nun versuchen, die Lösung zu finden, die es dem Bauern ermöglichte, seine Güter gefahrlos von einem Ufer zum anderen zu transportieren.

Gefahr droht immer, wenn die Lampen F₃ oder F₄ aufleuchten (F₁ und F₂ bleiben unberücksichtigt.)

Eine Lösung sieht so aus (Ausgang vom linken Ufer; A, B, C, D = 0):

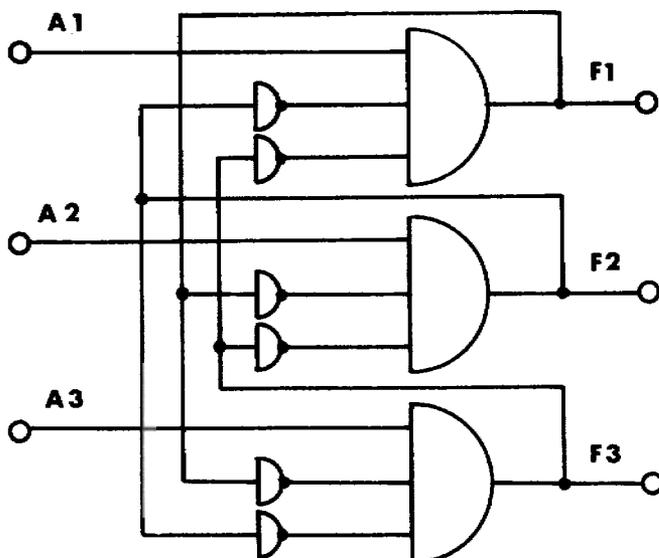
1. Bauer fährt mit der Ziege nach rechts. (D = 1, A = 1)
2. Bauer kehrt leer zurück. (D = 0)
3. Bauer fährt mit dem Kohlkopf nach rechts. (D = 1, C = 1)
4. Bauer nimmt die Ziege nach links zurück. (D = 0, A = 0)
5. Bauer bringt Wolf nach rechts. (D = 1, B = 1)
6. Bauer kehrt leer zurück. (D = 0)
7. Bauer fährt die Ziege nach rechts. (D = 1, A = 1)

Diese Lösung ist nicht die einzig mögliche. Sicherlich gelingt es Ihnen, die anderen mit Ihrem Computer zu ermitteln.

13.9. Reaktionstest mit einer Verriegelungsschaltung

Mit dieser Schaltung können Sie testen, ob Sie schneller sind als Ihre Freunde.

Dazu ist eine Schaltung notwendig, bei der mit drei Schaltern jeweils eine zugehörige Lampe eingeschaltet werden kann. Das allein reicht natürlich noch nicht, denn so würden schließlich alle drei Lampen leuchten. Sobald also die erste Lampe eingeschaltet wird, müssen die anderen gesperrt werden. Die Programmierung muß so gewählt werden, daß durch den Ausgang des einen die Eingänge der anderen Bausteine verriegelt werden. Das geschieht durch Inverter.



Sie erkennen aus dieser Schaltung, daß es sich jeweils um eine UND-Funktion mit 2 negierten Eingängen handelt. Für jede dieser Funktionen kann eine Tabelle aufgestellt werden.

A1	F2	F3	F1
A2	F1	F3	F2
A3	F1	F2	F3
A	B	C	F

0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Sie erkennen aus dieser Tabelle, daß z.B. F1 nur 1 sein kann, wenn lediglich A1 = 1 ist. Entsprechendes gilt für F2 und F3. Daraus ergibt sich, daß die Programmierung aller drei Logik-Bausteine gleich ist. Durch die Verknüpfung der Ausgänge mit jeweils zwei Eingängen wird erreicht, daß z.B. F2 und F3 verriegelt werden, wenn A1 zuerst geschaltet wurde. Wenn allerdings zufällig zwei oder drei Schalter gleichzeitig gedrückt werden, leuchten die entsprechenden Ausgänge zusammen auf. Hier die vollständige Schaltung.

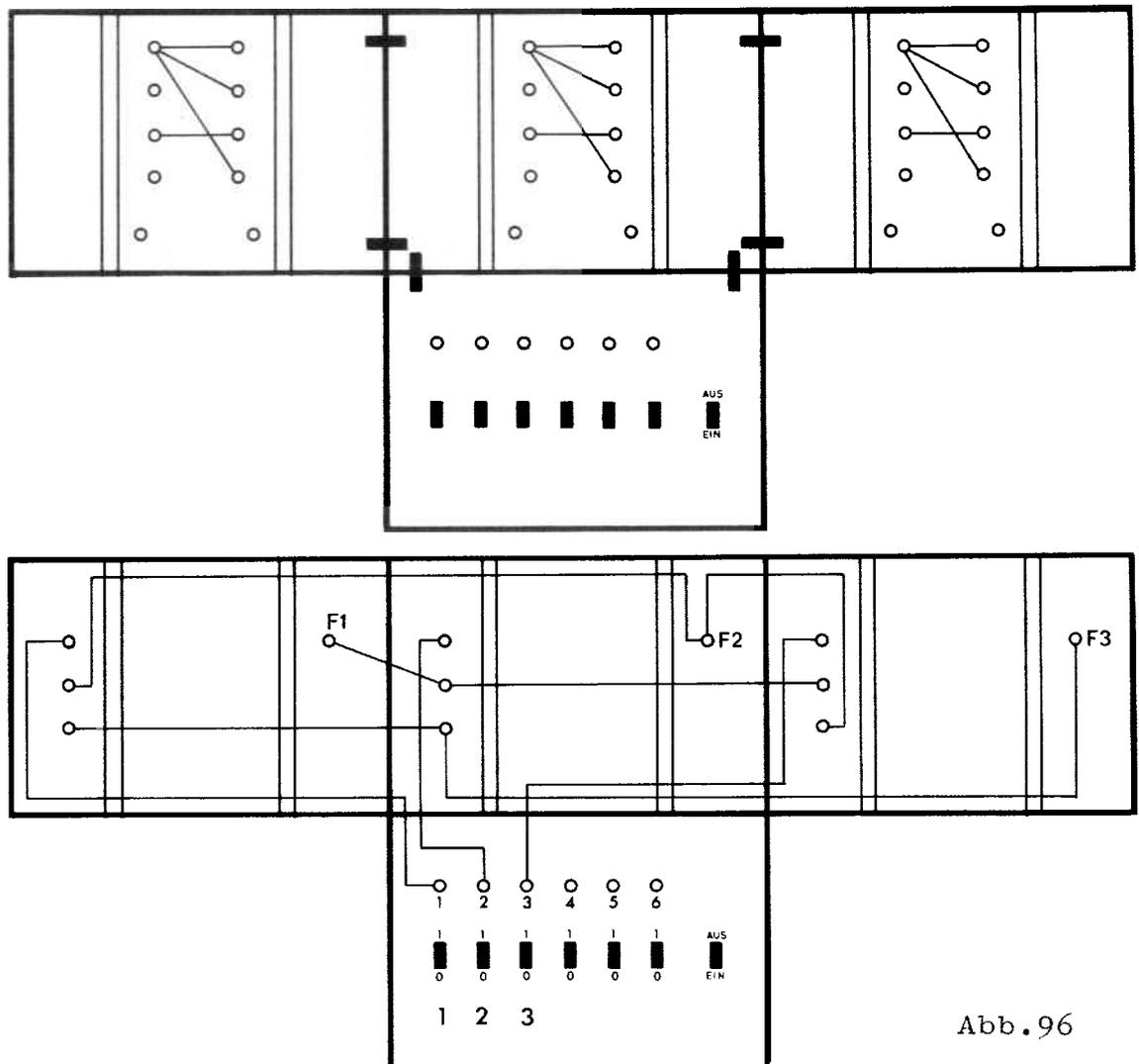


Abb. 96

13.10. Wer gewinnt gegen den Computer?

Ein Zahlenspiel

An diesem Spiel können zwei Spieler teilnehmen. Einer bedient sich dabei des Computers, und der andere tritt gegen diese "Mannschaft" an.

Die Spielregeln: Der erste Spieler nennt eine Zahl von 1 bis 3, und der nächste zählt eine Zahl, ebenfalls von 1 bis 3, hinzu. Abwechselnd addieren die Spieler eine dieser Zahlen, und wer schließlich 33 oder mehr erreicht, der hat verloren.

Zur Erläuterung hier ein Spiel ohne die Hilfe des Computers:

Spieler	A	fängt an mit z.B.	1		
Spieler	B	addiert	2	= 3	
	A		+ 3	6	
	B		3	9	
	A		1	10	
	B		3	13	
	A		3	16	
	B		3	19	
	A		2	21	
	B		3	24	
	A		1	25	
	B		1	26	
	A		3	29	
	B		2	31	
	A		1	32	
	B		1	33	B hat verloren.

Wenn Sie dieses Spiel mit dem Computer spielen, sind Sie nicht zu schlagen, sofern Sie Ihren Gegenspieler beginnen lassen. Wenn Sie selbst anfangen, macht Sie der Computer auf Fehler Ihres Gegenspielers aufmerksam, und dann können Sie ebenfalls nicht mehr verlieren. Der Computer sagt Ihnen nämlich immer die richtige Zahl, die Sie addieren müssen.

Zu dieser Schaltung benötigen Sie vier Logik-Bausteine, die zu zwei Binärzählern programmiert werden. Baustein 1 und 2 sowie 3 und 4 stellen je einen Binärzähler dar. Das Ergebnis, das Sie jeweils addieren müssen, wird an den Ausgängen F2 und F4 - und zwar dual - angezeigt. Dabei ist der dezimale Wert für F2 -2- und für F1 -1-.

Sie müssen während des Spiels den Schalter S so oft von 0 nach 1 und zurück nach 0 schieben, wie es der von Ihnen oder Ihrem Mitspieler addierten Zahl entspricht (z.B. +3 = dreimal Schalter S betätigen).

Aus der nachfolgenden Tabelle können Sie entnehmen, welche Zahl Sie bei den Ausgangszuständen F2 und F4 addieren müssen.

F2 F4

0 0 = 0 bedeutet, daß Sie eigentlich eine 0 hinzuzählen müßten. Da das nicht erlaubt ist, wählen Sie eine beliebige Zahl. In dieser Situation ist Ihr Gegner im Vorteil.

1 1 = 3 bedeutet, daß Sie eine 3 addieren müssen. Jetzt kann Sie der Gegner nicht mehr schlagen.

1 0 = 2 eine 2 muß addiert werden, und Sie sind nicht mehr zu schlagen.

0 1 = 1 Sie müssen eine 1 hinzuzählen, um zu gewinnen.

Programmieren Sie bitte zunächst Ihre Logik-Bausteine, bevor wir Ihnen ein Beispiel mit Computerunterstützung bringen.

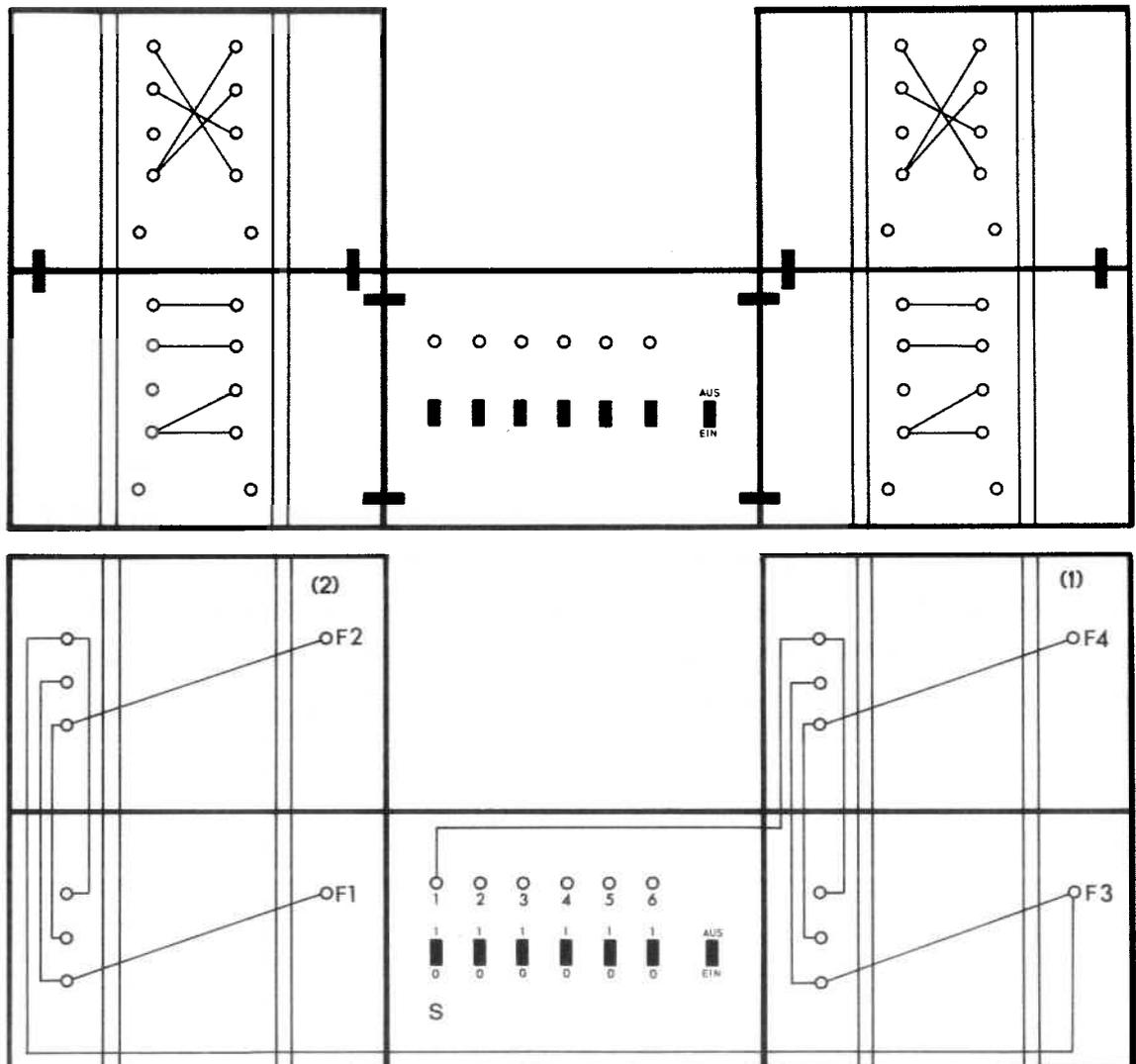


Abb.97

Und hier das Spiel: Zunächst müssen Sie die Flipflops mit dem Schalter S auf die Ausgangsposition 00 stellen. Alle Lampen sind in diesem Fall dunkel.

Der Spieler A spielt mit dem Computer, Spieler B allein. - A fängt an:

<u>Spieler</u>	<u>wählt</u>	<u>Spieler A schaltet</u>	<u>Computer zeigt .. an</u>	<u>A muß .. sagen</u>
A	2	2 x	1 0	-
B	+ 1 = 3	1 x	0 1	1
A	+ 1 = 4	1 x	0 0	-
B	+ 3 = 7	3 x	0 1	1
A	+ 1 = 8	1 x	0 0	-
B	+ 3 = 10	2 x	1 0	2
A	+ 3 = 12	2 x	0 0	-
B	+ 1 = 13	1 x	1 1	3
A	+ 3 = 16	3 x	0 0	-
B	+ 3 = 19	3 x	0 1	1
A	+ 1 = 20	1 x	0 0	-
B	+ 2 = 22	2 x	1 0	2
A	+ 2 = 24	2 x	0 0	-
B	+ 2 = 26	2 x	1 0	2
A	+ 2 = 28	2 x	0 0	-
B	+ 3 = 31	3 x	0 1	1
A	+ 1 = 32	1 x	0 0	-

B muß an dieser Stelle 33 oder mehr sagen und hat deshalb verloren.

Das Prinzip dieses Spiels besteht darin, dem Gegenspieler eine Zahl weiterzugeben, die ein Vielfaches von 4 beträgt. Wenn Sie einmal eine solche Zahl erreicht haben, so müssen Sie nur noch die Zahl, die der Mitspieler nennt, auf 4 ergänzen, und damit erreichen Sie stets ein Vielfaches von 4. Der Computer zeigt diese Zahl an. Er zählt auch jeweils nur bis 4, gleichgültig, ob Sie bereits bei 8, 12, 16, 20 usw. sind.

Wenn Sie die Schaltung nach der folgenden Abbildung mit dem fünften Logik-Baustein versehen, kann die Anzeige erweitert und vereinfacht werden. Jetzt zählt jede Lampe als 1. Die Zahl der aufleuchtenden Lampen der Logik-Bausteine 2, 4 und 5 gibt die Zahl an, die Sie addieren müssen.

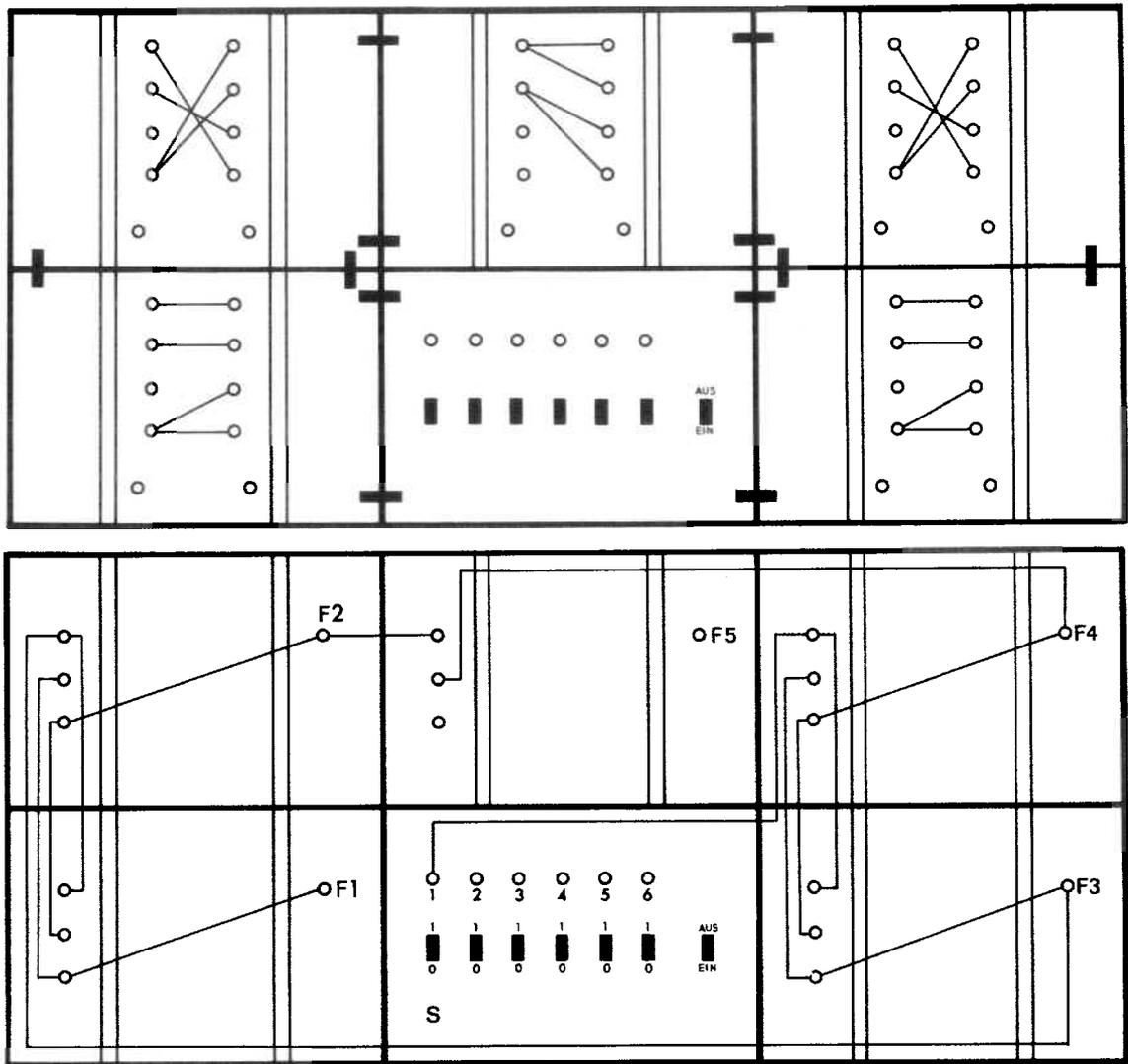
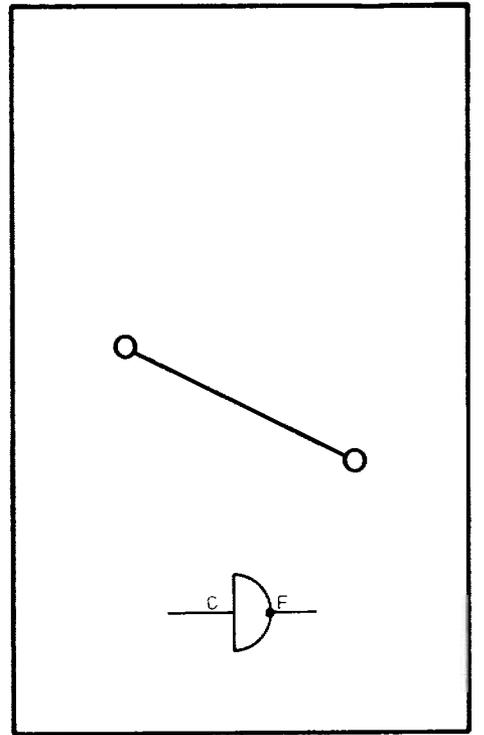
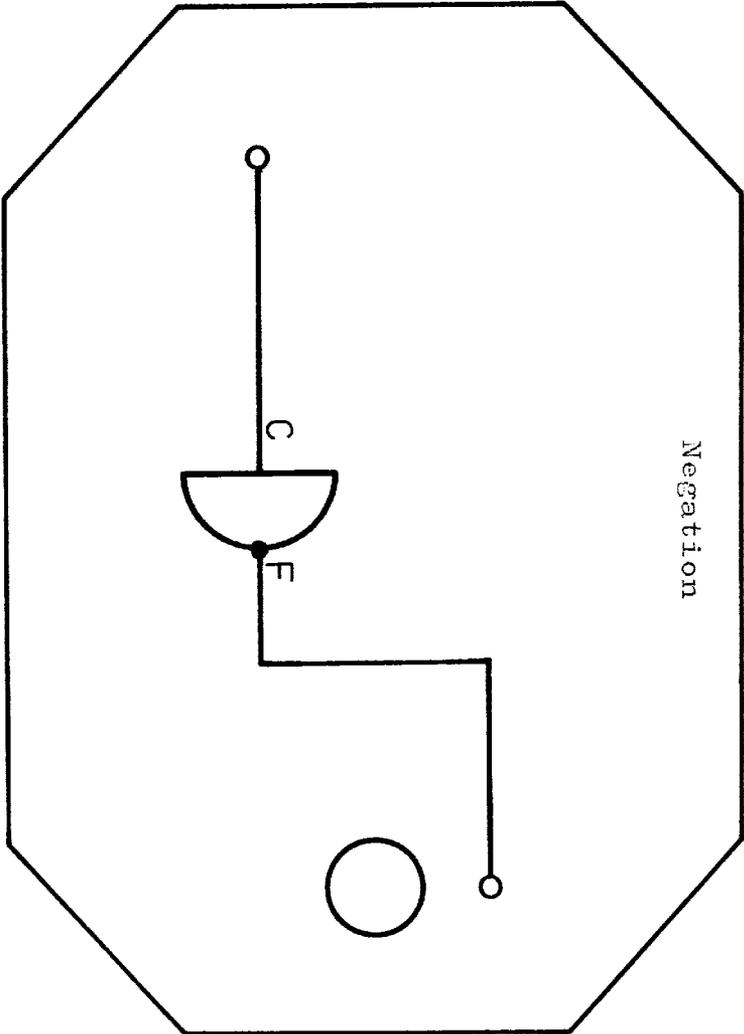
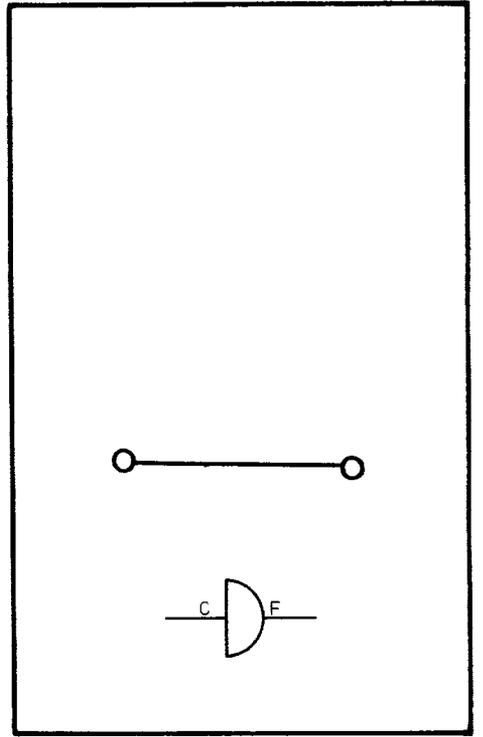
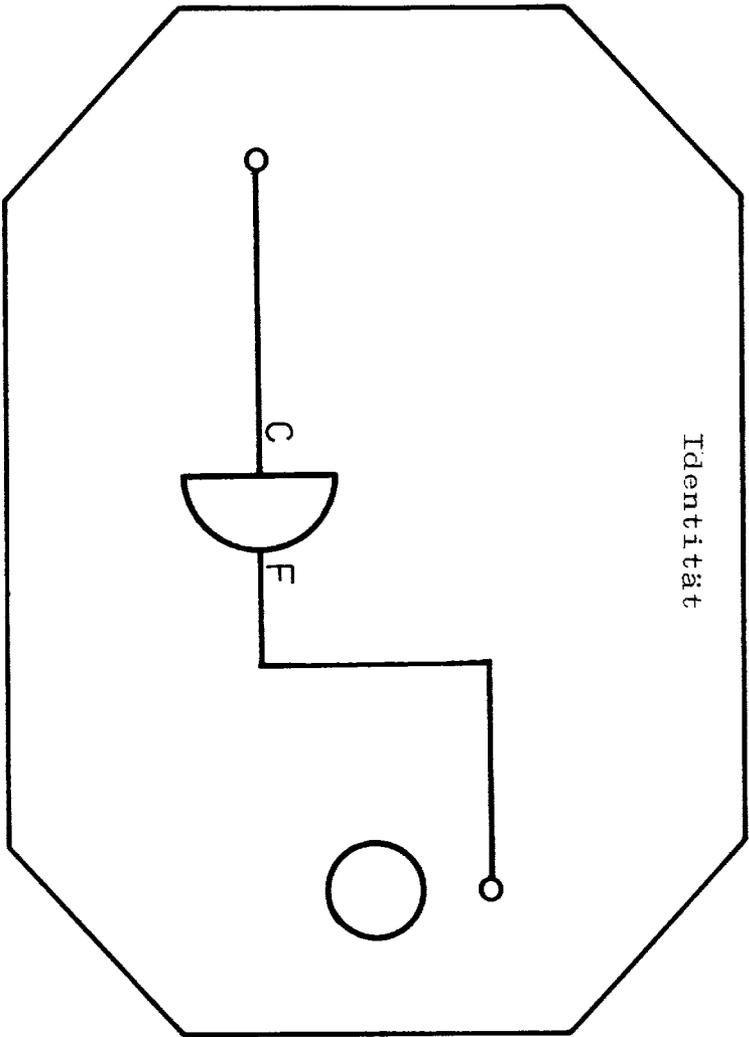


Abb. 98

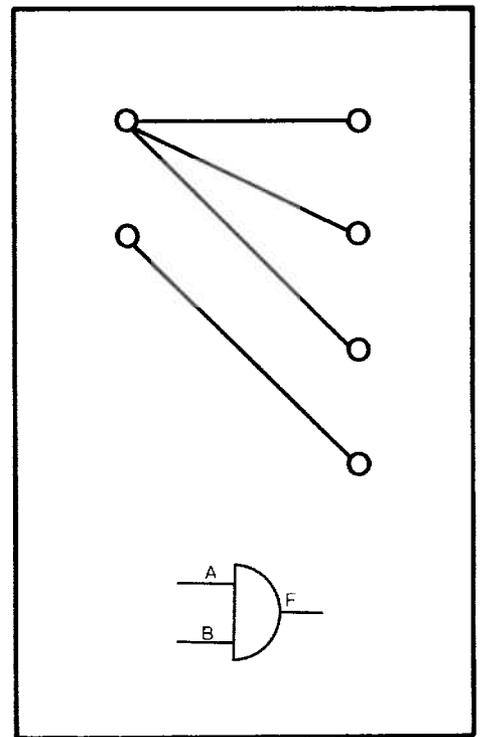
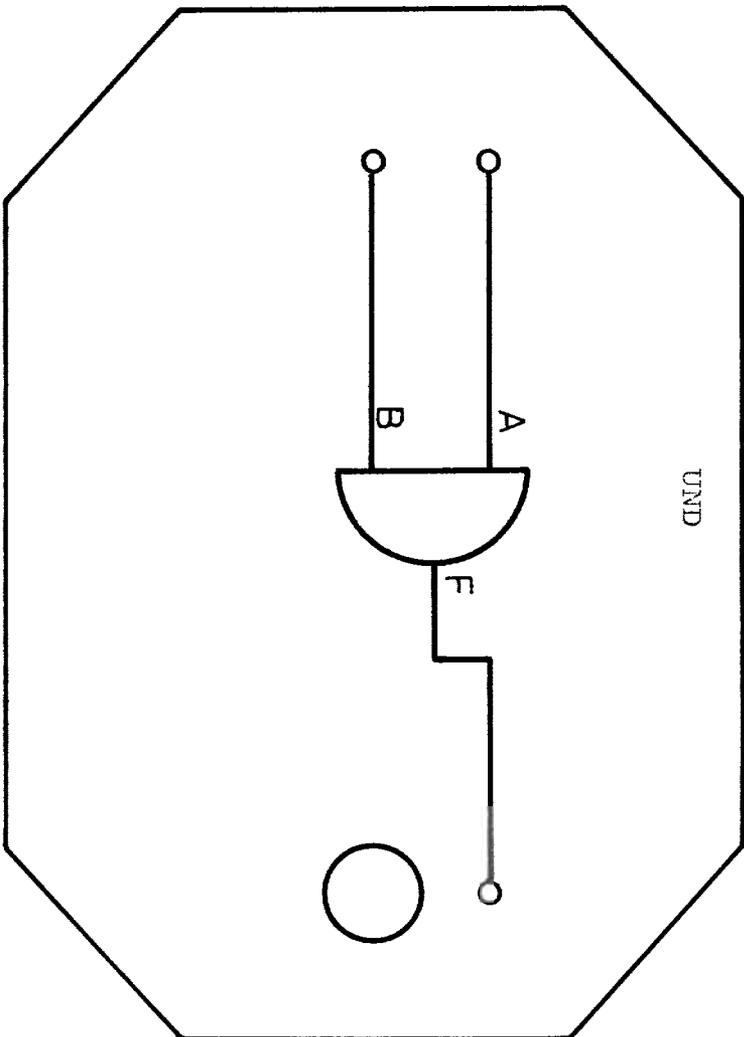
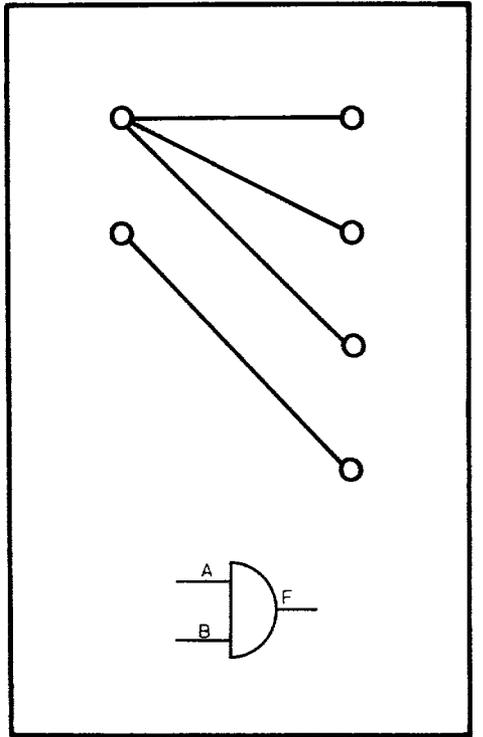
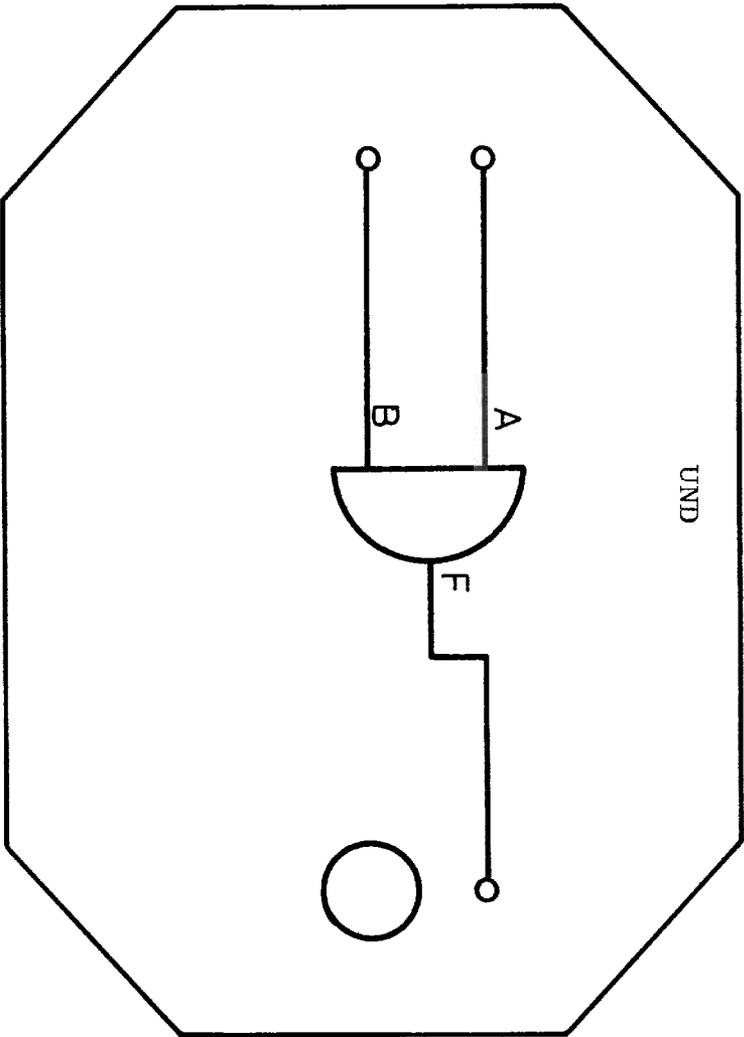
Der Computer-Lehrbaukasten CL 1601 ist
der Grundkasten eines ausbaufähigen
Computer-Lehrprogramms.

Wir werden uns erlauben, Ihnen Zusatz-
kästen und weitere Spiele sowie neue
Schaltungen nach Erscheinen anzubieten.



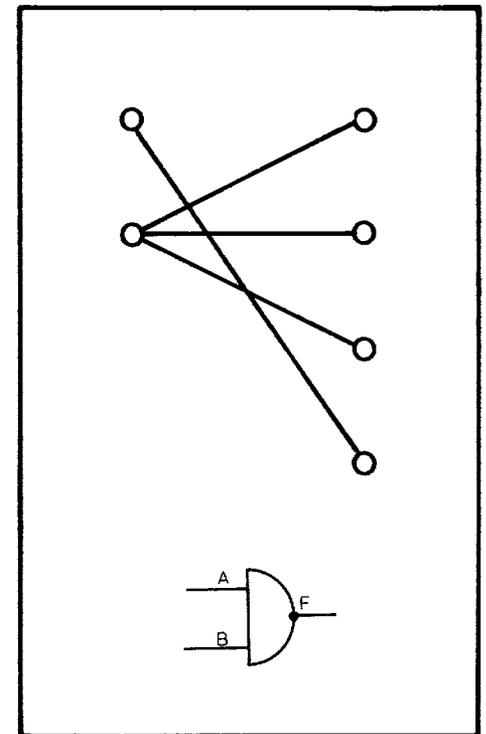
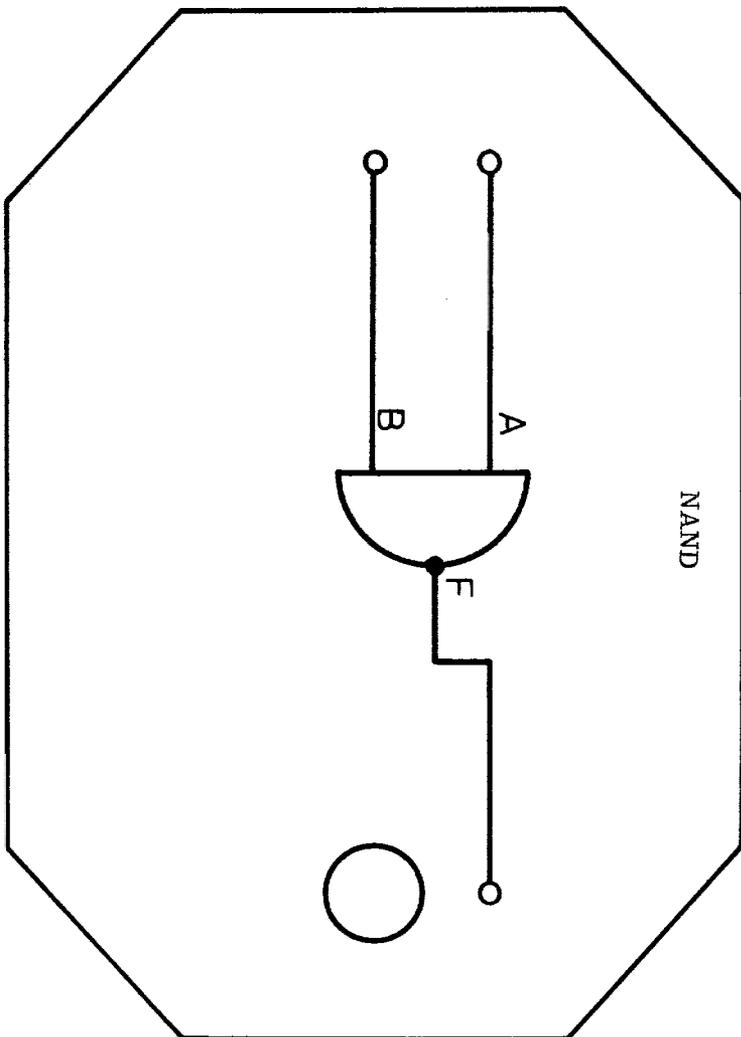
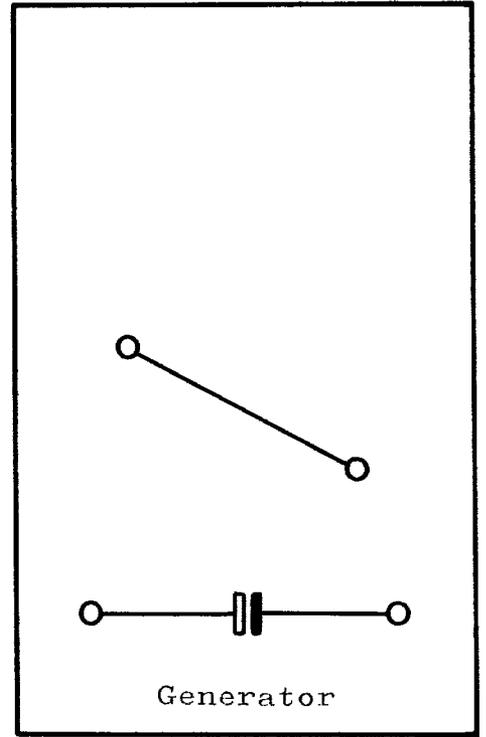
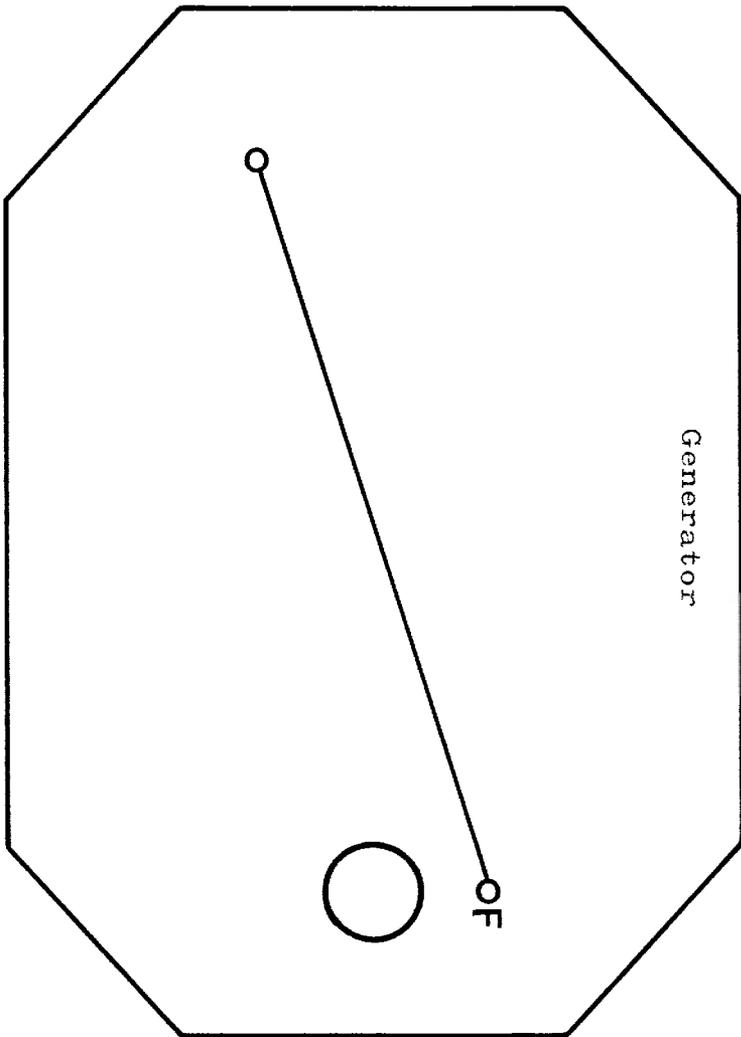
(

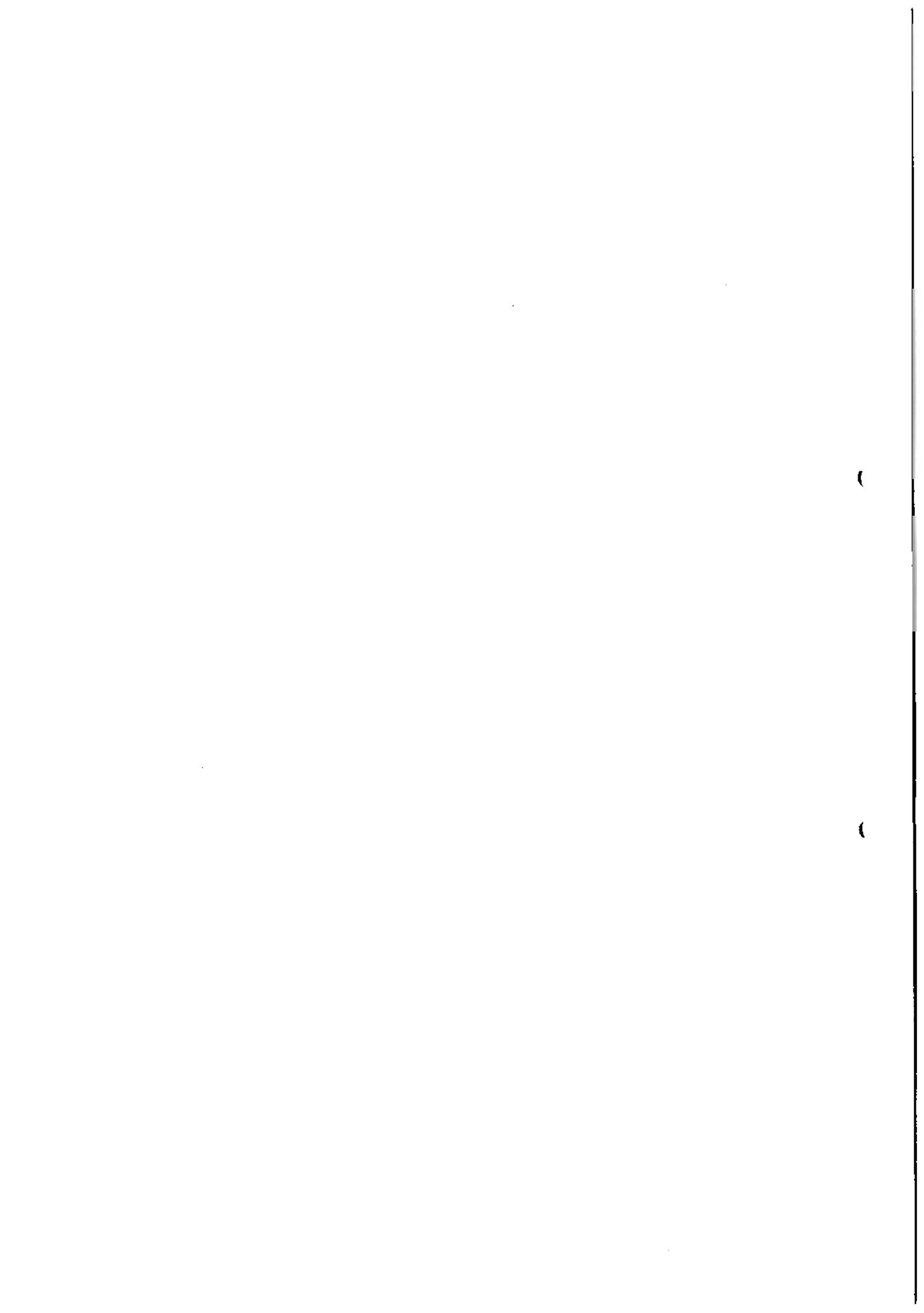
(

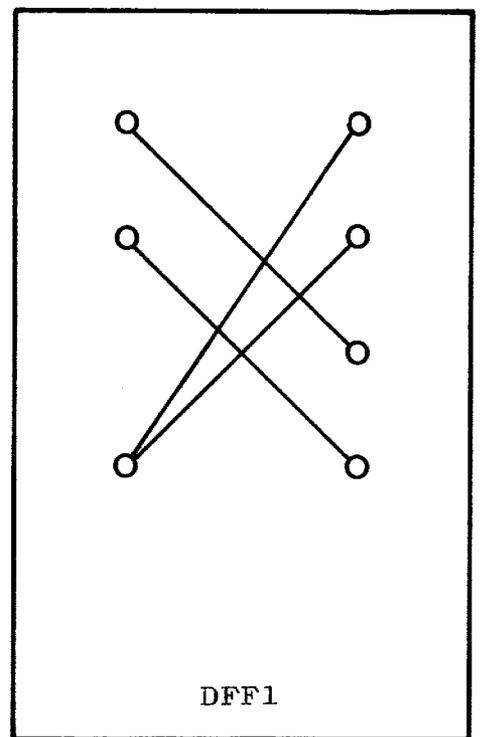
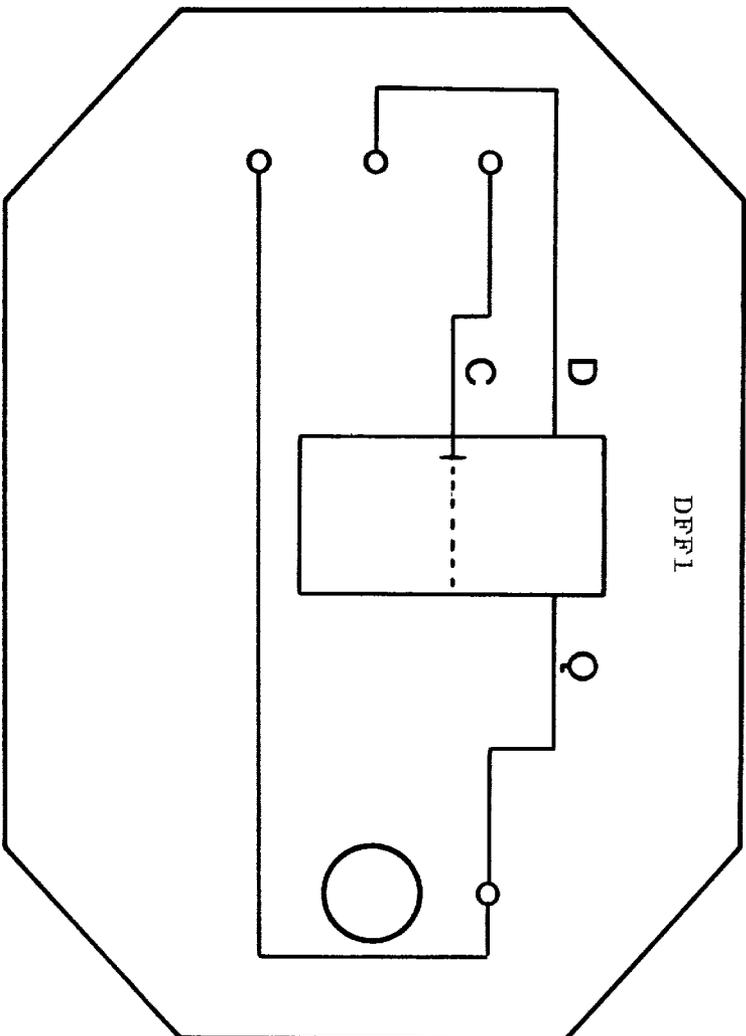
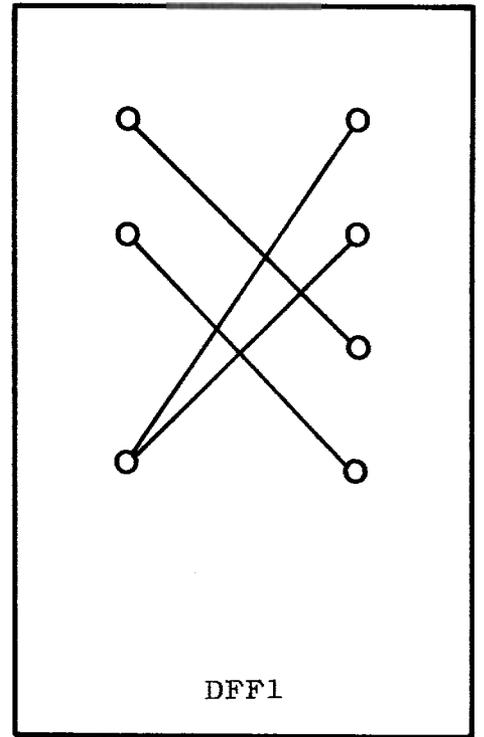
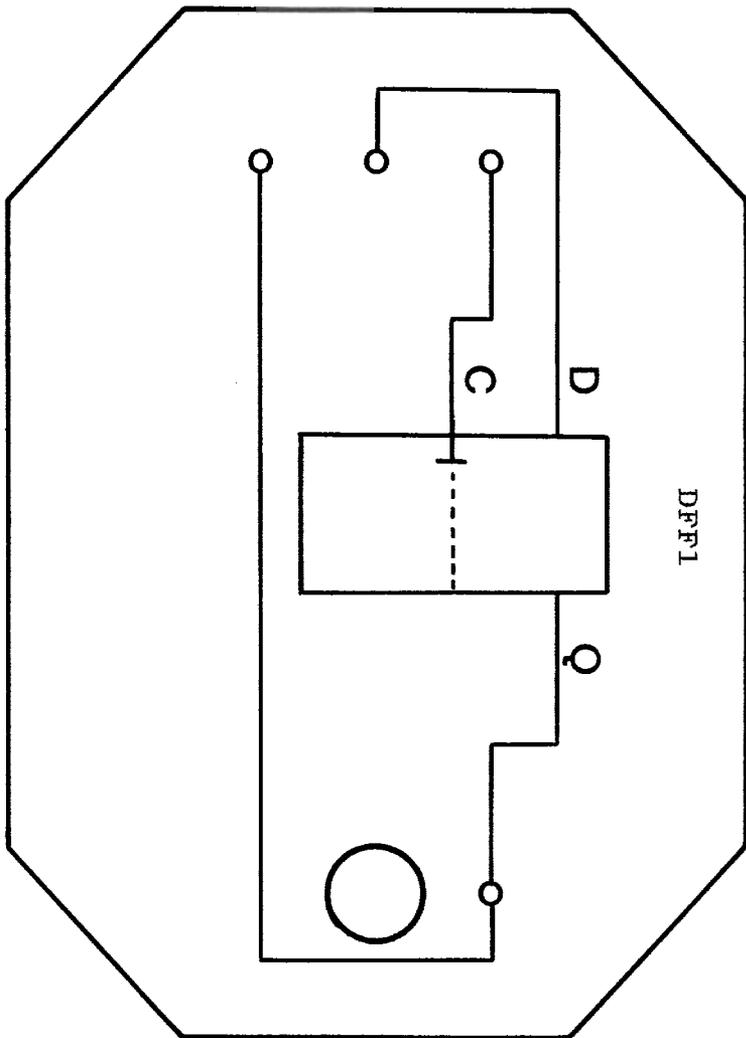


(

(

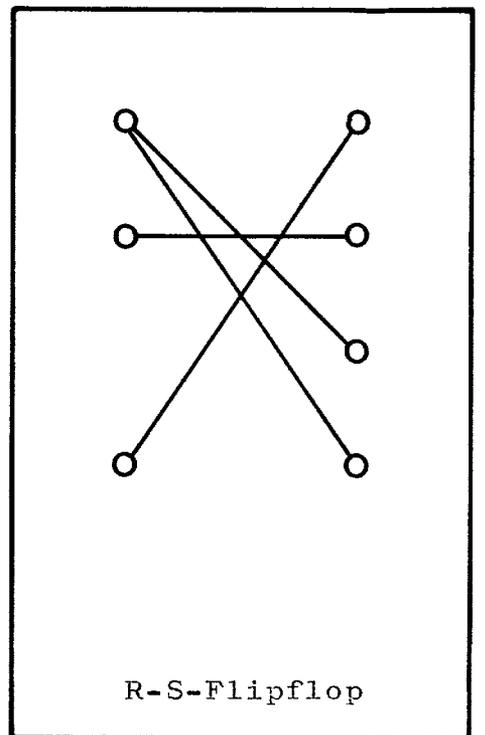
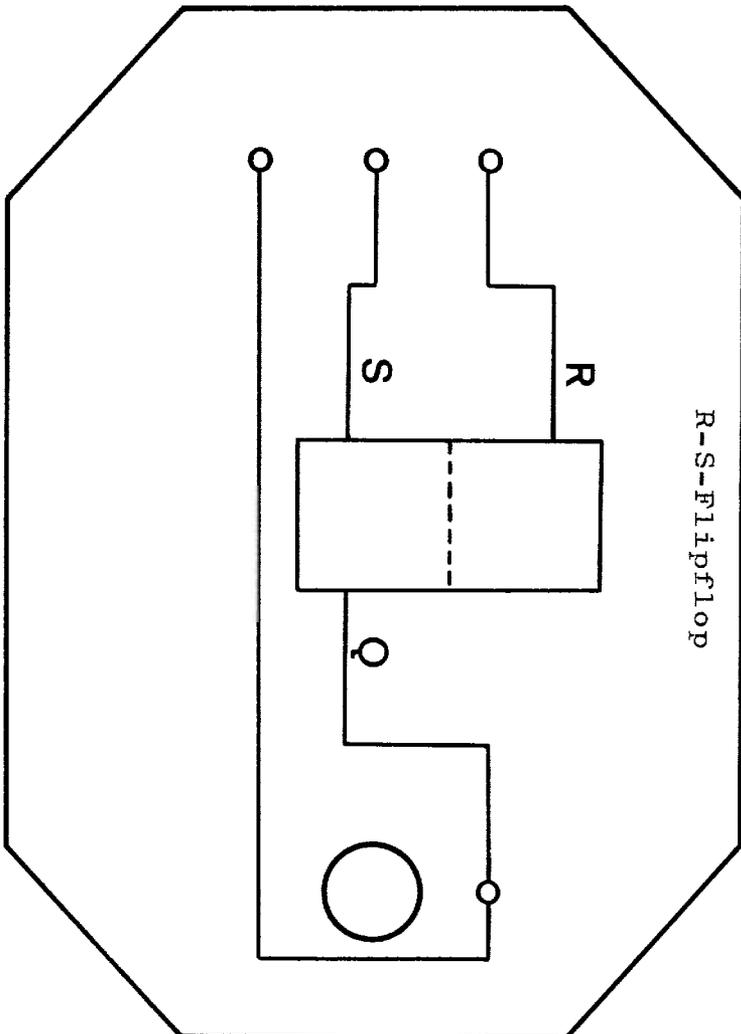
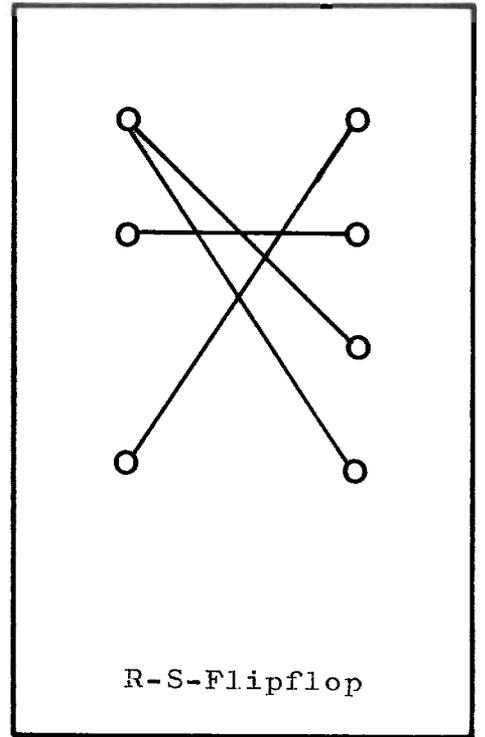
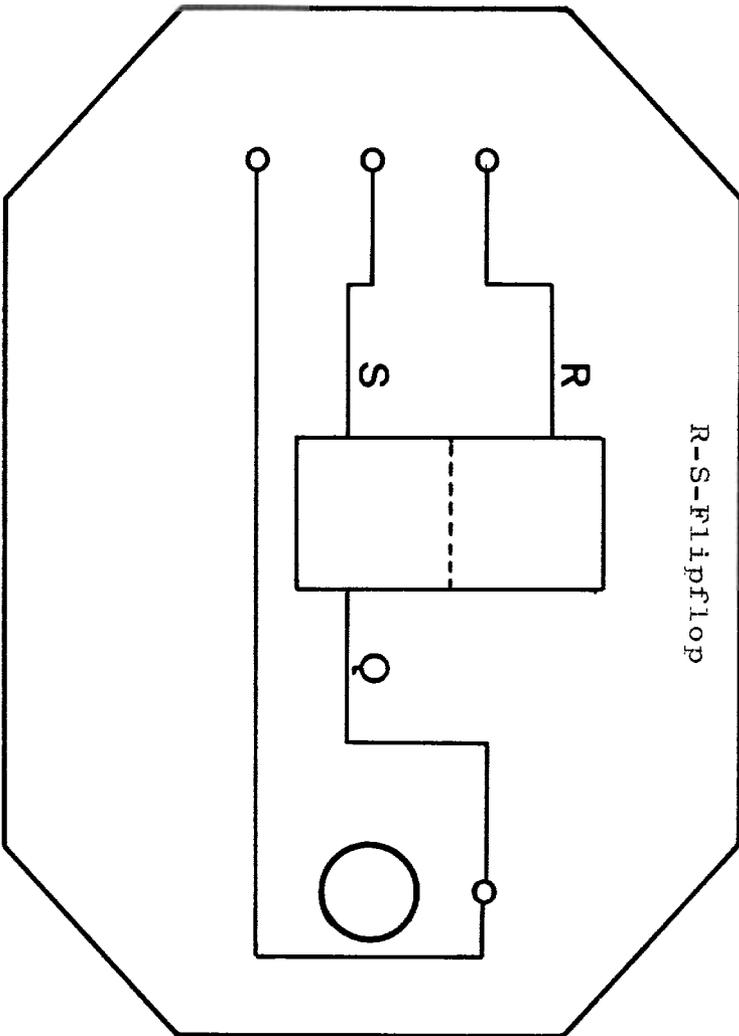






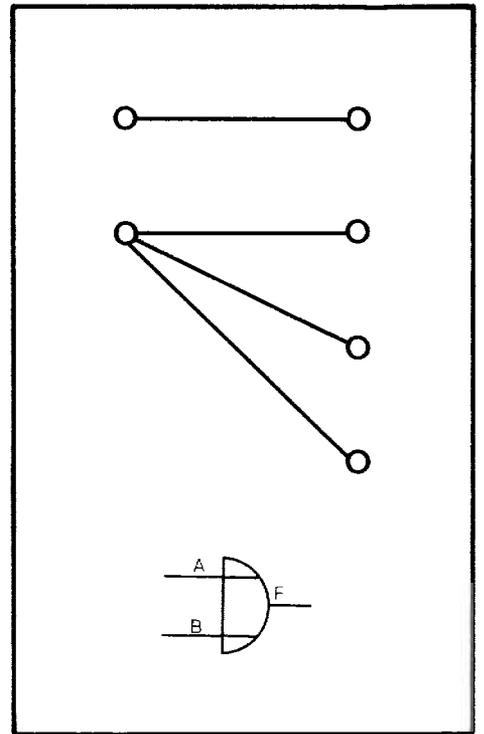
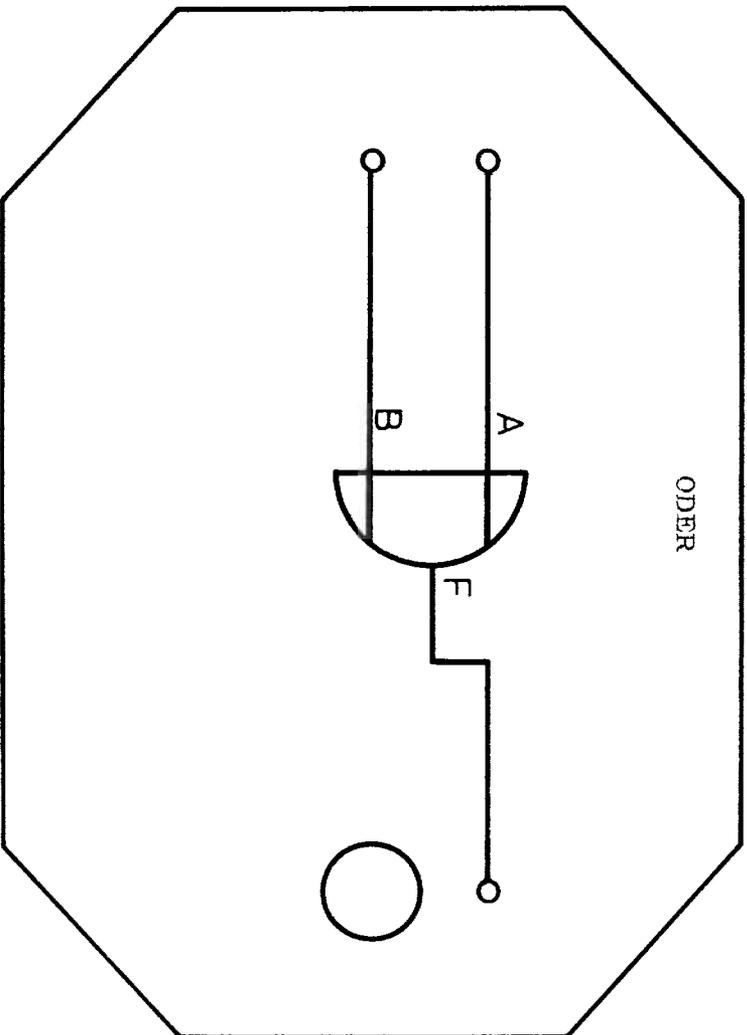
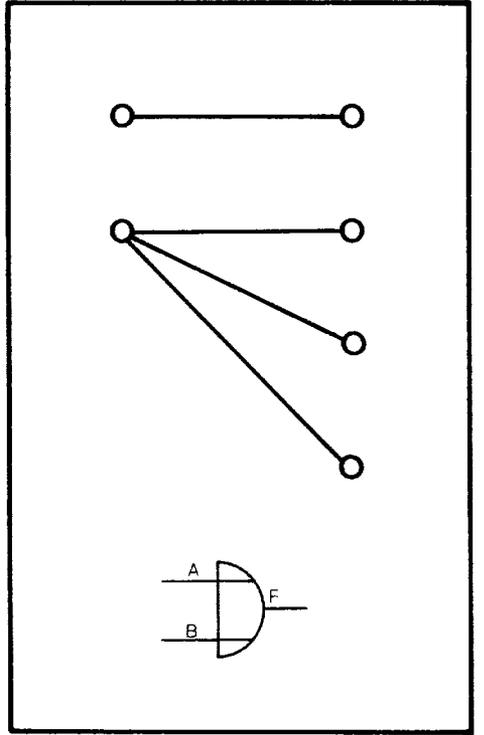
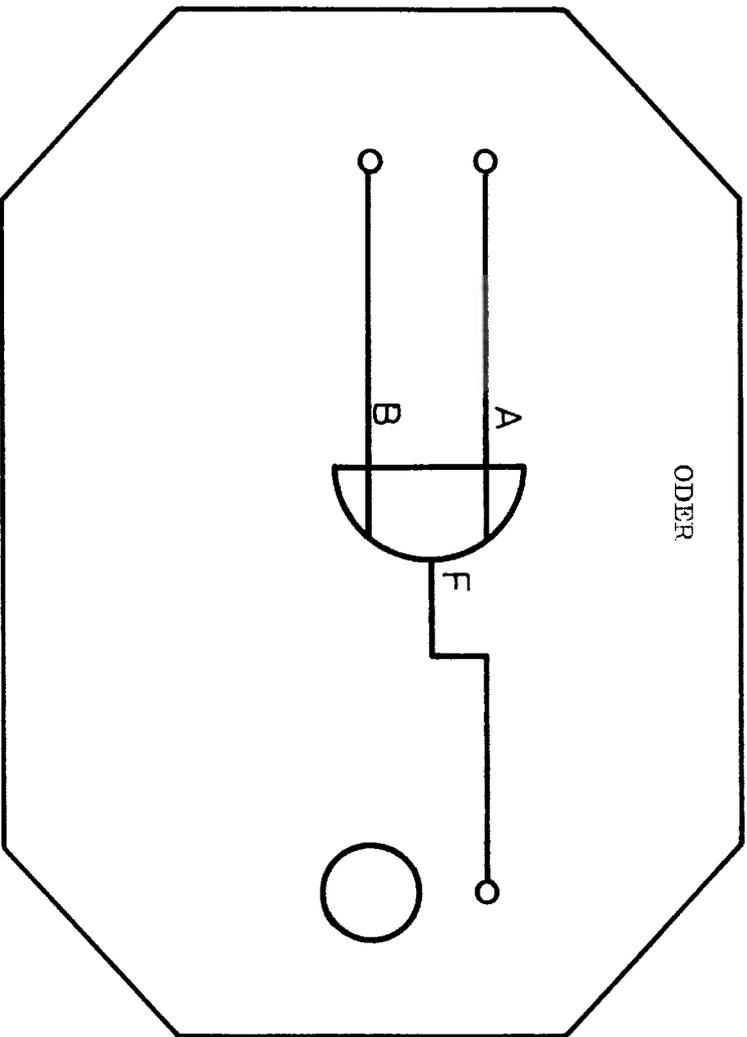
(

(



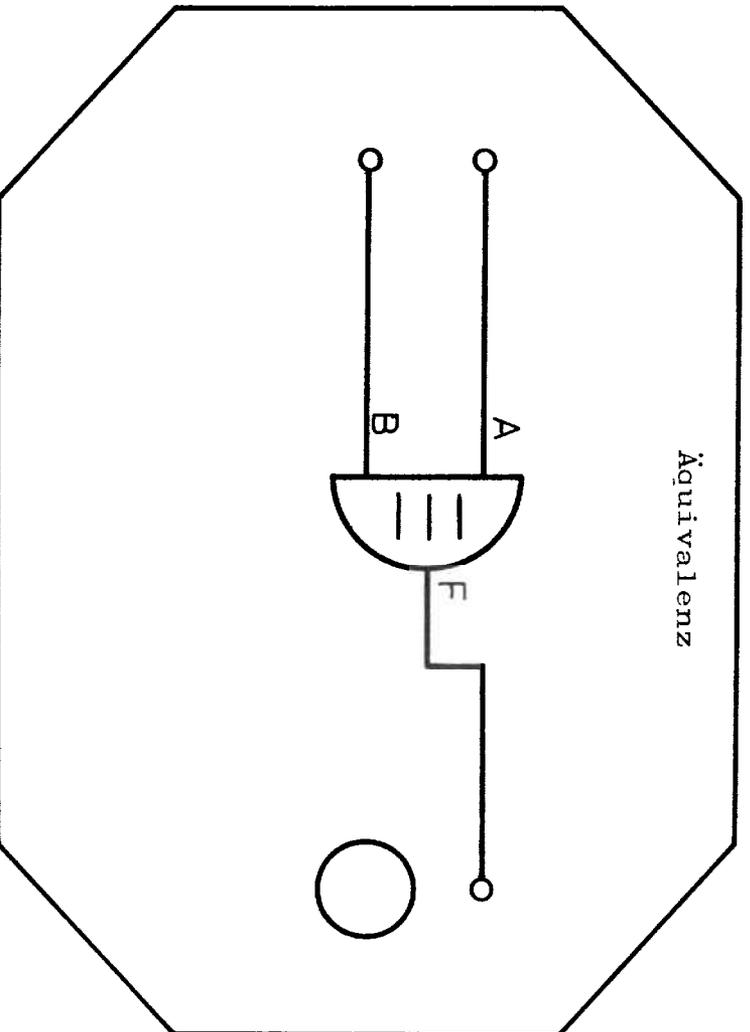
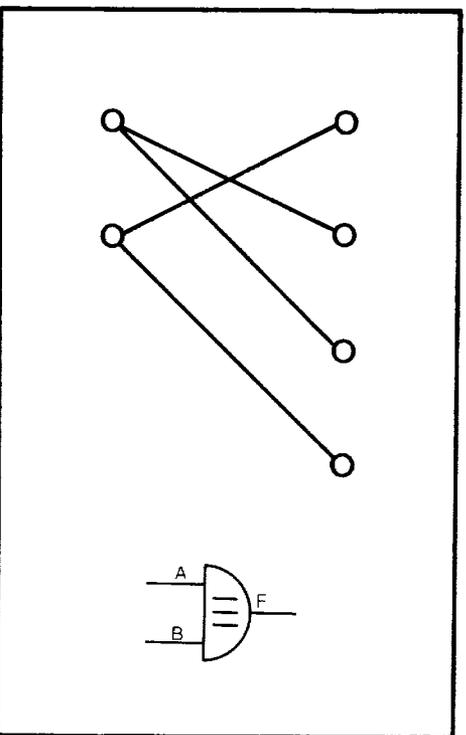
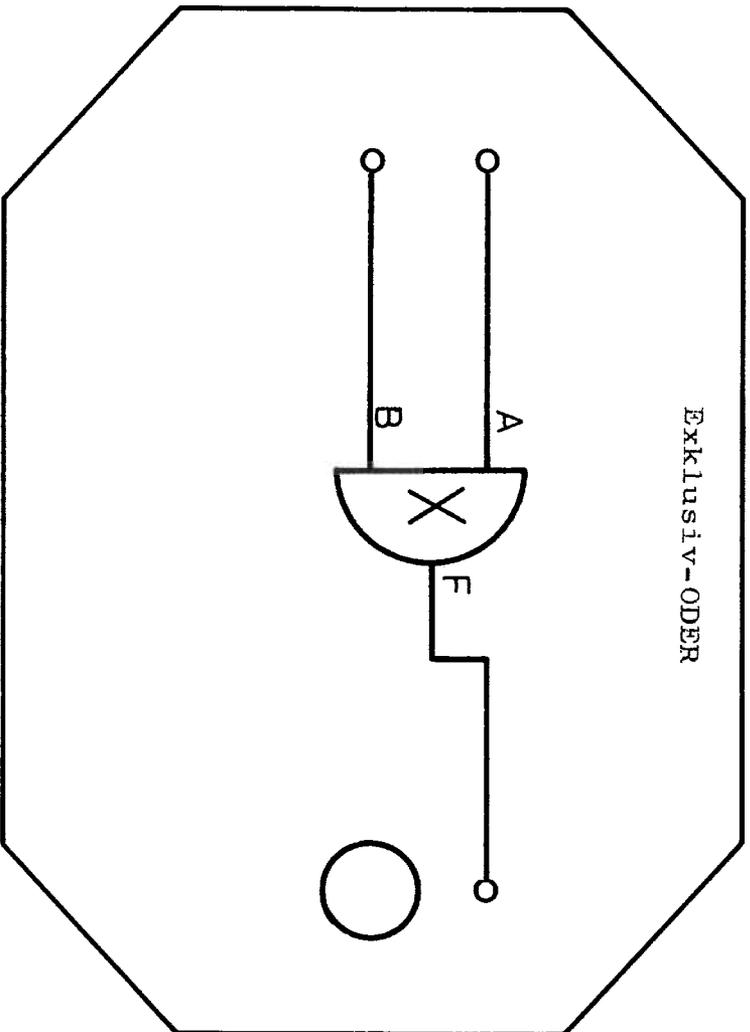
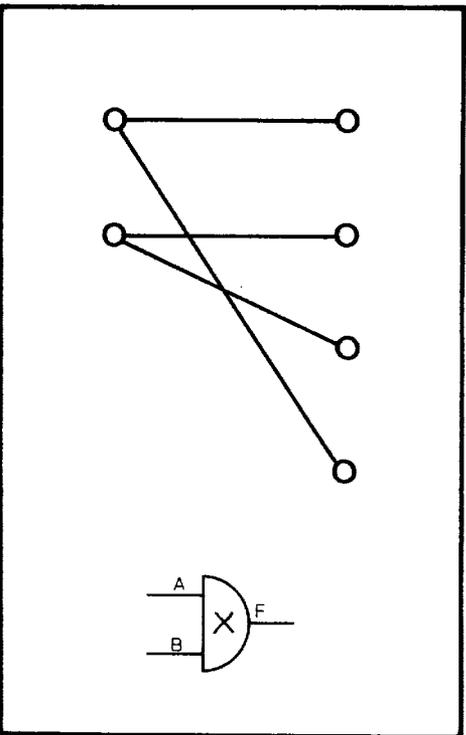
(

(



(

(



(

(

