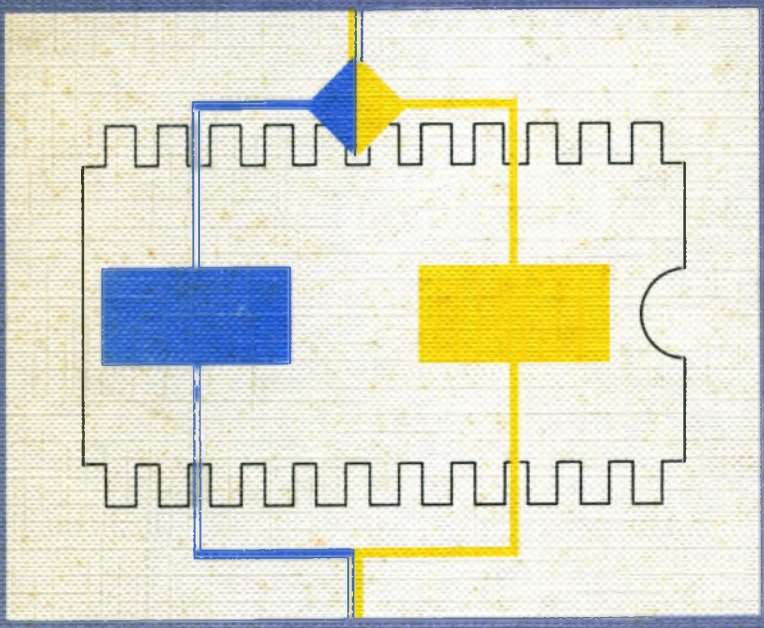
Franzis
Computer-Praxis

Klein
Mikrocomputer selbstgebaut und programmiert

Vom Bauelement zum fertigen Z-80-Computer



Stromversorgungen
Definition der Signalpegel
Flip-Flop-Schaltungen
Fragen zur Digitaltechnik
Messen an Digitalschaltungen
Aufbau eines TTL-Prüfstiftes
Der Mikrorechner
Speicherbausteine
Die Z-80-CPU
Die SBC-Karte
ROM/RAM-Karte
Serielles Interface
Sound-Generator
Aufbau eines Daten-Sichtgerätes
Assembler
CRT-Controller-Software
Strukturierte Programmierung
Pinbelegung und Timings
Tastatur – Anschluß

Franzis'

Klein
Mikrocomputer selbstgebaut und programmiert

In der Gruppe

Computer-Praxis

sind erschienen:

Busch, Basic für Einsteiger
Haugg, Software-Engineering
Klein, Basic-Interpreter
Klein, Mikrocomputersysteme
Klein, Mikrocomputer Hard- und Softwarepraxis
Klein, Z-80-Applikationsbuch
Klein, Was ist Pascal?
Klein, Mikrocomputer selbstgebaut
Piotrowski, IEC-Bus
Plate, Pascal: Einführung – Programmentwicklung – Strukturen
Wunderlich, Erfolgreicher mit CBM arbeiten

Franzis Computer-Praxis

Rolf-Dieter Klein

Mikrocomputer selbstgebaut und programmiert

Vom Bauelement zum fertigen Z-80-Computer

Mit 332 Abbildungen

Franzis'

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Klein, Rolf-Dieter:
Mikrocomputer selbstgebaut und programmiert: vom Baulement zum fertigen Z-80-Computer /
Rolf-Dieter Klein. – München: Franzis, 1983.
(Franzis-Computer-Praxis)
ISBN 3-7723-7161-2

Satz: Composer-Satz Nowak, 8137 Berg/Starnberger See
Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2

Printed in Germany · Imprimé en Allemagne

ISBN 3-7723-7161-2

Vorwort

Der Mikrocomputer dringt in alle Bereiche des Lebens ein. Daher kann es nur von Nutzen sein, sich mit dieser neuen Technik vertraut zu machen. Für den Anfänger ist es aber schwierig, überhaupt einen Einstieg zu finden. Gerade fertige Systeme sind sehr teuer und meist schlecht dokumentiert, Schaltpläne werden oftmals gar nicht mitgegeben und Software ist ein streng gehütetes Geheimnis. In dem vorliegenden Buch wird ein Experimentiersystem für Mikrocomputer vorgestellt, das dieses Übel beseitigen soll. Alle Mikrorechnerschaltungen sind mit eigens für das Buch entwickelten Platinen unterstützt. Aus den einzelnen Platinen kann ein kompletter Mikrorechner entwickelt werden, der bis zum Vollausbau auf 64 K-Speicher erweiterbar ist. Durch das modulare Konzept der Platinen gibt es eine Vielzahl von Kombinationsmöglichkeiten. Zu allen Schaltungen ist eine ausführliche Test- und Aufbauanleitung gegeben. Aber auch der Selbstbau der Platinen ist zumindest dem geübteren Bastler möglich, denn alle Layouts sind abgedruckt. Um die Platinen nachbauen zu können, braucht er aber keine Vorkenntnisse in der Mikrocomputerei zu haben. Im Softwareteil finden sich ausführliche Ausdrucke aller im Rechner verwendeten Programme.

Ziel des Buches ist es von Anfang an die Mikrocomputertechnik zu verstehen und zu begreifen. Der Weg dazu führt über viele Schaltungen und Programmbeispiele. Am Ende eines Kapitels gibt es Fragen, um den Lernerfolg zu überprüfen. So wird das Buch auch zu einem Lehr- und Lernbuch.

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden*).
Alle Schaltungen und technischen Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, daß sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung eventueller Fehler sind Autor und Verlag jederzeit dankbar.

---

*) Bei gewerblicher Nutzung ist vorher die Genehmigung des möglichen Lizenzinhabers einzuholen.

Inhalt

1 Stromversorgungen

Eines der wichtigsten und oftmals stiefmütterlich behandelten Kapitel bei Mikrorechnern ist die Stromversorgung. Damit soll hier begonnen werden.

Für ein Mikrorechnersystem werden im allgemeinen bis zu vier verschiedene Versorgungsspannungen benötigt. Einmal +5V für die Versorgung der Mikrorechnerchips und der TTL-Bausteine, dann +/−12 V für einige spezielle Zusatz-Bausteine sowie −5V ebenfalls für einige Bausteine, wie z. B. dynamische Speicher älterer Bauart. Außerdem wird eine 26V-Spannungsquelle für das Programmieren von löschbaren Festwertspeichern benötigt. Als Strombedarf wird für unser System bei der 5V-Versorgung 1A genügen, bei den anderen Spannungen sogar weniger. Daher werden wir eine Stromversorgung mit 1A-Ausgangsstrom aufbauen. Eine Erweiterung auf mehr als 1A ist im Prinzip dann auch möglich.

1.1 5V-Versorgung

Abb. 1.1.1 zeigt die Schaltung einer einfachen 5V-Versorgung. Benötigt wird dazu einmal ein Transformator mit etwa 7,5 V Ausgangsspannung und 1,5 bis 1A Ausgangsstrom. Die Wechselspannung gelangt an einen Brückengleichrichter und dann als Gleichspannung über einen Siebkondensator an einen integrierten Spannungsregler. Dieser Spannungsregler liefert direkt die benötigten 5V-Ausgangsspannungen. Eine integrierte Schaltung empfiehlt sich bei Mikrorechnern, da sie die Spannung sehr genau in den benötigten Grenzen hält. Vor selbstgebauten „Geradenoch-Versorgungen" kann nur gewarnt werden, da der Ärger danach sehr groß wird; die Mikrorechner-Schaltung arbeitet nur hin und wieder richtig und niemand weiß genau, wo der Fehler liegt.

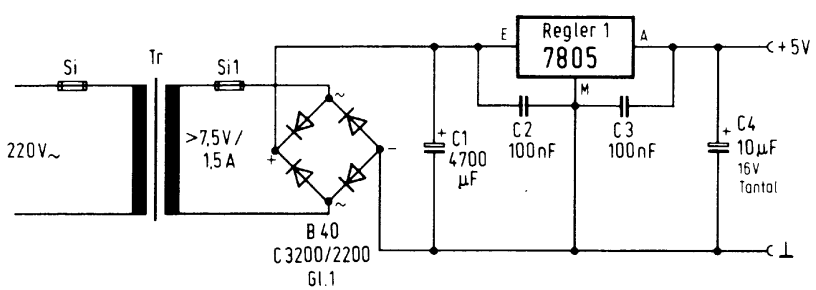


Abb. 1.1.1 Schaltung eines 5V Netzteils

Die beiden Kondensatoren C2 und C3 vermeiden HF-Störungen und Schwingneigungen der integrierten Schaltung. Es sind dort 100 nF keramische Scheiben-Kondensatoren zu verwenden. Keinesfalls dürfen Wickelkondensatoren benützt werden, da sie eine viel zu große Eigeninduktivität besitzen und damit an dieser Stelle völlig wirkungslos sind. Der Kondensator C4 dient nochmals zur Glättung der Ausgangsspannung. Der Regler 7805 schafft ca. 1A. Es gibt aber auch Varianten dieses Reglers die bis zu 10A regeln können. Sie sind allerdings etwas teuerer (ca. 20 bis 30 DM). Dabei müssen dann aber auch Trafo, Gleichrichter und Siebelko (C1) anders ausgelegt werden.

1.2 +/– 12V und 5V-Versorgung

Abb. 1.2.1 zeigt die dazugehörige Schaltung. Benötigt wird ein Trafo mit 2 x 15V Ausgangsspannung und 1A Ausgangsstrom. Mit Hilfe der integrierten Regler 7812, 7912 und 7905 werden dann die drei Ausgangsspannungen erzeugt. An –12V und –5V darf insgesamt nur 1A Strom abgenommen werden, doch da bei Mikrorechnerschaltungen gerade diese Versorgungsspannungen nur sehr gering belastet werden, ist dies kein Problem.

1.3 26V-Versorgung

Abb. 1.3.1 zeigt die Schaltung für eine 26V-Stromversorgung, die später für ein Programmiergerät benötigt wird. Da es keinen 26V-Regler-Baustein gibt, wird hier eine Schaltung

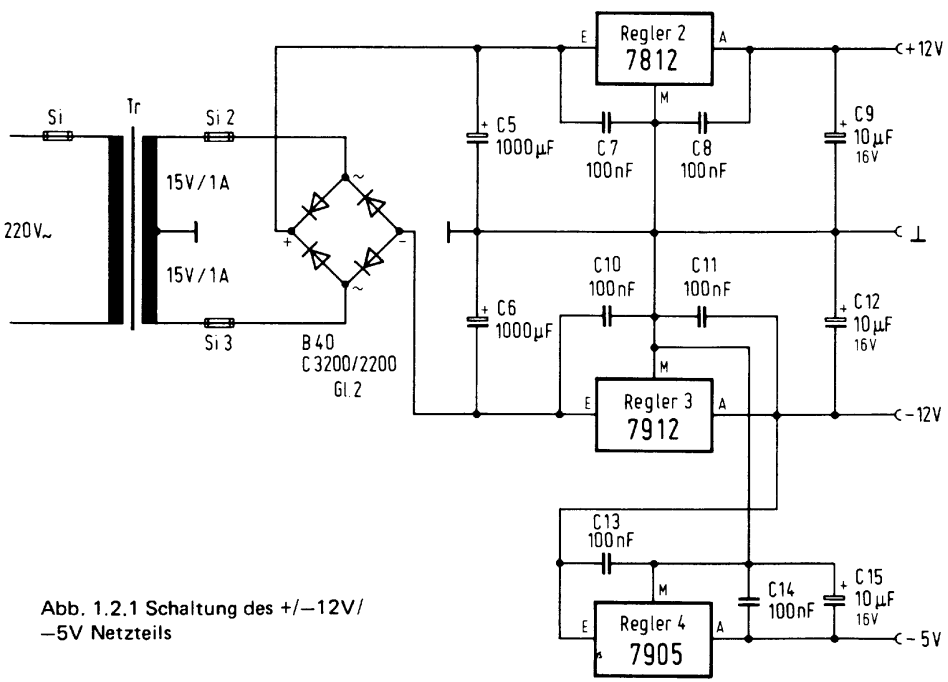


Abb. 1.2.1 Schaltung des +/–12V/ –5V Netzteils

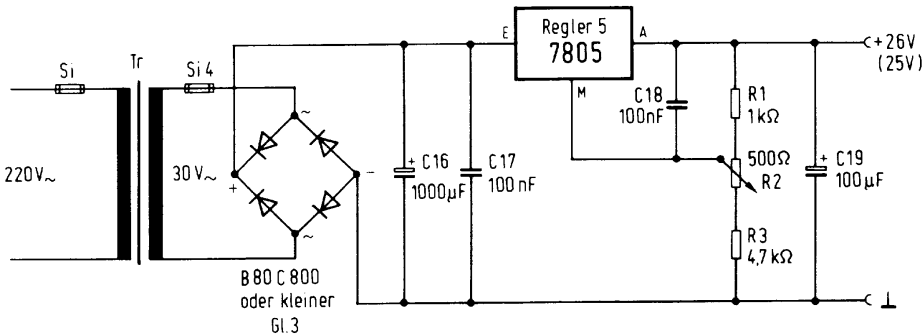


Abb. 1.3.1 Schaltung zur Erzeugung von 26V

mit 5V-Regler verwendet. Der Regler stellt eine Spannung von 5V zwischen seinem Masse-Anschluß und dem Ausgangspin her. Wird nun die Masse nicht an 0V gelegt, sondern an 21V, so erscheint am Ausgang eine Spannung von 26V. Dieser erhöhte Massepegel läßt sich mit den Widerständen R1, R2, R3 einstellen, wobei R2 ein Trimmer ist. Mit Hilfe eines Meßinstrumentes muß die Ausgangsspannung gemessen und der Trimmer so eingestellt werden, daß am Ausgang 26V liegen.

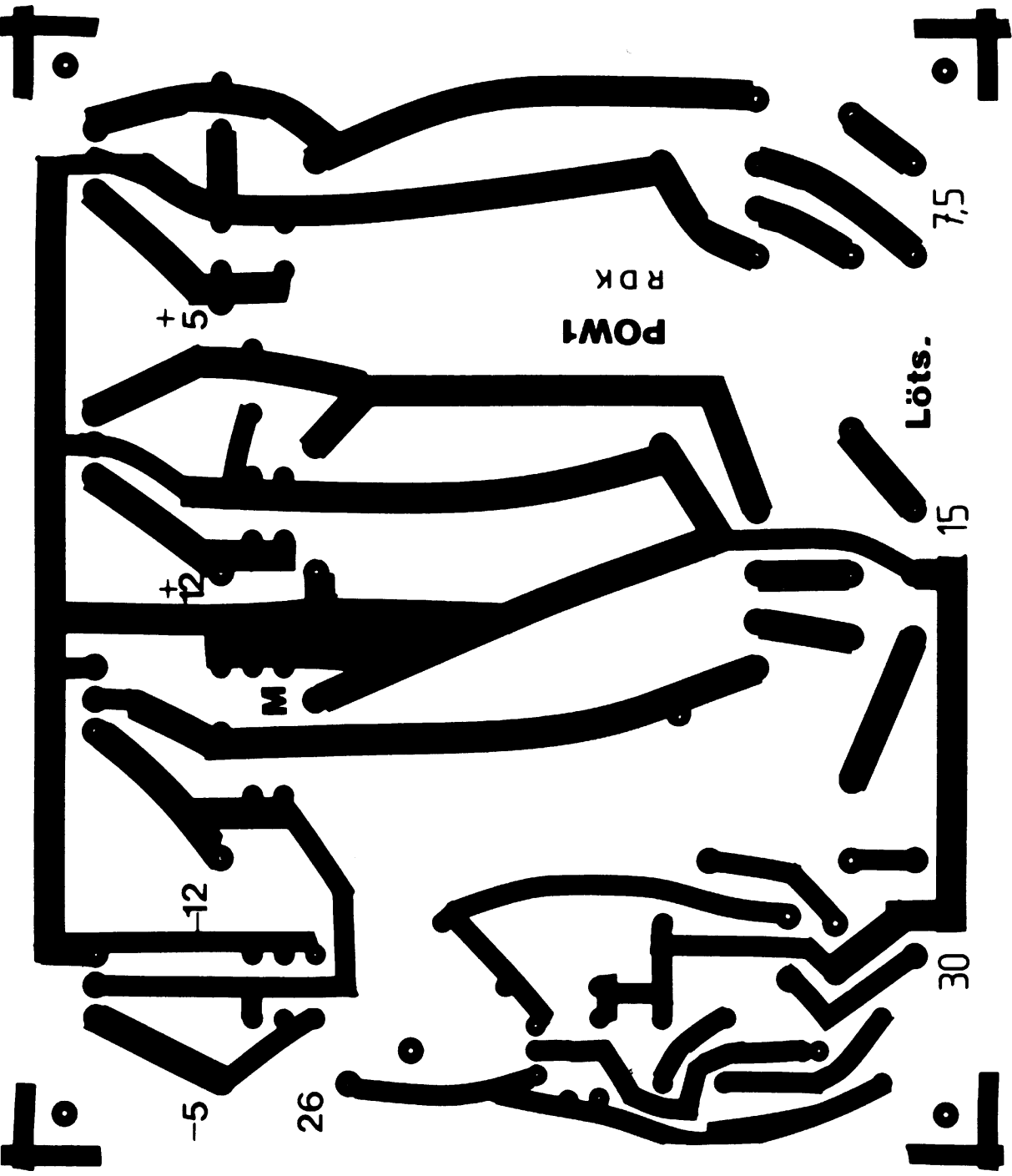
1.4 Praktischer Aufbau

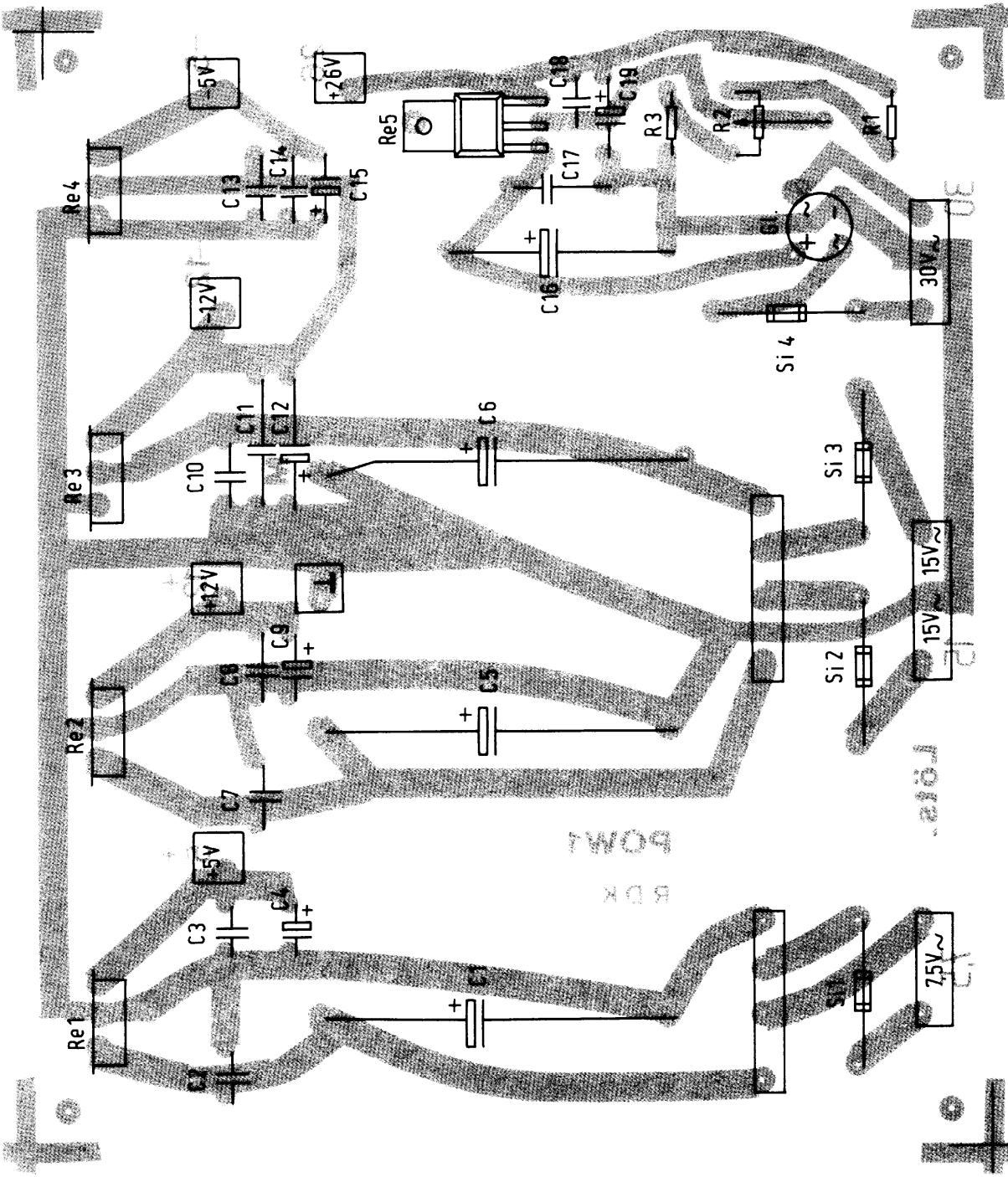
Um dem Leser den Nachbau zu erleichtern, wurde eine Platine entworfen. *Abb. 1.4.1* zeigt das Layout der Platine. *Abb. 1.4.2* zeigt den Bestückungsplan. Beim Einbau der Teile ist besonders auf die Polung der Elkos und der Spannungsregler zu achten.

1.5 TEST

Der Test einer Schaltung wird in allen Kapiteln eine sehr große Rolle spielen. Daher ist auch bei dieser einfachen Schaltung ein entsprechender Abschnitt zu finden.

Um eine Schaltung gut testen zu können, ist es am einfachsten, sie Stück für Stück aufzubauen. Nach jedem Schritt kann dann geprüft werden, ob der letzte Schritt richtig war. Bei unserem Netzteil bedeutet dies, daß zunächst die Stromversorgung der Trafos mit 220V hergestellt wird. Die Versorgung muß dabei über eine Sicherung geführt werden und ggf. über einen zentralen Netzschalter. Dabei sind unbedingt die VDE-Bestimmungen einzuhalten. Ist dieser Schritt ausgeführt, wird eingeschaltet und werden die verschiedenen Ausgangsspannungen der Trafos mit einem Meßinstrument kontrolliert, wobei die einzelnen Trafospannungen etwas höher als angegeben liegen dürfen. Dann werden als nächstes die Brückengleichrichter und Sicherungen für die Sekundärwicklungen eingebaut. Danach erfolgt wieder ein Einschalttest und anschließend werden die Siebkondensatoren C1, C5, C6 und C16 eingebaut. Danach muß wieder getestet werden. Vor dem Einbau der Spannungsregler sollte man sich mit Hilfe der Pin-Belegung (siehe Anhang) versichern, daß diese auch richtig eingebaut wrden. Nach dem Einbau der Regler und sonstigen Kleinteile er-


POW1
R D K
Löts.
+5
+12
M
-12
-5
26
15
30
7,5


Re1
Re2
Re3
Re4
Re5
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
R1
R2
R3
Si1
Si2
Si3
Si4
Si
-5V
+12V
-12V
+26V
7,5V~
15V~
15V~
30V~

folgt der Endtest. Es werden nach dem Einschalten die verschiedenen Einzelspannungen an den Ausgängen gemessen. Zur Probe sollte mit Hilfe eines Widerstandes einmal die Belastung simuliert werden, um sicherzugehen, daß das Netzteil auch unter Last funktioniert. Oszilloskopbesitzer sollen auch einmal die Ausgangsspannungen begutachten, ob sich Schwingungen oder Brumm zeigen.

1.6 Fragen zum Thema Stromversorgung

1. Wozu benötigt man Siebelkos?
2. Wozu dienen die Blockkondensatoren (hier 100 nF), die direkt an dem Spannungsregler liegen?

2 Kurze Einführung in die Digitaltechnik

Hier sollen noch einmal kurz die Grundlagen der Digitaltechnik besprochen werden. Alle Leser, die sich darin schon ganz gut auskennen, sollten dieses Kapitel trotzdem überfliegen, um die Begriffsweise, die in diesem Buch verwendet wird, kennenzulernen.

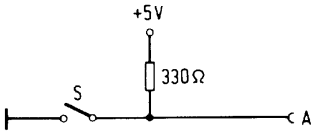
2.1 Definition der Signalpegel

Bei unseren Schaltungen arbeiten wir im allgemeinen mit TTL-Signalen. Das bedeutet 0V für eine logische 0 und 5V für eine logische 1. Die meisten TTL-Gatter liefern aber nicht genau diese Werte, sondern einen Pegel von 2.4 bis 5V für eine logische 1 (was auch als High-Pegel bezeichnet wird) und 0 bis 0.7V für eine logische 0 (Low-Pegel genannt). Die Eingänge der TTL-Gatter akzeptieren ebenfalls diesen Bereich, so daß die Signalpegel auch wieder richtig verstanden werden.

Bei manchen integrierten Schaltungen kann es aber sein, daß sie einen höheren Pegel als 2.4V für ein 1-Signal benötigen. Dann findet man bei den Eingängen meist einen sogenannten PULL-UP-Widerstand nach +5V. Er hat die Aufgabe den Pegel etwas nach 5V zu „ziehen". Der Widerstandswert liegt i. a. zwischen 1 kΩ und 330 Ω, je nach Art der Eingangsstufe. Warum ist dies überhaupt möglich? Die TTL-Ausgänge besitzen in der 5V-Leitung der Ausgangsstufe einen internen Widerstand, hingegen bei der 0V-Zuführung der Ausgangsstufe nicht. Wird extern ein Pull-Up-Widerstand angebaut, so steigt die Spannung bei einem 1-Signal an, bleibt bei einem 0-Signal aber fast unverändert auf dem Wert, den sie ohne Pull-Up-Widerstand hätte.

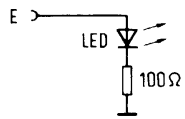
Zur Ansteuerung von TTL-Gattern ist in *Abb. 2.1.1* eine kleine Schaltung gezeigt, die die richtigen Signalpegel erzeugt. Bei geschlossenem Schalter liegt ein 0-Signal am Ausgang; bei geöffnetem ein 1-Signal. Ein solcher Schalter hat allerdings den Nachteil, daß er beim Schließen prellt: Beim Schließvorgang gibt er mehrere Male Kontakt und erzeugt damit am Ausgang eine Pulsfolge mit einer Dauer von bis zu 10 ms (je nach Schalter). Für manche TTL-Schaltungen, wie z. B. Zähler, ist sie daher nicht zu gebrauchen; wir werden später noch eine bessere Schaltung kennenlernen.

Um Signal-Pegel optisch darstellen zu können, benötigen wir auch eine Ausgabeeinheit. *Abb. 2.1.2* zeigt eine solche Schaltung mit einer Leuchtdiode. Liegt der Eingang auf einem 0-Signal, so ist die LED dunkel; bei einem 1-Signal leuchtet sie. Einen Nachteil besitzt diese Schaltung noch: wir wissen bereits, daß TTL-Schaltungen bei einem 1-Signal einen Widerstand in der Ausgangsleitung haben. Dies führt dazu, daß die LED nicht sehr hell leuchtet. Daher ist in *Abb. 2.1.3* eine andere Schaltung gezeigt. Diesmal wird die LED



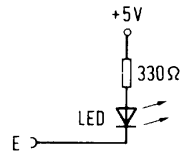
S	A
Schalter zu	0
Schalter auf	1

Abb. 2.1.1 Einfacher Signalgeber



E	LED
0	dunkel
1	leuchtet

Abb. 2.1.2 Anzeige eines Logikzustands



E	LED
0	leuchtet
1	dunkel

Abb. 2.1.3 Anzeige eines Logikzustands invers

immer dann leuchten, wenn ein 0-Signal am Eingang anliegt; sie ist dunkel bei einem 1-Signal. Doch halt, nun ist die Logik umgedreht. Dafür leuchtet die LED auch mit dem größeren Widerstand heller als im vorigen Beispiel. Diese Schaltung wird i. a. bei Mikrorechnerausgaben auf LEDs verwendet. Der Nachteil, daß die Anzeige verkehrt (invers) zu der vorherigen ist, spielt dabei keine Rolle, da sich dies per Programm leicht ausgleichen läßt, wie wir später noch sehen werden.

2.2 Logische Verknüpfungen

Das UND-Gatter

Abb. 2.2.1 zeigt das Schaltzeichen und die Wahrheitstabelle eines Und-Gatters. Am Ausgang erscheint genau dann und nur dann ein 1-Signal, wenn an beiden Eingängen (gleichzeitig) ein 1-Signal anliegt. Die boolsche Gleichung lautet dazu:

$$A = E1 \wedge E2$$

Als Beispiel für ein Und-Gatter sei hier der Schaltkreis 7408, der vier Gatter beinhaltet, genannt.

Das ODER-Gatter

Abb. 2.2.2 zeigt die Symbolik und Funktion eines Oder-Gatters. Am Ausgang eines Oder-Gatters ist immer dann ein 1-Signal, wenn an einem der beiden Eingänge oder an beiden Eingängen ein 1-Signal anliegt. Die Gleichung:

$$A = E1 \vee E1.$$

Ein Beispiel für ein Oder-Gatter ist der Baustein 7432, der vier Gatter enthält.

Das NICHT-Gatter

Abb. 2.2.3 zeigt das Nicht-Gatter. Liegt am Eingang ein 1-Signal, so erscheint am Ausgang ein 0-Signal und umgekehrt. Die Gleichung:

$$A = \neg E1$$

Ein Schaltkreis mit sechs Nicht-Gattern ist der 7404.

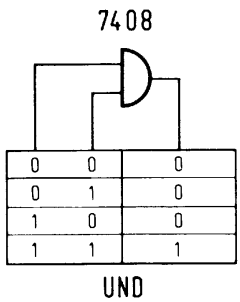


Abb. 2.2.1 Funktionsdiagramm eines Und-Gatters

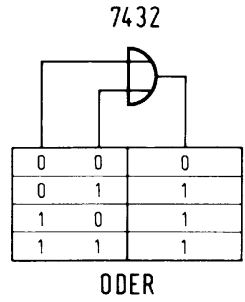


Abb. 2.2.2 Funktionsdiagramm eines Oder-Gatters

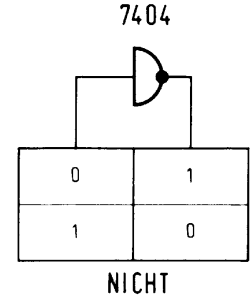


Abb. 2.2.3 Funktionsdiagramm eines Nicht-Gliedes

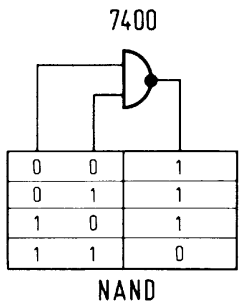


Abb. 2.2.4 Funktionsdiagramm eines Nand-Gatters

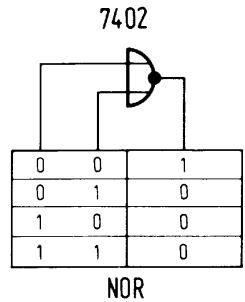


Abb. 2.2.5 Funktionsdiagramm eines Nor-Gatters

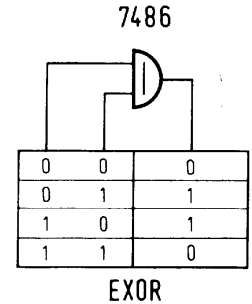


Abb. 2.2.6 Funktionsdiagramm eines Exclusiv-Oder-Gatters

Mit diesen drei Gatter-Typen läßt sich jede beliebige logische Schaltung realisieren. An dieser Stelle sei einmal das De-Morgansche Theoren erwähnt, welches aussagt, daß wenn ein Und-Gatter mit je einem Nicht-Gatter an den Eingängen und einem am Ausgang beschaltet wird, es die Eigenschaften eines Oder-Gatters zeigt und umgekehrt. Die Gleichung dazu lautet:

$$E1 \vee E2 = \neg(\neg E1 \wedge \neg E2)$$

und $$E1 \wedge E2 = \neg(\neg E1 \vee \neg E2)$$

Mit einer aufgebauten Schaltung läßt sich diese Tatsache am besten nachvollziehen, was ich jedem Leser anrate.

Das Nand-Gatter

Das Nand-Gatter wird wegen seiner Universalität sehr oft eingesetzt. *Abb. 2.2.4* zeigt das Schaltbild und die dazugehörige Wahrheitstabelle. Das Nand-Gatter ist einfach eine Und-Schaltung mit nachfolgendem Nicht-Gatter.
Die Gleichung dazu:

$$A = \neg(E1 \wedge E2)$$

Vier Nand-Gatter sind z. B. in dem Schaltkreis 7400 untergebracht.

Das Nor-Gatter

Hier wird einem Oder-Gatter ein Nicht-Gatter nachgeschaltet. *Abb. 2.2.5* zeigt das Gatter. Die Gleichung dazu:

$$A = \neg(E1 \vee E2)$$

Das Exclusiv-Oder-Gatter

Am Ausgang des Exclusiv-Oder-Gatters erscheint immer dann ein 1-Signal, wenn an dem einen oder an dem anderen Eingang ein 1-Signal anliegt, aber nicht, wenn an beiden Eingängen (zugleich) ein 1-Signal liegt, wie auch in *Abb. 2.2.6* gezeigt wird.

$$A = (E1 \wedge \neg E2) \vee (\neg E1 \wedge E2)$$

Als IC-Beispiel gibt es den Schaltkreis 7486, in dem sechs solcher Gatter untergebracht sind. Die Schaltung ist deshalb so praktisch, weil man sie auch als steuerbaren Inverter betrachten kann. Liegt an dem einen Eingang ein 0-Signal fest, so folgt der Ausgangspegel exakt dem Pegel am anderen Eingang. Wird der eine Eingang an ein 1-Signal gelegt, so verhält sich die restliche Schaltung wie ein Inverter.

2.3 Bus-Schaltkreise

Bisher wird jeder, der Digitaltechnik schon kannte, mühelos gefolgt sein. Nun kommen wir aber zu speziellen Bausteinen, die besonders für Mikrorechner-Schaltungen verwendet werden. Dort genügt eine reine Logik-Funktion nicht, sondern es gibt auch Funktionen, die darüber hinaus gehen. So ist das mit den sogenannten Bus-Treibern. Mit Standard-Logikschaltungen können wir beliebige Schaltungsnetze aufbauen, aber bei Mikrorechnern gibt es auch noch eine andere Struktur, die sogenannte Bus-Struktur. Dazu betrachten wir zunächst die Ausgangsstufe eines Standard-Bausteins. *Abb. 2.3.1* zeigt das Innenleben eines Nand-Gatters 7400. Bei der Ausgangsstufe ist immer einer der beiden Transistoren T3 oder T4 leitend. Damit liegt am Ausgang der Signalpegel 1 oder 0 an.

Wird der Transistor T4 weggelassen, so ergibt sich ein OPEN-COLLECTOR-Ausgang. *Abb. 2.3.2* zeigt die Innenschaltung des ICs 7401, das vier solche Open-Collector-Nand-Gatter beinhaltet. Wenn an den beiden Eingängen E1 und E2 gleichzeitig ein 1-Signal anliegt, so ist die Nand-Funktion erfüllt und am Ausgang liegt ein 0-Signal. Liegt aber eine andere Eingangsbelegung vor, so leitet T3 nicht und der Ausgang ist offen. Um damit weiterarbeiten zu können, wird ein Pull-Up-Widerstand benötigt, der auch bei offenem Zustand den Signal-Pegel 1 garantiert. Wozu soll das nun gut sein? Mit diesen Open-Collector-Schaltungen lassen sich neue Schaltungsstrukturen bilden.

Dazu betrachten wir *Abb. 2.3.3*, dort sind drei Ausgänge der Open-Collector-Inverter 7406 zusammengeschaltet. Mit normalen Gattern geht das nicht, da sich zwischen zwei Ausgängen ein Kurzschluß bilden kann. Wir wollen versuchen, die Logikfunktion dazu aufzustellen. Liegt E1 auf einem 1-Pegel, so ist der Ausgang des Gatters N1 auf 0. Wegen der Open-Collector-Eigenschaften gilt das auch für den Ausgang A der gesamten Schaltung.

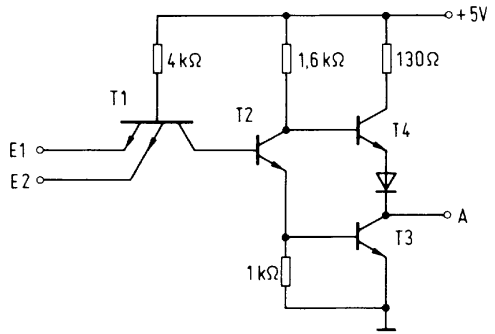


Abb. 2.3.1 Innenleben eines TTL-Gatters

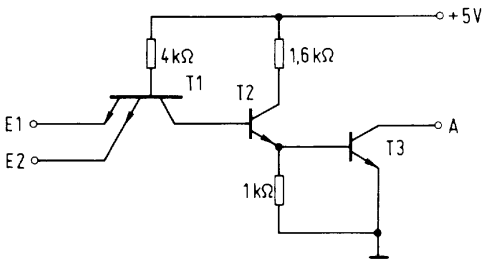


Abb. 2.3.2 Innenleben eines Gatters mit Open-Collector

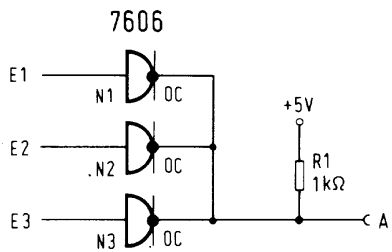


Abb. 2.3.3 Wired-Or-Schaltung

Damit gilt: Falls einer der Eingänge auf einem 1-Signal liegt, so liegt der Ausgang A auf einem 0-Signal. Ein 1-Signal am Ausgang ist nur dann vorhanden, wenn an allen Eingängen ein 0-Signal liegt, dann wird der Pull-up-Widerstand R1 am Ausgang das 1-Signal erzeugen. Übrigens wird durch den Querstrich am Ausgang gezeigt, daß es sich um eine Open-Collector-Schaltung handelt. In der einschlägigen Literatur ist das leider nicht einheitlich aber, in diesem Buch soll nur dieses Zeichen verwendet werden.

Die Schaltung liefert also die NOR-Funktion. Der Vorteil einer solchen Schaltungsart liegt darin, daß die einzelnen Gatter N1, N2 oder N3 auch räumlich voneinander getrennt liegen können und nur durch die Ausgangsleitung miteinander verbunden sind. Bei Mikrorechnern gibt es die Aufgabe, Information (in Form von 0- und 1-Signalen über Leitungen von einem Teil der Schaltung zur anderen und zurück zu übertragen, wobei die Information von verschiedenen Untereinheiten kommen kann. Für diese Aufgabe eignet sich eine Bus-Struktur. *Abb. 2.3.4* zeigt ein Blockschema. Von den Quellen 1, 2 und 3 sollen Daten zum Ziel übertragen werden. Wie läßt sich dies realisieren? Natürlich mit unseren Open-Collector-Bausteinen.

Abb. 2.3.5 zeigt die Schaltung. Dabei besteht der Bus hier nur aus einer Leitung. In Wirklichkeit besteht ein Bus jedoch normalerweise aus mehreren Leitungen. Beliebige Logikpegel kommen von den Quellen 1, 2 und 3 und sollen an das Ziel durchgeschaltet werden. Damit nicht alle gleichzeitig Signale auf den Bus legen können, muß eine Freigabe für die einzelnen Quellen durchgeführt werden. Dies geschieht mit den Eingängen Frei 1, 2 und 3. Von diesen Eingängen darf immer nur einer ein 1-Signal führen, die anderen müssen

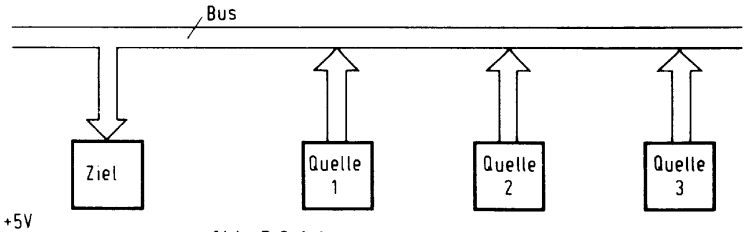


Abb. 2.3.4 Aufbau eines Bus-Systems

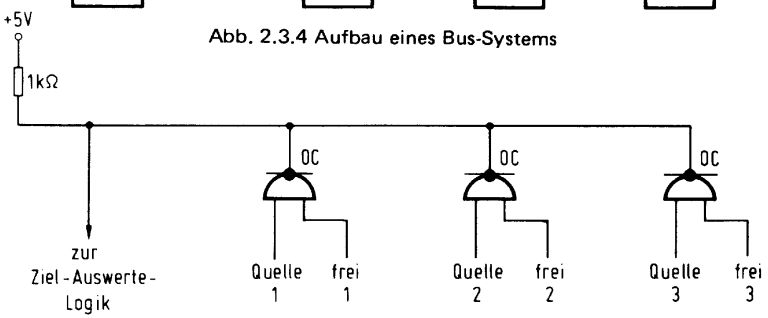


Abb. 2.3.5 Realisierung einer Bus-Leitung

und 3. Von diesen Eingängen darf immer nur einer ein 1-Signal führen, die anderen müssen ein 0-Signal haben. Die Auswahl geschieht mit einer Adresslogik, zu der aber weitere Bus-Leitungen benötigt werden, wie später noch gezeigt wird. Das Prinzip ist aber durch diese Schaltung gezeigt.

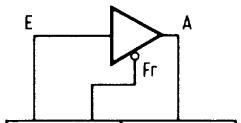
Noch sehr unschön ist, daß ein Widerstand nach +5V geschaltet werden muß; besser wäre es, wenn die Signalpegel von den Gattern schon richtig erzeugt würden. Auch dies ist möglich, dazu gibt es die sogenannten TRI-State-Ausgänge. In der Ausgangsstufe sind dann wieder beide Transistoren vorhanden, doch ist es mit einem zusätzlichen Eingang möglich, beide Transistoren stromlos zu schalten und dabei wieder den gewünschten offenen Zustand zu erreichen. *Abb. 2.3.6* zeigt die Funktionstabelle und das Schaltzeichen des Bus-Treibers 74367. Liegt der Freigabe-Eingang auf 0, so wird der Signalpegel vom Eingang E auf den Ausgang A durchgeschaltet. Liegt der Freigabe Eingang Fr auf 1, so ist der Ausgang im offenen Zustand unabhängig vom Eingang E.

Bi-Direktionale-Bustreiber

Nun kann man sich leicht vorstellen, daß zwei solche Bus-Treiber auch gegeneinander geschaltet werden können, also der Ausgang des einen Gatters mit dem Eingang des anderen und umgekehrt. Werden die Freigabeeingänge abwechselnd geschaltet, so wird einmal von der einen Seite zur anderen und im anderen Fall umgekehrt übertragen.

Abb. 2.3.7 zeigt einen solchen Schaltkreis (74245). Dieser besitzt die beiden Ein-/Ausgänge A und B und zwei Kontroll-Eingänge Dir und G-Quer. Liegt an G-Quer ein 1-Signal, so sind beide Treiberstufen im offenen Zustand und damit die gesamte Schaltung. Liegt G-Quer auf einem 0-Pegel, so ist immer eine der beiden Treiberstufen freigegeben. Liegt Dir auf 0, so wird von B nach A übertragen; und liegt Dir auf 1, so wird von A nach B über-

74244 / 74367

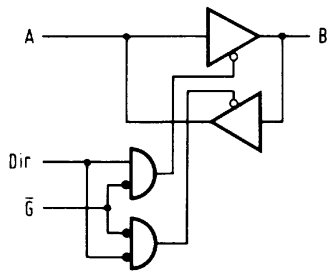


0	0	0
1.	0	1
0	1	offen
1	1	offen

Bus-Treiber

Abb. 2.3.6 Tristate-Treiber

74245



$\overline{G}$	Dir	Funktion
0	0	B → A
0	1	A → B
1	0	offen
1	1	offen

Abb. 2.3.7 Bidirektionaler Bustreiber

tragen. Diese Bus-Schaltung wird verwendet, um an einen Datenbus Signale in beiden Übertragungsrichtungen anlegen zu können.

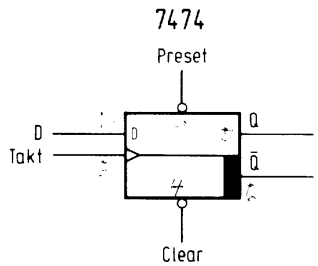
In dem Schaltkreis 74245 (74LS245) sind acht solche Bi-Direktionalen-Bustreiber und ein gemeinsamer Dir und G-Quer-Anschluß untergebracht. Damit kann auf einen 8-Bit breiten Datenbus zugegriffen werden.

2.4 Flip-Flop Schaltungen

Die bisherigen Gatterschaltungen waren nicht in der Lage, Signalzustände zu speichern. Dies soll sich nun ändern. Es sollen hier aber nur einige von vielen Möglichkeiten dargestellt werden.

D-Flip-Flop

Abb. 2.4.1 zeigt das Schaltbild und die Wahrheitstabelle eines D-Flip-Flops. Es hat folgende Eigenschaft: der Signalpegel, der während der ansteigenden Signalflanke des Taktsignals an dem D-Eingang anlag, wird in das Flip-Flop übernommen. Als ansteigende Signalflanke wird dabei ein Wechsel des Pegels von 0 auf 1 bezeichnet. Man nennt den Übergang auch positive Signalflanke. Das Flip-Flop besitzt noch zwei zusätzliche Eingänge, einen Setz- und einen Rücksetz-Eingang. Damit ist es möglich, durch einen kurzen Puls auf 0 das Flip-Flop in eine bestimmte Lage zu bringen. Ein Puls auf 0 am PRESET-Eingang setzt den Ausgang Q auf 1 und Q-Quer auf 0. Ein Puls auf 0 am Eingang CLEAR bewirkt das Gegenteil. Die beiden Eingänge liegen während des Takt-Betriebes auf 1. Die Besonderheit eines solchen Flip-Flops liegt darin, nur auf den Übergang am Takt zu reagieren, nicht etwa auf einen statischen Pegel. Dann erst ist es möglich, solche Elemente zu verketten. *Abb. 2.4.2* zeigt ein Beispiel. Der Ausgang des Flip-Flops 1 ist mit dem D-Eingang des nachfolgenden



Preset	Clear	Takt	D	Q	$\bar{Q}$	
0	1	x	x	1	0	Setzen
1	0	x	x	0	1	Rücksetzen
1	1	↓	1	1	0	1-einschreiben
1	1	↓	0	0	1	0-einschreiben
1	1	0	x	Q_0	$\bar{Q}_0$	Speichern
0	0	x	x	1*	1*	Zustand instabil

Abb. 2.4.1 Funktion eines Flip-Flops

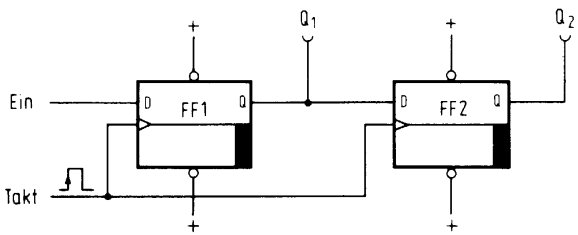
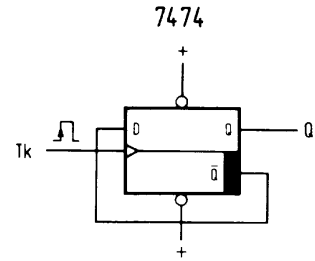


Abb. 2.4.2 Schieberegister-Schaltung

Flip-Flops verbunden. Dadurch ergibt sich ein Schieberegister. Dies hat folgende Eigenschaften: Ein Signalpegel am Eingang des Schieberegisters wird bei jeder ansteigenden Taktflanke von einem zum nächsten Flip-Flop verschoben. Liegt beispielsweise am Eingang der logische Pegel 1, so erscheint dieser Pegel nach dem ersten Taktpuls am Ausgang Q1 und nach dem zweiten Taktpuls am Ausgang Q2. Entsprechend kann der Eingangspegel nach jeder Übernahme wechseln; er wird dann wieder bei der nächsten positiven Signalflanke nach rechts verschoben. An Q1 und Q2 liegen dann die Pegel, die zuvor einmal nacheinander am Eingang gelegen haben, parallel an. Damit wurde eine sogenannte Serien-Parallel-Wandlung erreicht. Diese spielt bei den Mikrorechnern eine große Rolle, wie wir später noch sehen werden.

Eine andere Grundschaltung läßt sich bei der Verwendung eines D-Flip-Flops ebenfalls erreichen. *Abb. 2.4.3* zeigt das Schaltbild. Diesmal wird der Ausgang Q-Quer an den Eingang D desselben Flip-Flops geschaltet. Es wird dadurch erreicht, daß sich bei jeder positiven Flanke an dem Takt-Eingang der Zustand des Ausgangssignals ändert. War es zuvor auf 1, so wird es danach auf 0 liegen und umgekehrt. Die Schaltung wirkt als Zähler. Sie ist ebenfalls ein Frequenzteiler. Liegt am Takt-Eingang eine Frequenz von 4 MHz, so wird am Ausgang eine Frequenz von 2 MHz erscheinen.

Abb. 2.4.3 Zähler



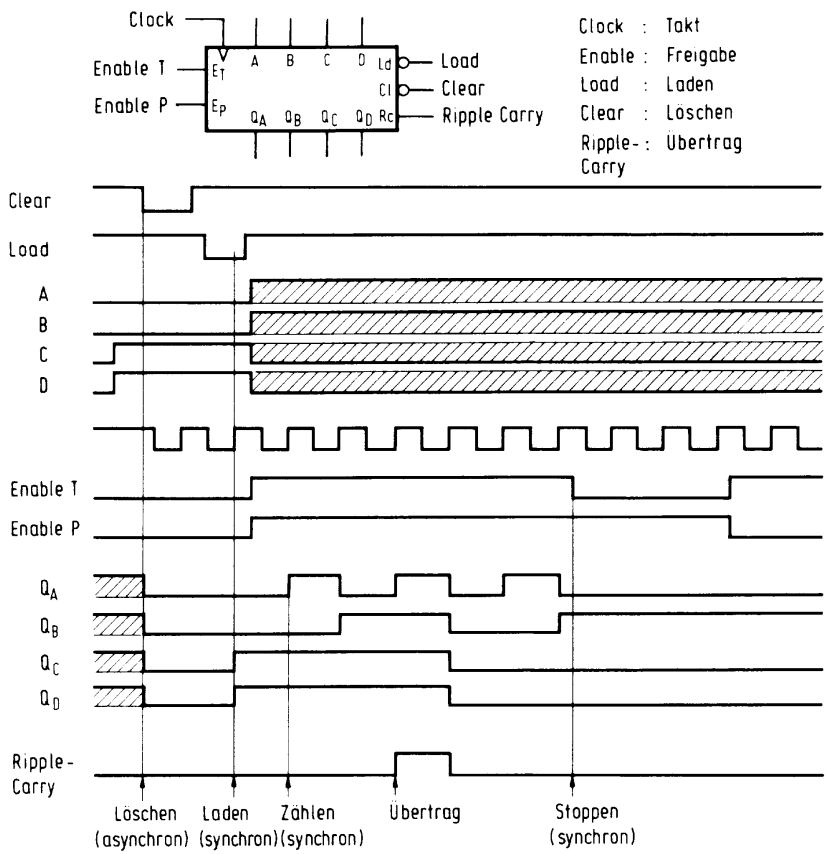


Abb. 2.4.4 Ein integrierter Vierfach-Zähler

Solche Zähler gibt es aber auch schon fertig integriert mit mehreren Stufen in einem Gehäuse. *Abb. 2.4.4* zeigt ein Beispiel. Der Schaltkreis 74161 enthält vier Zählerstufen, die ähnlich wie im vorherigen Beispiel aufgebaut sind, jedoch eine Reihe weiterer Eigenschaften besitzen. Die Zählerstufe kann von 0 bis 15 zählen, ferner besitzt sie vier Eingänge, über die sich der Zählerstand vorbelegen läßt. Dazu wird ein Puls auf 0 an den Load-Eingang gelegt. Mit dem Eingang Clock kann der Zählerstand pro positive Signalflanke um

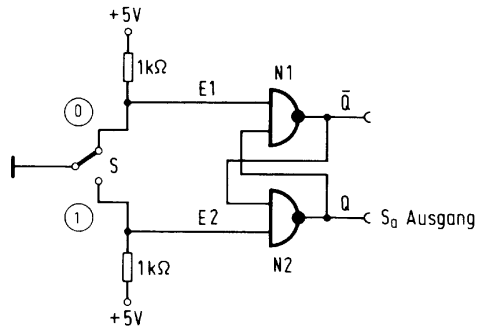


Abb. 2.4.5 RS-Flip-Flop

eins erhöht werden. Dazu müssen die Freigabe-Eingänge „Enable T" und „Enable P" beide auf 1 liegen. Die Eingänge dienen dazu, mehrere Zählerschaltungen hintereinander anzuordnen. Der Ausgang „Ripple carry" liefert ein Übertragssignal, sobald der Zählerstand 15 erreicht ist und beim nächsten Takt der Zähler wieder den Stand 0 annehmen wird. Mit dem Eingang Clear kann der Zähler auf 0 gesetzt werden. Dies ist wichtig, um z. B. nach dem Stromeinschalten einen definierten Anfangszustand zu erreichen. Es könnte auch durch Laden einer 0 mit Hilfe von Load erreicht werden. Um diesen Baustein besser kennenzulernen, werden Experimente damit sehr empfohlen.

RS-Flip-Flop

Eine wesentlich einfachere Flip-Flop-Schaltung zeigt *Abb. 2.4.5*. Es handelt sich um ein RS-Flip-Flop, eine Abkürzung von Rücksetz-Setz-Flip-Flop. Mit einem solchen Flip-Flop kann zum Beispiel ein Schalter entprellt werden: Wir denken uns zunächst einmal den Schalter weg. Dann soll z. B. der Ausgang Q auf 0 liegen. Damit wird Q-Quer auf 1 liegen und deshalb Q wieder auf 0. Umgekehrt fangen wir mit Q-Quer auf 0 an, so muß Q auf 1 liegen und deshalb Q-Quer auf 0, da die beiden anderen Eingänge des Flip-Flops auf 1 liegen. Das Flip-Flop hat also zwei stabile Zustände. Wird nun einer der beiden zusätzlichen Eingänge auf 0 gelegt, so kann das Flip-Flop in den anderen Zustand kippen. Liegt Q auf dem Wert 0 und wird 0 an den Eingang E2 gelegt, so liegt anschließend der Wert 1 an diesem Ausgang. Mit E1 kann durch einen Puls auf 0 umgekehrt wieder 0 am Ausgang Q geschaltet werden. Der Schalter S tut nun genau das. Es wird abwechselnd E1 oder E2 auf 0 gelegt und damit folgt der Ausgang Q genau der Schalterstellung (Q-Quer folgt invers dazu). Das Prellen des Schalters wird unterdrückt, da der Schalter, ein Umschalter eigentlich drei Signalzustände besitzt. Einmal Kontakt oben, kein Kontakt und Kontakt unten. Gibt er nun mehrere Male Kontakt, so kann er dies nur zwischen zwei von diesen Zuständen tun – als Prellen zwischen oberem Kontakt und dem offenen Zustand und Prellen zwischen dem unteren Kontakt und offen. Das Flip-Flop wechselt den Zustand aber nur bei einem kompletten Wechsel des Schalters von der oberen auf den unteren Kontakt oder umgekehrt. Dadurch wird das Prellen vollständig unterdrückt.

2.5 Fragen zur Digitaltechnik

1. Nur aus Nand-Gattern ist ein Oder-Gatter aufzubauen.
2. Wie läßt sich mit dem Flip-Flop 7474 ein Schalter entprellen?

3 Messen an Digitalschaltungen

Schon einige Male wurde fast unmerklich ein Meßhilfsmittel verwendet: Einmal beim Testen der Stromversorgung und dann auch im Digitaltechnik-Teil, als es galt, den logischen Zustand eines Signals mit Hilfe einer LED festzustellen.

Wir werden in dem nachfolgenden Kapitel andauernd messen müssen und beschäftigen uns daher etwas genauer mit diesem Thema.

3.1 Messen mit einem Multimeter

Eine Grundvoraussetzung für alle Arbeiten mit Mikrorechnern ist das Vorhandensein eines einfachen Multimeters. Dabei kommt es überhaupt nicht auf die Genauigkeit an, sondern nur auf einige wenige Punkte. Einmal wird ein 5V Bereich benötigt, den wohl jedes Multimeter besitzt. Dann benötigt man einen Widerstandsmeßbereich, um z. B. Durchgänge in Schaltungen prüfen zu können. Schön wäre auch ein Strommeßbereich bis 2A, um die Stromaufnahme einer Mikrorechnerschaltung bestimmen zu können.

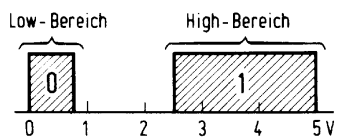


Abb. 3.1.1 High und Low-Pegel Bereiche

Mit dem Meßbereich von 5V kann auch grob die Funktion eines Gatters getestet werden. Dazu zeigt *Abb. 3.1.1* den Pegelbereich eines TTL-Gatters. Beim Zustand 0 liegt die Ausgangsspannung zwischen 0 und 0.7V und bei einem 1-Signal zwischen 2.4 und 5V. Eingänge liegen, wenn sie unbeschaltet sind, meist um 1V herum. So kann zum Beispiel bestimmt werden, ob ein Eingang nicht beschaltet wurde. Dabei gilt fast immer, daß ein offener Eingang wie mit „1" beschaltet reagiert. Doch sollte man sich davor hüten, Eingänge von TTL-Gattern unbeschaltet zu lassen, da die Ergebnisse gerade bei takteten Systemen nicht immer zuverlässig sind. Unbeschaltete Eingänge sind sehr störanfällig und fangen gerne fremde Signale auf.

3.2 Messen mit dem Oszilloskope

Jeder, der sich ernsthaft mit Mikrorechnern beschäftigen will, wird um die Anschaffung eines Oszilloskopes nicht herumkommen. Dabei muß es nicht das teuerste sein, denn wir müssen fast immer nur Pegel von 0V und 5V messen, doch gibt es zwei wichtige Punkte.

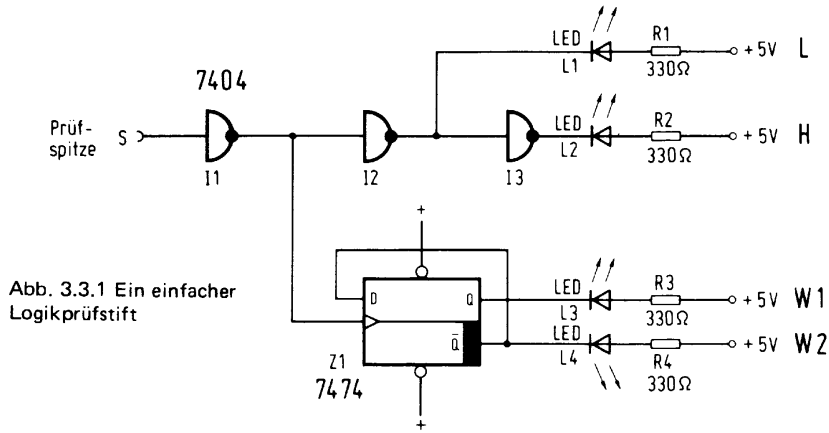


Abb. 3.3.1 Ein einfacher Logikprüfstift

Einmal muß es einen externen Triggereingang besitzen, der auch gut funktioniert, um Vergleichsmessung durchführen zu können; zum andern sollten Frequenzen bis 10 MHz noch dargestellt werden können, je höher je besser, da Mikrorechner mit sehr hohen Taktfrequenzen arbeiten. Spannungsbereiche sind dagegen nicht so sehr wichtig. Besonders elegante Messungen ermöglicht natürlich ein Zweikanal-Scop, doch ist dies nicht unbedingt erforderlich, da auch der Triggereingang ähnliches leistet.

3.3 Aufbau eines TTL-Prüfstiftes

Besonders praktisch ist die Benutzung eines Logikprüfstiftes, der hier im folgenden zum Nachbau vorgestellt wird. Ein Logikprüfstift soll zum einen die beiden Pegel 0 und 1 unterscheiden können, zum anderen muß auch erkannt werden, wenn eine Pulsfolge anliegt. Genau dies kann unser Logik-Stift.

Abb. 3.3.1 zeigt die dazugehörige Schaltung. Der Prüfstift besitzt vier Leuchtdioden als Ausgang, die die Unterscheidung von verschiedenen Logik-Zuständen ermöglichen. Das Signal S wird an den Eingang des Prüfstiftes enkoppelt und über den Inverter I1 an das Innenleben geführt. Es gelangt an das Gatter I2 und den Zähler Z1. Von I2 aus gelangt das Signal einmal direkt an die Leuchtdiode L1 und über einen weiteren Inverter an die Leuchtdiode L2. Es leuchtet bei einem statischen Eingangssignal immer nur eine der beiden Leuchtdioden. Ein Wechsel des Eingangssignals von 0 auf 1 läßt das Flip-Flop Z1 umkippen und am Ausgang dieses Flip-Flops ergibt sich ein Signalwechsel. Leuchtete zuvor die Led L3, so wird dann L4 leichten und umgekehrt.

Abb. 3.3.2 zeigt die verschiedenen Zustände der Leds in Abhängigkeit vom Eingangssignal. Bei den Leds wird in drei Leuchtzustände unterteilt. Zum einen kann die Led hell leuchten, dann dunkel sein und zum dritten leuchtet sie weder hell noch gar nicht, also halbhell.

	L L1	H L2	W1 L3	W2 L4
S= 0-Signal	✱	○	✱	○
S= 0-Signal	✱	○	○	✱
S= 1-Signal	○	✱	✱	○
S= 1-Signal	○	✱	○	✱
S= ⎍⎍	⊛	⊛	⊛	⊛
S= ⎍⎍	✱	○	⊛	⊛
S= ⎍⎍	○	✱	⊛	⊛

Abb. 3.3.2 Analysetafel des Logikstiftes

✱ leuchtet hell
⊛ leuchtet halb hell
○ dunkel

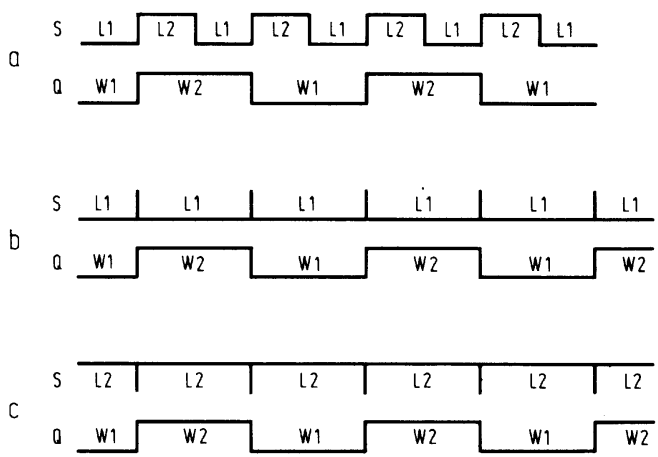


Abb. 3.3.3 Unterschiedliches Verhalten bei Signalen

Liegt ein statisches Eingangssignal vor, so zeigen die Leds L1 und L2 den Signalpegel an. Leuchtet L1, so liegt ein 0-Signal an, leuchtet dagegen L2, so liegt ein 1-Signal am Eingang (auch wenn dieser offen ist). Die Leds L3 und L4 spielen dabei keine Rolle. Es leuchtet aber nur eine von beiden Leds auf. Anders verhält sich das ganze bei einem anliegenden Takt. Ist dieser symmetrisch, so leuchten alle vier Leds auf. Bei einer Pulsfolge mit 1-Pulsen

leuchten L3 und L4 und diesmal ist L2 dunkel (oder fast dunkel). Umgekehrt bei einer 0-Pulsfolge ist L1 fast dunkel.

Abb. 3.3.3 zeigt das Impulsschema zu dieser Schaltung. Durch das Flip-Flop Z1 wird eine unsymmetrische Pulsfolge in eine symmetrische mit halber Frequenz geteilt; daher leuchten L3 und L4 gleich hell.

3.4 Fragen zum Thema Messen

1. Wie kann mit einem Triggereingang eines Oscilloskopes festgestellt werden, ob zwei Impulse nicht gleichzeitig kommen?
2. Kann eine Pulsfolge, in der paarweise zwei Pulse dicht aufeinanderfolgen, mit dem Prüfstift festgestellt werden?

4 Der Mikrorechner

Ohne Mikrorechner wäre der heutige Alltag kaum denkbar. In den folgenden Kapiteln wird systematisch in die Materie eingeführt. Dabei wird zunächst anhand einer selbst gebauten kleinen Ablaufsteuerung in die Prinzipien der Mikrorechner-Technik eingeführt, um dann mit einem konkreten Prozessor, dem Z80 fortzufahren.

4.1 Einführung in die Mikrorechner-Technik

Als es die ersten integrierten Schaltungen mit Logikfunktionen gab, wurde für jedes Schaltungsproblem eine logische Schaltung entworfen, die die Probleme mit Flip-Flops und Gatter löste. Beispiel: Der Bau eines Frequenzmeßgerätes. Die Eingangsstufe wurde mit Hilfe einer Zählerkette gebildet und der Zählerstand über verschiedene Multiplexer und Dekoder an die Anzeige geführt. Sollten außer der Frequenzmessung noch andere Messungen wie Periodendauermessung durchgeführt werden, so wurde weitere Logik nötig. Als die Integrationsdichte stieg, änderte sich zunächst nichts an diesem Prinzip. Die ganze Schaltung eines Frequenzmeßgerätes wurde dann z. B. als Kunden-Schaltkreis integriert und als ein IC verfügbar. Der Nachteil dieser Methode lag darin, daß die Kosten zur Herstellung eines Kunden-ICs sehr groß sind und es sich daher nur bei einer großen Stückzahl lohnt, einen Kundenschaltkreis zu bauen. Nun war man in der Lage, hohe Integrationsdichten zu beherrschen, konnte aber keinen universellen Schaltkreis auf den Markt bringen, der einer Vielzahl von Kunden nützen konnte. Das war der Punkt für die „Erfindung" des Mikroprozessors. Dieser Baustein kann wie bei Großrechnern programmiert werden. Es ist dann möglich, eine Standard-Schaltung in großer Stückzahl herzustellen und dennoch für jeden Kunden eine individuelle Problemlösung zu bieten. Der Kunde braucht „nur" ein Programm zu erstellen, das ebenfalls in einem Standard-Speicher-Baustein abgelegt wird und schon hat er eine individuelle Schaltungsfunktion.

Den Schritt zum Mikrorechner wollen wir im folgenden nachvollziehen.

Ein Mikrorechner muß Eingänge besitzen, um auf Signale zu reagieren und Ausgänge, um Aktionen weiterzuleiten. *Abb. 4.1.1* zeigt das Blockschema.

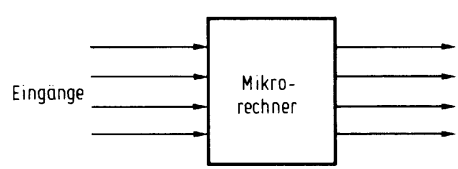


Abb. 4.1.1 Schema eines Mikrorechners

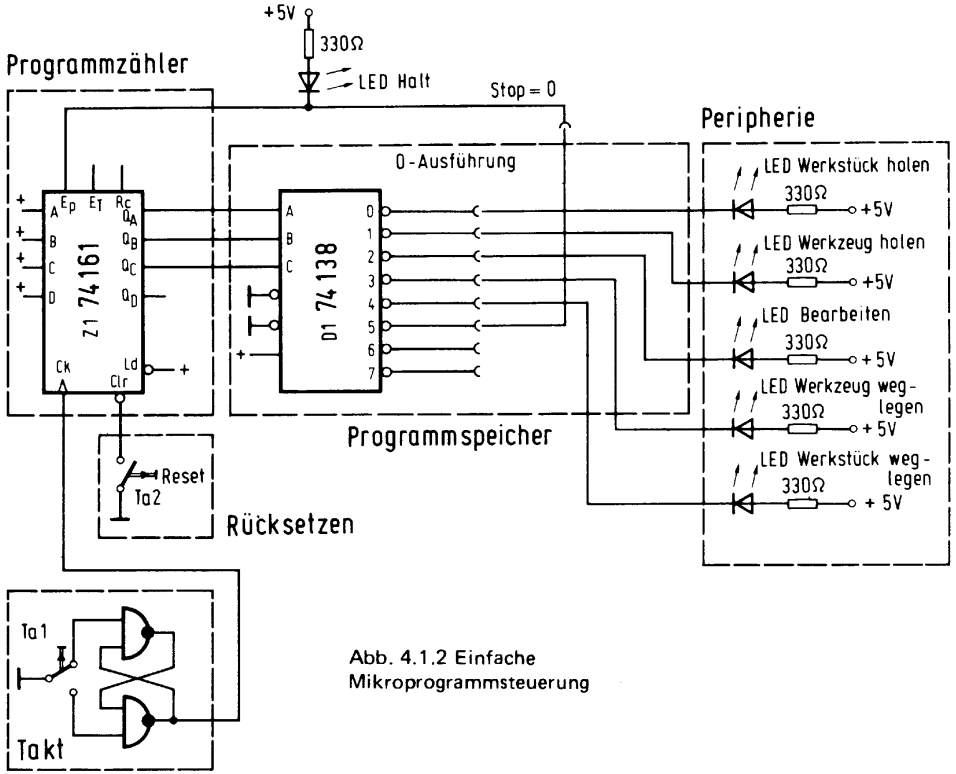


Abb. 4.1.2 Einfache Mikroprogrammsteuerung

Doch nun zur Praxis. Dazu sei eine kleine Aufgabe gestellt. Der Mikrorechner soll nacheinander folgende Aktionen ausführen:

1. Werkstück holen
2. Werkzeug holen
3. bearbeiten
4. Werkzeug weglegen
5. Werkstück weglegen

Dabei sollen die Aktionen durch ein logisches Signal eingeleitet werden; es sei dahingestellt, wie sie dann jeweils ausgeführt werden. Der Ablauf soll durch einen Tastendruck ausgeführt werden.

Abb. 4.1.2 zeigt die Lösung. Es sind einzelne Programmschritte auszuführen. Dazu dient der Programmzähler. Im Programmspeicher ist der Ablauf festgelegt, der bei unserem Beispiel natürlich ganz einfach ist, da er aus einer Abfolge von verschiedenen Aktionen besteht. Mit einem Taktgeber wird die Geschwindigkeit des Ablaufes festgelegt. In dem Peripherieteil werden die Aktionen ausgeführt, wobei hier jeweils eine LED für eine Aktion steht.

Der Takt ist nun periodisch mit dem Taster Ta1 zu geben. Wird die Taste Ta2, also die Reset-Taste betätigt, so beginnt der Ablauf. Der Zähler Z1 beginnt von 0 an aufwärts zu

zählen. Der aktuelle Zählerstand wird an einen Dekoder D1 geführt, der zu jedem Zählerstand eine dazugehörige Signal-Leitung auf den Pegel 0 bringt. So wird zuerst der Ausgang 0 auf 0 gehen, dann der Ausgang 1 usw. Die Leds reagieren jeweils auf diesen 0-Pegel und leuchten dabei auf. Wird der Zählerstand 5 erreicht, so wird der Zähler Z1 über den Eingang Ep blockiert und zählt nicht mehr weiter. Der Ablauf ist beendet. Die Led HALT zeigt diesen Zustand an. Ein neuer Ablauf beginnt dann wieder, sobald die Schaltung erneut durch Betätigen des Reset-Tasters in Gang gebracht wird.

Zwei Nachteile hat diese Schaltung noch: Einmal ist nicht gesichert, was unmittelbar nach dem Stromeinschalten passiert. Der Zählerstand ist dann undefiniert und kann einen beliebigen Wert annehmen und so z. B. schon eine Aktion auslösen. Dieses Problem bleibt auch bei realen Mikrorechnern bestehen und wird dadurch gelöst, daß mit einer speziellen Schaltung unmittelbar nach dem Stromeinschalten, ein Rücksetzvorgang ausgelöst wird. Der andere Nachteil besteht darin, daß bei dem Betätigen des Reset-Tasters die erste Aktion (Werkstück holen) sofort durchgeführt wird und das Signal so lange ansteht, wie die Taste betätigt wird. Um diesen Nachteil auszugleichen, werden einfach alle Eingänge der Peripherie an den nächsten Ausgang des Dekoders D1 geschaltet. Die Aktion „Werkstück einlegen" wird dann erst im Schritt 1 und nicht mehr im Schritt 0 ausgeführt.

Die hier gezeigte Programmsteuerung ist aber noch nicht universell genug, wie man leicht einsehen wird. Dazu eine neue Aufgabe: Es soll eine Verkehrsampel angesteuert werden; dabei sollen folgende Sequenzen dauernd ausgegeben werden:

1. ROT
2. ROT-GELB
3. GRÜN
4. GELB

Abb. 4.1.3 zeigt die Programmtabelle. Die Programmsteuerung besitzt also drei Ausgänge; Rot, Gelb und Grün.

Abb. 4.1.4 zeigt die Schaltung. Es gibt auch hier wieder die Teile Programmzähler und Programmspeicher. Der Zähler wird von einem Taktgeber mit der Taste Ta1 gespeist. Der Zähler besitzt diesmal keinen Reset, da er bei dieser Aufgabe permanent arbeiten soll. Zwei Ausgänge des Zählers werden an den Dekoder geführt, damit können vier Schritte ausgeführt werden. Die Ausgänge des Dekoders führen an eine Diodenmatrix, mit der sich der Programmablauf bestimmen läßt. Die Ausgänge der Dioden-Matrix führen an die Peripherie-Einheit. Nun werden einfach gemäß unserer Ablauftabelle die Dioden zu dem jeweiligen Schritt gehörig den Leds zugeordnet.

Noch etwas universeller als die vorige Schaltung ist die Lösung in *Abb. 4.1.5*. Dort sind wieder alle drei Ausgänge an den Dekoder angeschlossen. Nun würde der Zähler aber einfach weiterzählen, nachdem er Schritt 3 (also GELB) ausgegeben hat. Dazu wird nun mit einer weiteren Diode eine Verbindung zum Eingang LOAD des Zählers Z1 hergestellt. Beim nächsten Takt wird dann der an den Eingängen A bis D anstehende Wert in den Zähler übernommen. Damit wird ein sogenannter SPRUNG-BEFEHL ausgeführt. Der Zähler zählt also auch nur von 0 bis 3. Sollen die Programmschritte erweitert werden, so ist es nun einfach, den Sprung-Befehl weiter nach hinten zu versetzen; dadurch werden zusätzliche

Abb. 4.1.3
Aufgabe Verkehrsampel

Programm-schritt	
0	Rot
1	Rot-Gelb
2	Grün
3	Gelb

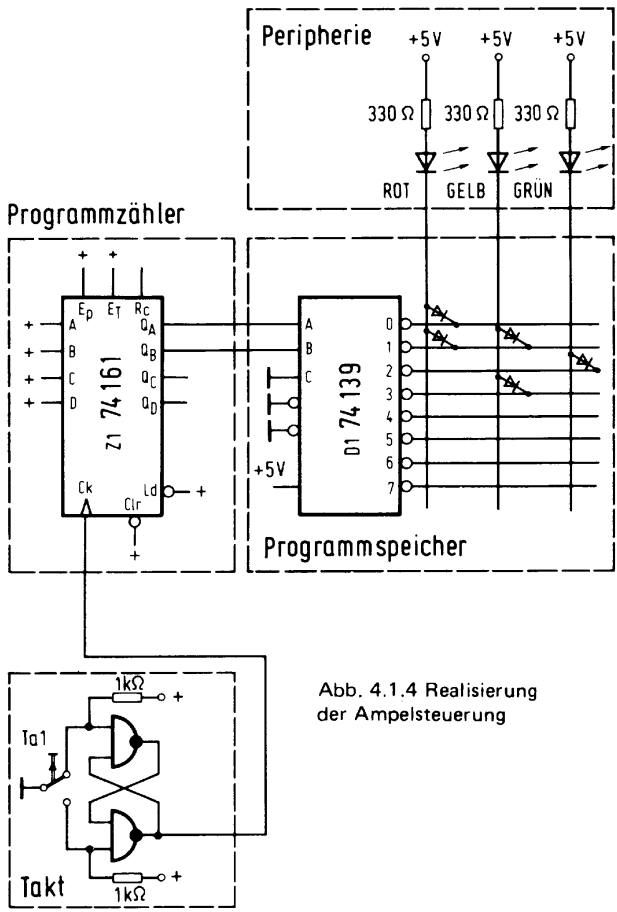


Abb. 4.1.4 Realisierung der Ampelsteuerung

Schritte gewonnen. Ein Sprung an eine beliebige Stelle ist auch einfach zu realisieren, indem die vier Eingänge des Zählers auch an die Diodenmatrix geführt werden. Dann läßt sich das SPRUNGZIEL ebenfalls einstellen.

Abb. 4.1.6 zeigt diese Lösung in einem Blockdiagramm. Tritt der Ladebefehl auf, so wird die Zieladresse an den Programmzähler übertragen; durch das Bit LADE wird das neue Ziel beim nächsten Takt übernommen. Der Lade-Eingang bestimmt also, ob die

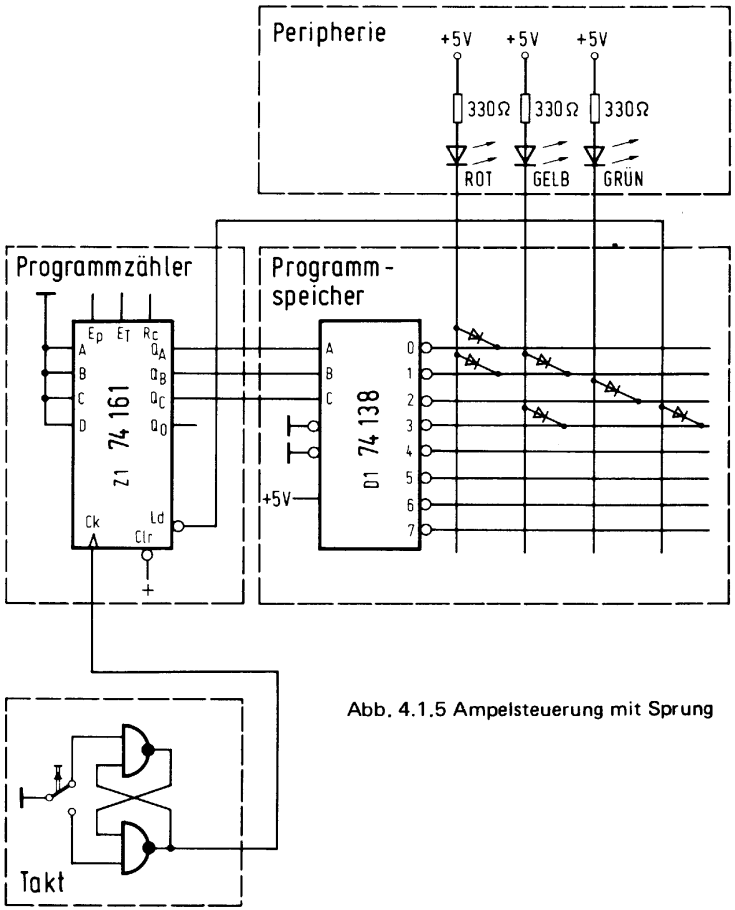


Abb. 4.1.5 Ampelsteuerung mit Sprung

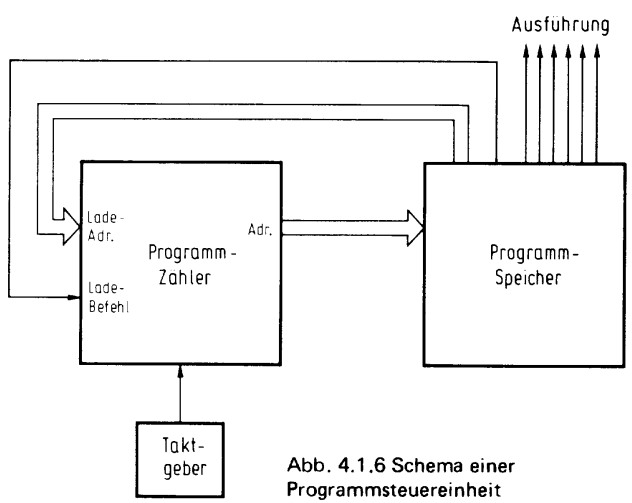


Abb. 4.1.6 Schema einer Programmsteuereinheit

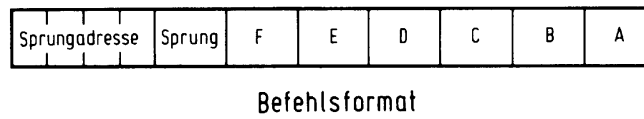


Abb. 4.1.7 Beispiel eines Befehlsformats

Sprungadresse übernommen wird oder nicht. Damit lassen sich auch bedingte Befehle realisieren, die wir bisher noch nicht kennengelernt hatten. Ein bedingter Sprung, wird wie der Name sagt, nur dann ausgeführt, wenn eine Bedingung erfüllt ist. Eine solche Bedingung kann zum Beispiel ein externes Signal sein. Dieses Signal wird z. B. über ein Und-Gatter mit dem Ladebefehl verknüpft und gelangt dann an den Lade-Eingang des Programm-Zählers. Der Sprung wird nur dann ausgeführt, wenn das Signal einen 1-Pegel führt. Auf die gleiche Art und Weise lassen sich die unterschiedlichsten Bedingungen realisieren.

Bei unseren vorherigen Beispielen hatten wir schon tatsächlich einen Befehlsatz mit einem Befehlscode, der auch Maschinenbefehl genannt wird. *Abb. 4.1.7* zeigt einen solchen Maschinenbefehl. Der Maschinenbefehl ist in verschiedene Gruppen unterteilt. Wir sehen da einmal einen Teil mit dem Namen SPRUNGADR, dann ein Bit mit der Bezeichnung SPRUNG und einzelne Bitfelder A bis F. Die Sprungadresse bestimmt das Sprungziel für den nächsten Befehl, und das Bit SPRUNG gibt an, ob dieser Sprung ausgeführt werden soll oder nicht. Die Bit A bis F kontrollieren die jeweilige Aktion, wie z. B. Lampe GRUEN einschalten. In unseren Beispielen wurde jeweils genau jedem Bit eine Funktion zugeordnet, es ist aber auch möglich, einer Reihe von einzelnen Bits codiert mehrere Funktionen zuzuordnen. Mit drei Bits lassen sich 8 verschiedene Funktionen ausführen, dabei kann aber nur jeweils eine Funktion ausgeführt werden. Ebenfalls ist es möglich, die Sprungadresse zur Steuerung von irgendwelchen Aktionen zu verwenden, wenn gerade kein Sprung ausgeführt werden soll, also das Bit SPRUNG auf 0 steht. Dann können diese Aktionen aber nicht gleichzeitig mit einem Sprungbefehl gegeben werden. Bei realen Mikroprozessoren ist dies ganz ähnlich, so gibt es dort eine Vielzahl von Maschinenbefehlen, wobei bei manchen Bitgruppen für einzelne Aktionen vorgesehen sind, bei anderen ein Befehlswort nur eine Aktion ausführen kann. Bei den Sprungadressen wurde auch einiges überlegt. So ist es zum Beispiel möglich, die Sprungadresse erst im nächsten Befehl anzugeben, der dann ganz allein als Adresse gilt. Dann muß dazu nur der vorherige Befehl gemerkt werden. Damit ist es möglich, die Wortbreite, also die Anzahl der Bits, die ein Maschinenbefehl belegt, klein zu halten.

4.2 Speicherbausteine

Unsere Diodenmatrix stellt den Programmspeicher des Rechners. In Zukunft werden wir mit einem eleganteren Speicher arbeiten, dem EPROM. Das EPROM, ein Elektrisch Löschbarer Nur-Lese-Speicher ist im Prinzip wie *Abb. 4.2.1* aufgebaut, was für alle Festwertspeicher, also Speicher, die normalerweise nicht verändert werden, gilt. Dem Speicher werden n Adressen zugeführt, diese gelangen über einen oder mehrere Dekoder an die einzelnen Speicherzellen. Die Ausgänge der Speicherzellen werden dann über Verstärker- und

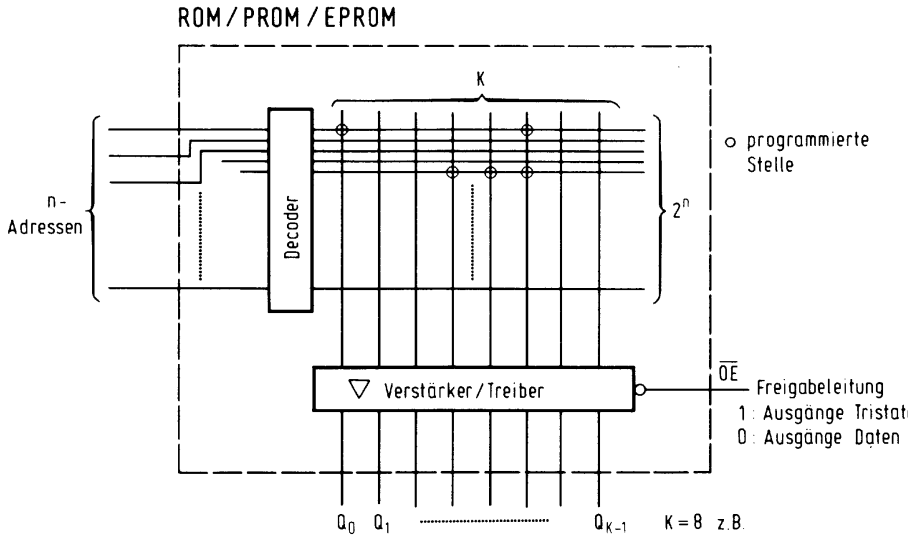


Abb. 4.2.1 Aufbau eines Festwertspeichers

Treiberstufen an die Datenausgänge (oder Ausgang) geleitet. Ein solcher Speicher besitzt dann noch einen Freigabe-Eingang, der es ermöglicht, die Ausgänge des Bausteins in den TRI-State-Zustand zu versetzen. Damit können mehrere solcher Bausteine zusammengeschaltet werden und über den Freigabe (enable)-Eingang wird bestimmt, welcher Baustein schließlich die gewünschten Daten liefern soll.

Bisher hatten wir zur Realisierung des Programmzählers einen Zählerbaustein verwendet. Es gibt aber auch noch einen anderen Weg. *Abb. 4.2.2* zeigt die Schaltung. Das Eprom oder Rom dient als Programmspeicher. Die Flip-Flops Lt1 bis Lt4 als Programmzähler. Dabei sind sie allerdings nicht als Zähler geschaltet. Sie erhalten einen gemeinsamen Takt vom Taktgeber. Mit jeder positiven Taktflanke wird der an den D-Eingängen stehende Wert an die Ausgänge übernommen. Über einen Rücksetzeingang können die Flip-Flops auf den Wert 0 gesetzt werden. Der Ablauf ist nun durch den Inhalt des EPROMs bestimmt. Es wird bei jedem Befehl eine Folgeadresse vorgegeben. Ist die Folgeadresse immer um eins höher als die Adresse bei der dieser Befehl steht, so wird eine Zählfunktion realisiert. Über die Eingänge E1 bis E7 können Bedingungen an die Schaltung gegeben werden. Es wird in Abhängigkeit der unterschiedlichen Kombinationen, die dort anliegen können, zusammen mit der Folgeadresse jeweils eine neue Adresse für das EPROM bestimmt, von der der nächste Befehl zu holen ist. Die Ausgänge Q4 bis Q7 können für Steuerfunktionen frei verwendet werden. Sollen mehr als nur 16 Befehlsadressen verwendet werden, so muß die Adreßbreite der Folgeadresse größer gewählt werden: dabei wird der Platz für die Steuerausgänge kleiner.

Mit dieser Lösung lassen sich universell sehr viele Steuerungsprobleme elegant lösen.

Abb. 4.2.3 zeigt die Lösung für die Verkehrsampelsteuerung. Die Ausgänge St1 bis St3 werden an die Leds Rot, Gelb und Grün geschaltet. Es werden nur vier verschiedene Zu-

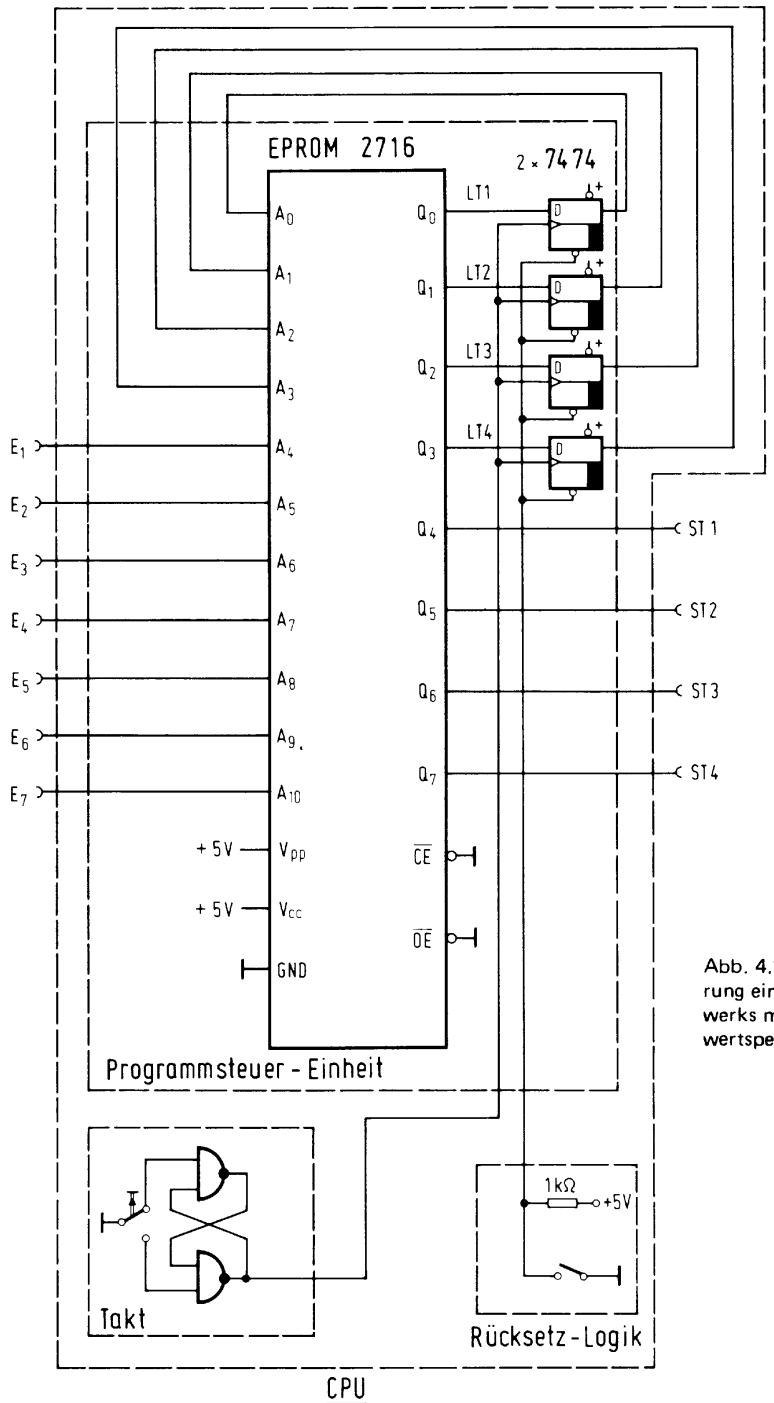


Abb. 4.2.2 Realisierung eines Steuerwerks mit Festwertspeicher

Ampelsteuerung

ST1 = ROT ST2 = GELB ST3 = GRÜN	E1 ... E7 = 0V ST1 ... ST3 aktiv Low 0 = LED an

A_{10}	A_4	A_3	A_2	A_1	A_0	Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	1	1	1	0	0	0	0	1
0	0	0	0	0	1	1	1	0	0	0	0	1	0
0	0	0	0	1	0	1	0	1	1	0	0	1	1
0	0	0	0	1	1	1	1	0	1	0	0	0	0
x	x	x	x	x	x	1	1	1	1	1	1	1	1

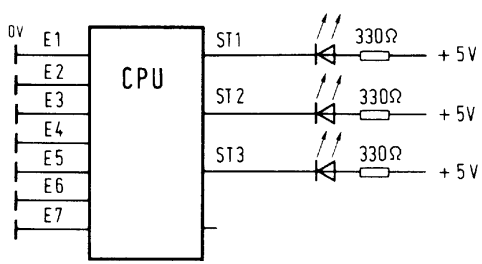


Abb. 4.2.3 Beispiel Ampelsteuerung mit EPROM

stände benötigt und keine Bedingungen. Daher sind nur die ersten vier Zellen des EPROMs mit Daten belegt. Die Folgeadresse wird an Q0 bis Q3 eingestellt und die Steuerung der Leds erfolgt bei Q4 bis Q6.

An die Stelle des EPROMs kann hier auch die Schaltung mit der Diodenmatrix treten.

Vorher ist der Begriff Festwertspeicher gefallen. Dazu gehören alle die Speicher, bei denen die Information permanent vorliegt. Dabei spielt es keine Rolle, ob die Information auch löschbar (z. B. durch UV-Lampe) ist und wieder eingeschrieben werden kann, da dies eine besondere Prozedur voraussetzt. Man kann folgende Festwertspeicher voneinander unterscheiden:

1. ROM. Read Only Memory. Der Inhalt dieses Speichers wird einmal festgelegt, z. B. durch eine Maske bei der Herstellung des integrierten Schaltkreises und kann nie mehr verändert werden.

2. PROM. Programmable Read Only Memory. Der Inhalt des Speichers wird durch einen elektrischen Programmiervorgang vorgegeben. Dabei erfolgt die Programmierung nach der Herstellung des ICs. Das IC behält diesen Inhalt permanent und kann nicht mehr verändert werden. Bei der Programmierung werden z. B. Metallsicherungen durchgebrannt.

Tabelle 6.1

EPROM-Typ	Kapazität	Versorgungsspannung
2708	1024 x 8	+5V, +12V, -5V
2758	1024 x 8	+5V
2716	2048 x 8	+5V
2732	4096 x 8	+5V
2764	8192 x 8	+5V
27128	16384 x 8	+5V

3. EPROM. Erasable Programmable Read Only Memory. Zusätzlich zum PROM kann hier die Information auch wieder gelöscht werden. Dies geschieht durch eine UV-Bestrahlung. Beim Programmiervorgang wird eine Ladung auf ein MOS-Gate aufgebracht. Durch die UV-Strahlung kann diese Ladung wieder abfließen. EPROMs halten ihre Ladung ca. 10 Jahre falls sie keiner Beleuchtung ausgesetzt werden. Deshalb sollte das Löschfenster immer abgedeckt werden.

4. EEPROM. Elektrical Erasable Programmable Read Only Memory. Hier erfolgt der Löschvorgang durch das Anlegen einer Spannung. Die Speicher sind eigentlich die elegantesten, doch sind sie leider nicht mit so hohen Kapazitäten lieferbar, wie dies bei EPROMs der Fall ist.

Tabelle 6.1 gibt eine Übersicht über verschiedene EPROM-Typen und deren Kapazität. Dabei sind hier nur die Intel-Typen aufgeführt, jedoch gelten die Kapazitäten auch für Typen anderen Hersteller.

Die Anschlußbelegung der EPROMs wurde weitgehend vereinheitlicht. *Abb. 4.2.4* zeigt die Belegung für alle gängigen Intel-Typen. EPROMs von Texas-Instruments haben

27128	2764	2732	2716	2708		2708	2716	2732	2764	27128
V_{PP}	V_{PP}				1 28				+5V	+5V
A_{12}	A_{12}				2 27				$\overline{PGM}$	$\overline{PGM}$
A_7	A_7	A_7	A_7	A_7	3 1 24 26	+5V	+5V	+5V	NC	A_{13}
A_6	A_6	A_6	A_6	A_6	4 2 23 25	A_8	A_8	A_8	A_8	A_8
A_5	A_5	A_5	A_5	A_5	5 3 22 24	A_9	A_9	A_9	A_9	A_9
A_4	A_4	A_4	A_4	A_4	6 4 21 23	-5V	V_{PP}	A_{11}	A_{11}	A_{11}
A_3	A_3	A_3	A_3	A_3	7 5 20 22	$\overline{CS}/\overline{WE}$	$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}$	$\overline{OE}$
A_2	A_2	A_2	A_2	A_2	8 6 19 21	+12V	A_{10}	A_{10}	A_{10}	A_{10}
A_1	A_1	A_1	A_1	A_1	9 7 18 20	Progr.	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
A_0	A_0	A_0	A_0	A_0	10 8 17 19	Q_7	Q_7	Q_7	Q_7	Q_7
Q_0	Q_0	Q_0	Q_0	Q_0	11 9 16 18	Q_6	Q_6	Q_6	Q_6	Q_6
Q_1	Q_1	Q_1	Q_1	Q_1	12 10 15 17	Q_5	Q_5	Q_5	Q_5	Q_5
Q_2	Q_2	Q_2	Q_2	Q_2	13 11 14 16	Q_4	Q_4	Q_4	Q_4	Q_4
⊥	⊥	⊥	⊥	⊥	14 12 13 15	Q_3	Q_3	Q_3	Q_3	Q_3

Abb. 4.2.4 Pinzuordnung bei gängigen Festwertspeichern

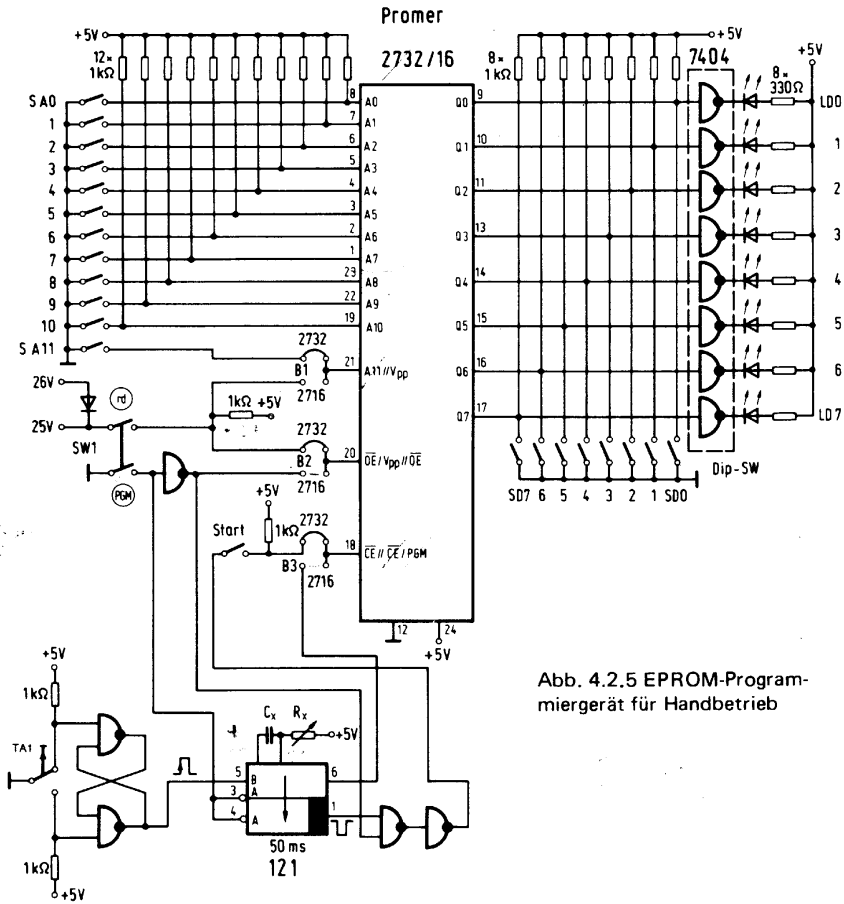


Abb. 4.2.5 EPROM-Programmiergerät für Handbetrieb

dabei manchmal trotz gleicher Typen-Bezeichnung eine unterschiedliche Belegung, so daß vor Anschluß eines EPROMs unbedingt die Datenblätter des jeweiligen Herstellers zu Rate gezogen werden sollten. Die EPROMs unterscheiden sich nur geringfügig in ihrer Anschlußbelegung, was den Entwurf von Systemen mit diesen Bausteinen sehr erleichtert. Die größten Unterschiede bestehen in der Programmierart.

Wir wollen nun ein einfaches Programmiergerät für EPROMs besprechen, daß für die Type 2716 und 2732 verwendbar ist. Damit lassen sich EPROMs erstellen, die wir in den späteren Abschnitten zum Test der CPU-Einheit benötigen. *Abb. 4.2.5* zeigt die Schaltung. Mit den Schaltern SA0 bis SA10 (SA11) läßt sich die Adresse einer Speicherzelle einstellen. An den LEDs LD0 bis LD7 erscheint der Inhalt des EPROMs zur Kontrolle. Zum Programmieren eines EPROMs wird zuvor die Art des PROMs eingestellt (2716 oder 2732 Brücken), dann wird der Schalter SW1 auf PGM gestellt und nun können die Daten an den Schaltern SD0 bis SD7 eingestellt werden. Dabei entspricht der geschlossene Schalter einer logischen 0 und der offene einer 1. An den LEDs ist das Muster kontrollierbar. Nach dem Eintellens der Daten und der Adresse wird die Taste TA1 betätigt. Danach sind die

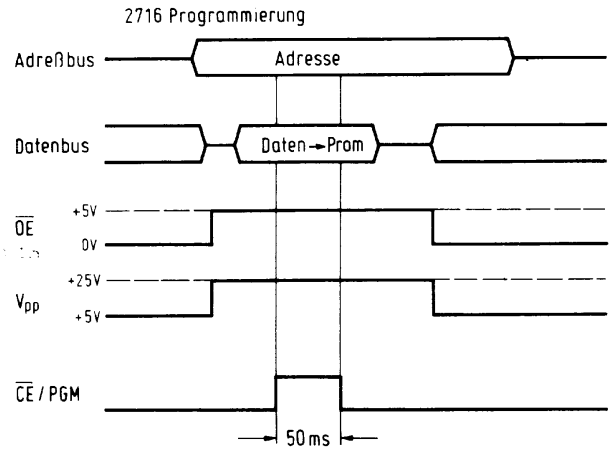


Abb. 4.2.6 Impulsdiagramm für den Programmiervorgang bei 2716

Daten in das EPROMs fest eingeschrieben. Es kann nun die nächste Adresse eingestellt werden und der nächste Datensatz. Wird SW1 auf RD gestellt so können die eingeschriebenen Daten wieder ausgelesen werden. Es empfiehlt sich, dies nach dem Programmieren aller gewünschten Daten zu tun. Vor dem Programmieren ist das EPROM mit lauter 1-Werten gefüllt. Das Gleiche gilt auch nach einem Löschvorgang, der mit einer UV-Lampe durchgeführt werden kann. Am besten sind dafür spezielle UV-Löschgeräte geeignet, doch kann auch eine Höhensonne verwendet werden. Einfache UV-Löschgeräte sind schon für 100 bis 200 DM erhältlich.

Vor der Verwendung des Programmierers muß dieser nach abgeglichen werden. Das Monoflop 74121 muß auf eine Zeit von 50ms sehr exakt eingestellt werden. Dazu wird am besten ein Oszilloscop verwendet. Andere Monoflop-Typen sind an dieser Stelle ebenfalls verwendbar. Bei einer positiven Flanke am Eingang muß das Monoflop triggern und am Ausgang einen 50ms Puls geben, wobei die Polarität zu beachten ist.

Abb. 4.2.6 zeigt das Timing-Diagramm für die Programmierung des Bausteins 2716. Vor dem Programmiervorgang wird der Eingang – OE auf 1 gelegt. Damit sind die Ausgangs-Treiber des Bausteins gesperrt. An den Eingang Vpp, an dem normalerweise ein 5V-Pegel anliegt, wird die Programmierspannung 25V angeschaltet, die übrigens auch sehr genau stimmen muß und unter keinen Umständen größer als 26V sein darf. Der -CE Eingang ist bis dahin noch auf 0V und wird für 50ms auf einen 5V-Pegel geschaltet. Dadurch wird der Programmiervorgang ausgelöst. Ist dieser Puls zu lang, so kann das EPROM zerstört werden. Ist er zu kurz, so wird die Programmierung nicht richtig durchgeführt und die eingespeicherte Information geht nach einer kurzen Zeit wieder verloren.

Abb. 4.2.7 zeigt das Timing für den 2732. Da hier wegen der zusätzlichen Adreßleitung ein Anschluß-Pin weniger zur Verfügung steht, ist der Programmiervorgang anders als beim 2716. Der Eingang -OE dient sowohl als Freigabe-Eingang für die Ausgangstreiber, als auch zur Versorgung für die Programmiespannung. Wird der Eingang auf 25V gelegt, so ist die Programmierung vorbereitet. Diesmal muß aber der -CE Eingang als Ruhesignal einen 5V Pegel liefern und beim Programmieren wird ein 50ms Puls auf 0V gegeben (genau umgekehrt wie beim 2716).

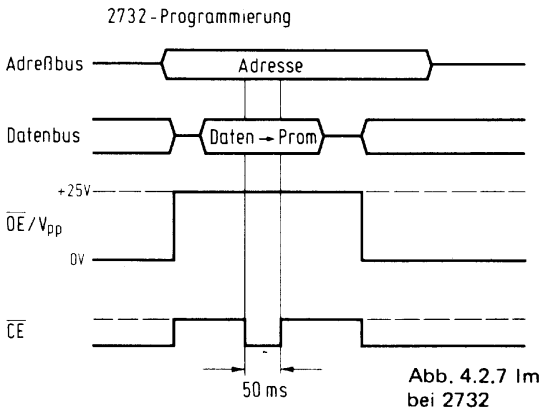


Abb. 4.2.7 Impulsdiagramm für den Programmiervorgang bei 2732

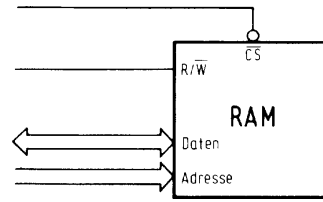


Abb. 4.2.8 Anschlüsse bei einem RAM-Speicher

Unsere Programmierschaltung kann beide Timings erzeugen, es sind jeweils nur die Brücken B1 bis B3 auf die Stellung 2716 oder 2732 zu bringen.

Doch nun zu einer anderen Sorte von Speichern, den RAMs, (Random Access Memory). Hier kann die Information beliebig eingeschrieben und wieder zurückgelesen werden. Bei Stromausfall geht die Information im allgemeinen verloren, es sei denn, es handelt sich um spezielle (teuere) Speicherbausteine. Es werden zwei Gruppen von RAMs unterschieden, zum einen gibt es statische Speicher und dann noch dynamische Speicher. Bei den statischen Speichern wird die Information in einem Flip-Flop festgehalten. Bei den dynamischen Speichern steht ein Kondensator als Speichermedium zur Verfügung. Da sich ein Kondensator (Gate-Anschluß) sehr klein realisieren läßt, sind dynamische Speicher mit einer um ca. den Faktor 4 höheren Kapazität erhältlich. Der Nachteil dieser Speicher liegt darin, daß die Information aus dem Kondensator sehr schnell abfließt (ca. 2 Millisekunden). Um dies zu vermeiden, muß die Information ständig wieder in den Speicher geschrieben werden. Man sagt auch der Speicher muß wieder aufgefrischt werden (refreshed). Dazu ist es aber zum Glück nicht nötig, den Wert auszulesen und dann erneut zu schreiben, sondern es genügt eine Adresse anzuwählen: durch eine Logik im RAM wird automatisch ein Refresh von 128 (oder 256) Zeilen vorgenommen. Innerhalb der 2 ms müssen somit nur 128 (oder 256) Zellen angesprochen werden, bei einer Kapazität von 16384 x 1 oder 65536 x 1, wobei die interne Organisation vom Speichertyp abhängt. Wir wollen uns der Einfachheit halber auf statische RAMs beschränken.

Abb. 4.2.8 zeigt das Blockschema eines RAM-Bausteins. Einmal gibt es Adreßleitungen dann natürlich einen Datenbus, der hier bidirektional eingezeichnet ist, was aber nicht immer so sein muß. Die eingehenden Daten und die Ausgänge können genausogut über getrennte Leitungen geführt werden, wobei allerdings mehr Anschlüsse belegt werden. Mit dem Eingangs -CS (– steht für den Überstreichungsstrich und wird als Quer ausgesprochen, als CS-Quer) wird der RAM-Baustein freigegeben. Der Eingang R/-W bestimmt ob ein Schreib- oder Lese-Zugriff erfolgen soll. Für einen Lese-Zugriff liegt der Eingang auf einem 1-Pegel (da R und nicht -R da steht) und das Auslesen erfolgt, sobald -CS einen 0-Pegel annimmt. Nach einer bestimmten Zeit, der Zugriffszeit (zwischen 150 ns und

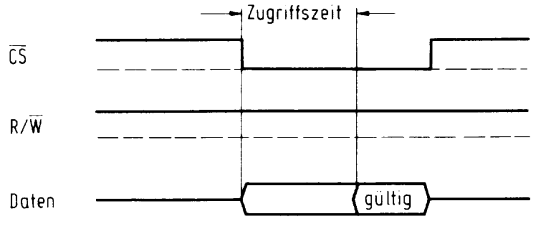


Abb. 4.2.9 Zeitablauf beim Auslesen von Daten

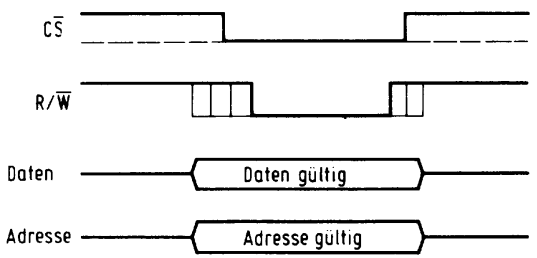
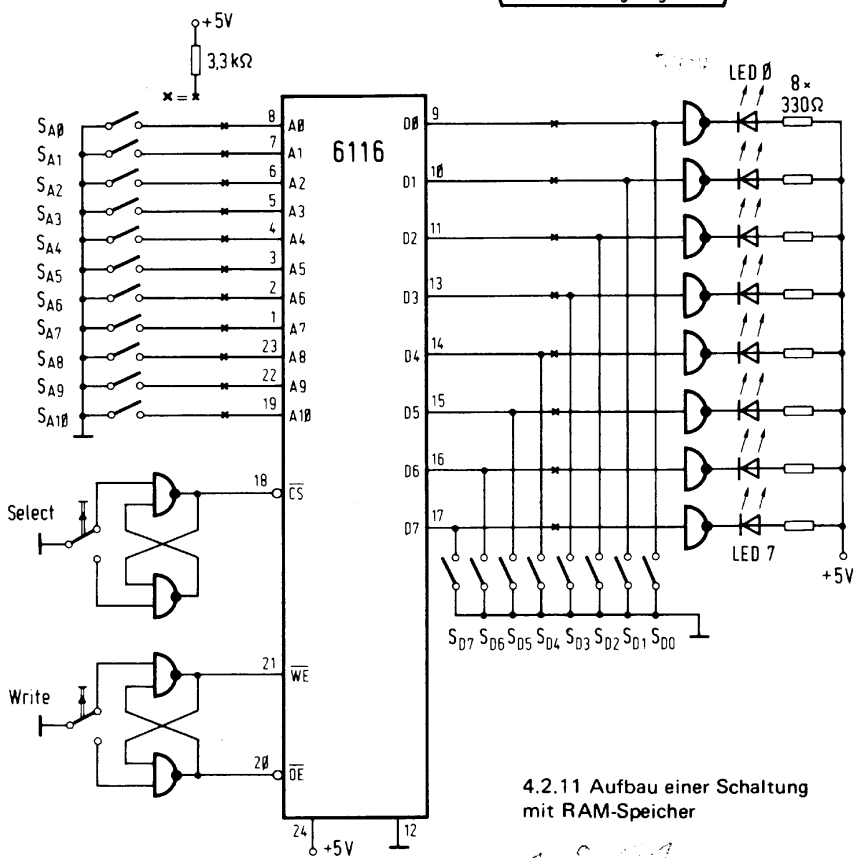


Abb. 4.2.10 Zeitablauf beim Einschreibevorgang



4.2.11 Aufbau einer Schaltung mit RAM-Speicher

450ns je nach Baustein), liegen die Daten an und können weiter verarbeitet werden. *Abb. 4.2.9* zeigt den Vorgang in einem Timing-Diagramm. Bei einem Schreibzugriff werden die einzuschreibenden Daten auf die Dateneingänge gelegt und R/-W und -CS werden auf 0 gelegt, nach einer bestimmten Zeit (ähnlich der Zugriffszeit) sind die Daten in das RAM eingeschrieben und die Signale können wieder auf einen 1-Pegel gebracht werden. *Abb. 4.2.10* zeigt den Zeitablauf.

Zur Übung im Umgang mit RAMs ist in *Abb. 4.2.11* eine praktisch ausgeführte Schaltung mit einem RAM-Baustein angegeben. Mit den Schaltern SA0 bis SA10 läßt sich die Adresse einer Speicherzelle anwählen, an SD0 bis SD7 können im Falle eines Schreibzugriffs die Daten angelegt werden und an LD0 bis LD7 läßt sich der Inhalt einer Zeile ablesen. Die Taste Select bewirkt die Aktivierung des Speicherbausteins und mit der Taste Write kann ein Schreibvorgang eingeleitet werden. Dazu werden zunächst Adresse und Daten eingestellt, dann wird die Taste Write betätigt und schließlich kurz die Taste Select. Danach sind die Daten eingeschrieben.

Damit sind im Prinzip alle Grundlagen für das Arbeiten mit Mikrorechnern geschaffen.

4.3 Die Z80-CPU

Zunächst noch ein paar Grundlagen. *Abb. 4.3.1* zeigt die Verbindung eines Mikroprozessors mit dem Speicher (ROM und/oder RAM). Als CPU werden wir den Prozessor Z80 von Zilog verwenden. Der Anschluß einer CPU an den Speicher erfolgt ähnlich zu dem, was wir bei dem vorigen Abschnitt schon getan haben.

Die CPU besitzt einen Adreßbus, auf dem z. B. die Programmzähleradresse liegen kann. Dann gibt es einen Datenbus, über den der gesamte Informationsaustausch durchgeführt wird. Als Steuersignal werden beim Z80 für den Speicher drei Leitungen verwendet. Die Leitung -RD gibt an wann ein Lese-Zugriff vorliegt. Dann liegt der Pegel auf 0. Bei -WR wird der Schreibzugriff durch ein 0-Signal angegeben. -RD und -WR sind daher niemals zur gleichen Zeit auf 0. Der Ausgang –MREQ schließlich zeigt an, ob ein Spei-

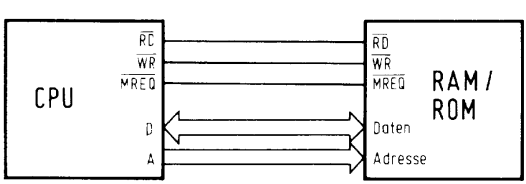


Abb. 4.3.1 Verbindung einer CPU mit einem Speicher

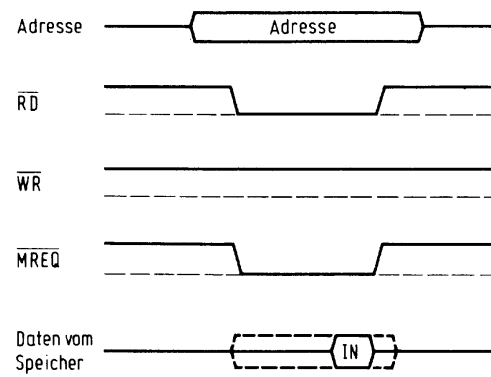


Abb. 4.3.2 Zeitablauf beim Z80, Auslesen.

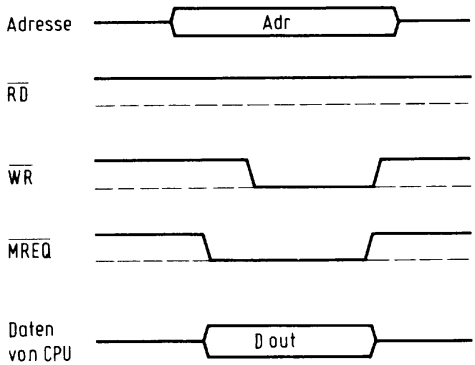


Abb. 4.3.3 Zeitablauf beim Z80, Einschreiben

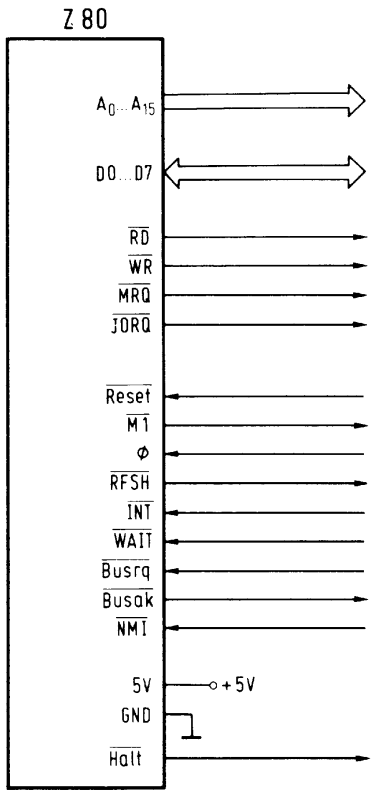


Abb. 4.3.4 Signale beim Z80

cherzugriff gewünscht wird. Denn beim Z80 gibt es auch noch einen IO-Zugriff, auf den wir später nocht zurückkommen.

Abb 4.3.2 zeigt einen Lese-Zugriff der CPU und *Abb. 4.3.3* zeigt einen Schreibzugriff.

Die Z80-CPU *Abb. 4.3.4* besitzt aber noch viele andere Signale, die wir im folgenden einzeln durchgehen wollen:

GND	Masseanschluß der CPU
+5V	Versorgungsspannung, die stabilisiert werden muß.
A0..A15	Tri-State Adreßbus. Bei Speicherzugriffen können damit bis zu 65536 (2 hoch 16) Speicherzellen angesprochen werden. Bei I/O-Zugriffen liegt auf den unteren 8 Leitungen (A0..A7) die Adresse eines Peripherie-Bausteins und beim Refresh-Zyklus (für dynamische Speicher) auf den unteren 7 Leitungen die Refresh-Adresse.
D0..D7	Tri-State Datenbus. Der Bus wird für die Übertragung von Daten an die CPU und von der CPU verwendet.

-M1 Maschine Cycle One. Ein 0-Signal auf dieser Leitung zeigt an, daß die CPU gerade einen Befehlscode holt. -M1 tritt auch zusammen mit -IORQ auf, wenn ein Interrupt quittiert wird (siehe Interrupts im Software-Kapitel).

-MREQ Memory Request. Dieser Tri-State-Ausgang zeigt bei einem 0-Signal an, daß der Adreßbus eine gültige Adresse für einen Speicherzugriff enthält.

-IORQ Input/Output Request. Durch diesen Tri-State Ausgang wird bei einem 0-Signal angezeigt, daß auf den unteren 8 Bits des Adreßbusses eine gültige Adresse für einen I/O-Zugriff vorliegt. Tritt -IROQ gleichzeitig mit -M1 auf, so wird ein Interrupt quittiert.

-RD Read Access. Mit dem Tri-State-Ausgang wird der Wunsch eines Lesezugriffs durch ein 0-Signal angegeben, dabei entscheidet sich durch das Signal – IORQ oder -MREQ ob von einer Peripherie oder Speichereinheit gelesen werden soll.

-WR Write Access. Durch diesen Tri-State-Ausgang wird bei einem 0-Signal ein Schreib-Zugriff angekündigt.

-RFSH Refresh. Ein 0-Signal gibt einen Refresh-Cyclus an. Dann liegt auf den unteren 7 Bits des Adreß-Busses eine Adresse an, die durch einen im Z80 befindlichen Zähler bestimmt ist. Diese Adresse kann bei dynamischen Speichern zum Wiederauffrischen der internen Zellen verwendet werden.

-HALT Halt State. Liegt der Ausgang auf einem 0-Signal dann wurde von der CPU zuvor ein HALT-Befehl ausgeführt. Bei dem Halt werden NOPs zur Aufrechterhaltung des Refresh ausgeführt. Aus dem Halt-Zustand kann nur durch Reset, oder eines freigegebenen Interrupts wieder herausgekommen werden.

-WAIT Wait. Bei diesem Eingang kann durch ein 0-Signal der CPU gesagt werden, daß ein Speicher oder Peripheriegerät noch nicht bereit für einen Datenaustausch ist. Damit können auch langsame Peripherie oder Speichergeräte an die CPU angeschlossen werden. Ein Refresh wird in dieser Zeit nicht durchgeführt.

-INT Interrupt Request. Durch ein 0-Signal kann ein Interrupt gegeben werden. Das Signal wird am Ende eines Instruktions-Zyklus akzeptiert, falls der Interrupt freigegeben wurde und -BUSRQ nicht aktiv ist. Wurde der Interrupt angenommen, so wird dies durch -IORQ und -M1 bestätigt.

-NMI Non Maskable Interrupt. Der Eingang reagiert auf die negative Flanke und wird immer angenommen. Nach einem -NMI-Signal wird die Adresse 66H angesprungen. -BUSRQ darf nicht vorliegen.

-RESET Reset. Damit wird die CPU in den Grundzustand gesetzt. Der Programmzähler wird auf 0 gesetzt und Interrupts werden gesperrt. Das Register I wird auf 0 gesetzt und ebenfalls das Register R. Der Interrupt wird auf Mode 0 gesetzt.

-BUSRQ Bus Request. Durch diesen Eingang wird bei einem 0-Signal der Zugriff auf den CPU-Bus verlangt. Damit kann von einem externen Gerät auf Speicher oder IO zugegriffen werden, ohne das die CPU daran beteiligt wird. Alle Tri-State-Ausgänge der CPU werden in den offenen Zustand überführt.

-BUSAK Bus Acknowledge. Die CPU gibt durch ein 0-Signal an, daß sie den Bus für externe Geräte freigegeben hat.

PHI Phi. Der Takt der CPU. Bei der Standart-CPU Z80 sind maximal 2MHZ erlaubt bei Z80A 4MHZ und bei Z80B 6MHZ. Der Eingang muß einen Pull-up-Widerstand von 330 Ohm erhalten, um von normalen TTL-Gattern angesteuert werden zu können; ansonsten können Störungen im Ablauf auftreten.

Die an den Anschlüssen vorliegenden Signale wollen wir nun im folgenden näher besprechen.

Abb. 4.3.5 zeigt die Relation des Taktes zu dem Befehlsablauf. Als Beispiel sei hier ein Befehl mit einem Lese- und nachfolgenden Speicherschreibzugriff gezeigt. Im M1-Cyclus wird der Befelhscode geholt. Dabei werden 4 Taktzyklen verbraucht. Eine Lese- oder Schreibzugriff benötigt nur 3 Taktzyklen. Alle zu einem Befehl gehörenden Zyklen nennt man Instruction-Cycles.

Abb. 4.3.6 zeigt nochmals genauer den Vorgang bei einem Instruktions-Zyklus. Bei den beiden Takt-Zyklen liegt auf dem Adreßbus die Programmzähleradresse an. Mit dem Signal -MREQ wird die Gültigkeit des Adreßbusses angegeben. Das Signal -M1 kennzeichnen eine Befehlshol-Phase. –MREQ bestimmt, daß ein Zugriff auf den Speicher erfolgen soll und das Signal -RD gibt an, daß vom Speicher gelesen werden soll. Im zweiten Teil des Befehl-Zyklus wird eine Refresh-Adresse für dynamische Speicher ausgegeben.

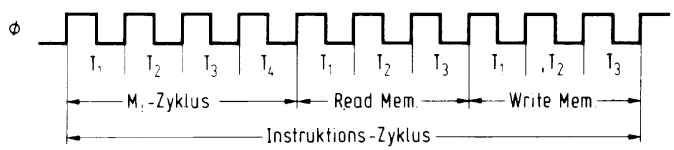


Abb. 4.3.5 Zeitablauf eines Instruktionszyklus

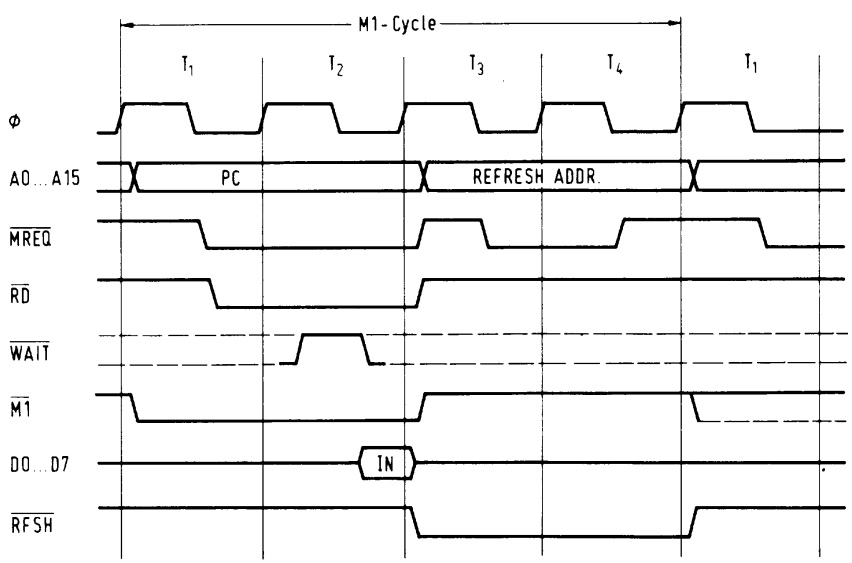


Abb. 4.3.6 Mi1-Zyklus

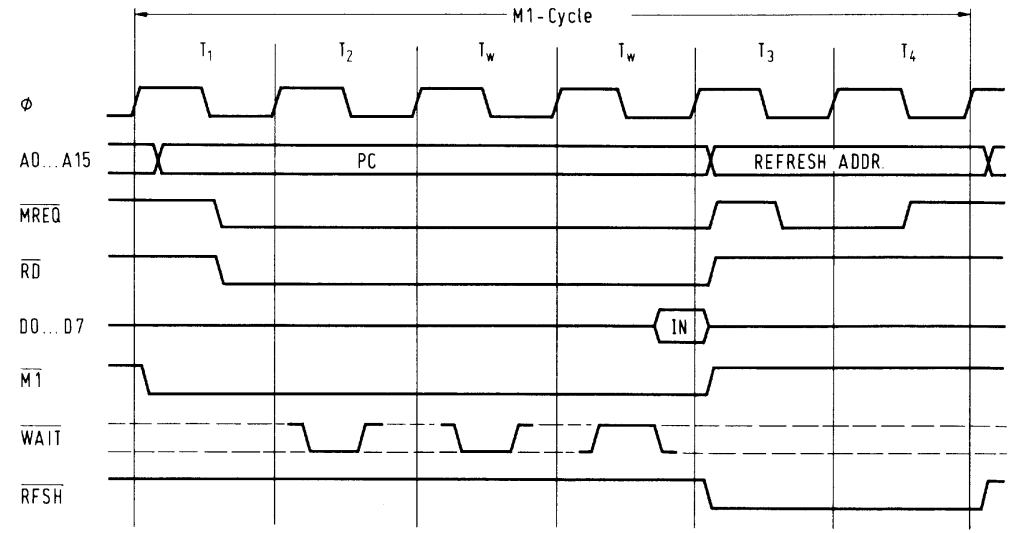


Abb. 4.3.7 Mi1-Zyklus mit Wait-Zyklen

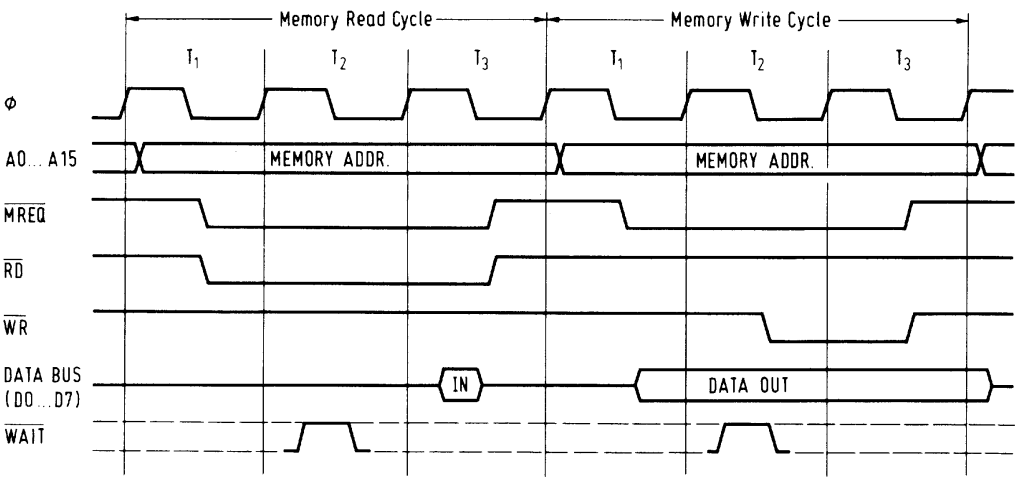


Abb. 4.3.8 Speicher Schreib- und Lese-Zyklus

Wird der Prozessor an Speicher angeschlossen, die zu langsam für einen normalen Zugriff sind, so kann die Zugriffszeit durch Einfügen eines sogenannten Wait-Zyklus verlängert werden. *Abb. 4.3.7* zeigt das Timing-Diagramm dazu. Der Eingang – WAIT wird bei der fallenden Flanke des Taktes bei einem Zugriff abgetastet. Ist die Leitung auf einem 0-Pegel, so wird ein Wait-Zyklus eingefügt. Ist die Leitung -WAIT beim nächsten Zugriff erneut auf 0, so wird wieder ein Wait-Zyklus eingefügt, bis die Leitung bei der fallenden Flanke des Takt-Signals einmal auf 1 war. Danach erfolgt der Zugriff, hier wird das Befehlswort eingelesen. Der M1-Zyklus ist besonders zeitkritisch, weshalb es oftmals

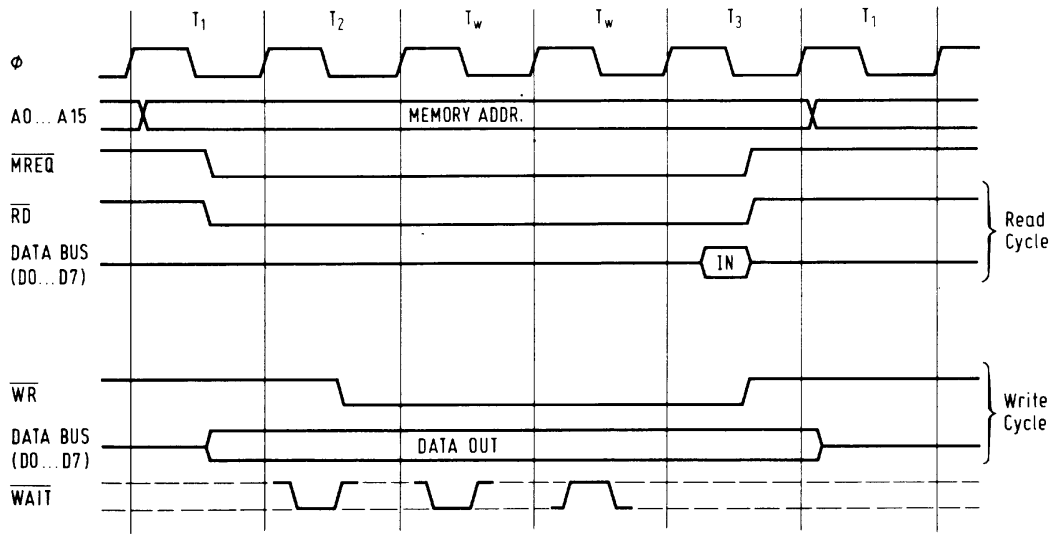


Abb. 4.3.9 Speicherzugriffe mit Wait-Zyklen

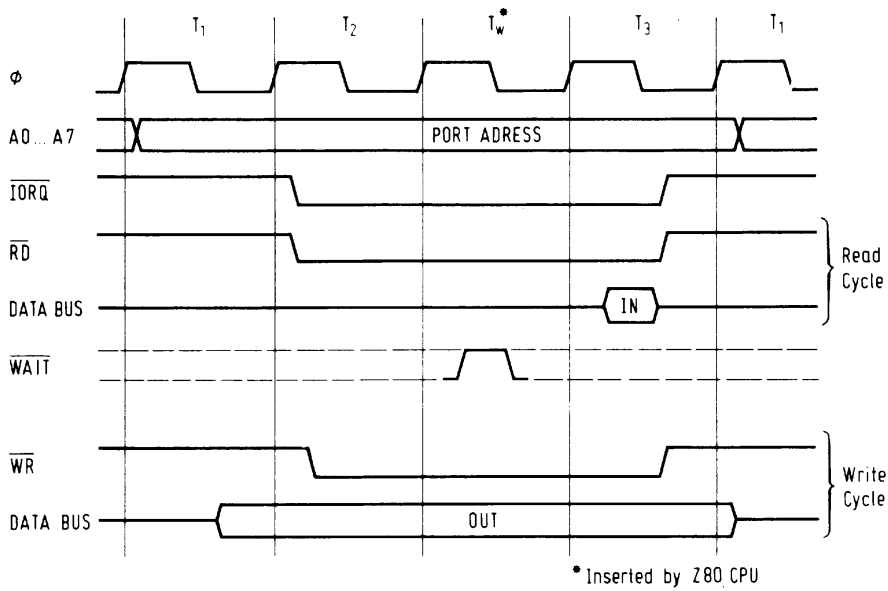


Abb. 4.3.10 I/O-Zugriffe

bei langsamen Speichern genügt, nur dann ein Wait-Signal zu liefern. Der Zeitverlust für die CPU ist dabei prozentual gesehen minimal.

Abb. 4.3.8 zeigt den zeitlichen Ablauf bei einem Speicher Lese- und/oder Schreibzugriff. Die Ankündigung eines Lesezugriffs erfolgt mit den Signalen -MREQ und-RD. Eingelesen werden die Daten am Ende dieser Signale. Ein Schreibzugriff erfolgt mit den Sig-

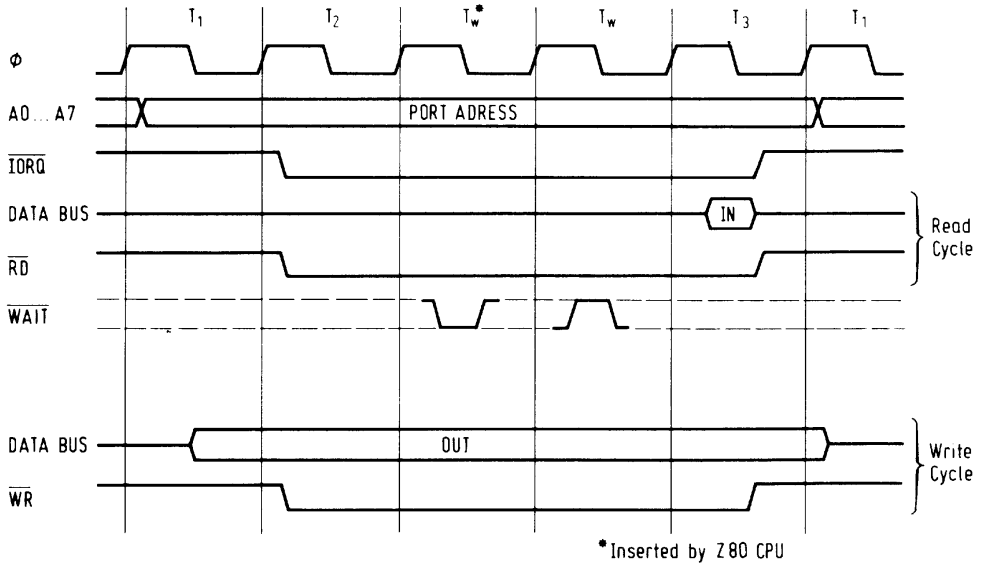


Abb. 4.3.11 I/O-Zugriffe mit Wait-Zyklen

nalen -MREQ und den etwas später folgenden -WR-Signal. Ebenso wie beim Instruktions-Zyklus kann auch hier durch Verwendung -WAIT-Eingangs ein Anschluß an langsame Speicher erreicht werden. *Abb. 4.3.9* zeigt die Situation.

Der Z80 kann neben den Speichern auch noch Peripherie adressieren. Als Adreßraum stehen in bei einem Speicherzugriff 64K zur Verfügung, wohingegen bei einem Zugriff auf eine Peripherie-Einheit nur 256 Adressen zur Verfügung stehen. Er verwendet zur Adressierung die unteren 8 Adreßleitungen.

Abb. 4.3.10 zeigt das Timing für einen Peripheriezugriff. Es läuft im Prinzip genauso wie bei einem Speicherzugriff ab, nur daß anstelle des Signals -MREQ die Leitung -IORQ aktiviert wird. Außerdem wird von der CPU automatisch ein Wait-Zyklus eingefügt, um es auch langsamen Peripherie-Bausteinen zu ermöglichen ohne Zusatzschaltung Daten mit der CPU auszutauschen. Sollte dies dennoch nicht ausreichen, so ist es natürlich auch möglich, den Zugriff noch weiter zu verlängern, wie in *Abb. 4.3.11* dargestellt ist. Dabei ist aber zu beachten, daß ein Refresh während der Zugirffszeit nicht durchgeführt wird und daher bei Verwendung von dynamischen Speichern eine maximale Zeit für einen Wait-Zyklus vorgegeben ist.

Damit zunächst genug Timing vom Z80. Auf die anderen Möglichkeiten soll an dieser Stelle nicht eingegangen werden, da wir sie für das weitere Verständnis nicht benötigen.

4.4 Die Z80-Busstruktur

Die Z80-Baugruppen lassen sich über einen gemeinsamen Bus miteinander verbinden, der in etwa den Anschlüssen des Z80 entspricht. Wir werden im folgenden ein Platinensystem verwenden, das einen solchen Bus bestitzt.

Die Forderungen an ein universelles Experimentier-System für Mikrorechner waren:

1. Übersichtliche Busstruktur. Die Signale die zusammengehören, sollen auch auf dem Bus benachbart sein.

2. Sicherer Bus. Kurzschlüsse zwischen Nachbarleitungen sollen nach Möglichkeit keine katastrophalen Folgen nach sich ziehen.

3. Die Platinen sollen so klein wie möglich sein. So sollen Platinen für Peripheriebausteine kleiner sein als solche für Speicher, da beim Z80 Speicher mehr Leitungen (16 Adreßleitungen) als Peripherie (8 Adreßleitungen) benötigen.

4. Ein billiges Stecksystem sollte Verwendung finden, um LOW-COST Lösungen zu ermöglichen.

Abb. 4.4.1 zeigt die gefundene Anordnung der Signalleitungen auf dem Bus. Auf einer Stelle (1..7) befinden sich die Versorgungsspannungen der Platinen. Dann folgt der Daten-Bus, da er immer benötigt wird. Die Masse trennt dabei die Versorgungsspannungen von dem Datenbus, da ein Kurzschluß zwischen +5V und dem Datenbus zur Zerstörung von Bausteinen führt, ein Kurzschluß nach Masse aber im allgemeinen nicht so gefährlich ist. Es folgen dann (16, 17, 18, 19) die Steuerleitungen des Z80, die den Zugriff auf IO und Speicher bestimmen. Anschließend folgen die 8 unteren Adressen um Peripheriegeräte ansprechen zu können. Bis dahin könnte schon eine Vielzahl von Peripherie-Geräte betrieben werden, ohne die restlichen Signale zu verwenden. Es wurden aber noch ein paar weitere Signale für den Peripherie-Bus hinzugefügt. -RESET dient dem Rücksetzen von Peripherie-Einheiten. -M1 wird für einige Z80-Peripherie-Bausteine verwendet und PHI dient der Taktsteuerung und ist der CPU-Takt. Mit -RFSH können einige Art-fremde Bausteine betreiben werden, die einen dauernden Takt benötigen, der aber wegen spezieller Eigenschaften nicht PHI sein kann. Damit ist der sogenannte SBC-Bus definiert. Durch das Hinzufügen zweier weiterer Signale entsteht der allgemeine IO-Bus, an den sämtliche Peripherie-Bausteine betrieben werden können. Mit -INT wird eine sogenannte Interrupt-Anforderung an den Prozessor gegeben, mit der ein laufendes Programm unterbrochen werden kann und ein Unterprogramm (siehe Software-Kapitel) aufgerufen wird. Nach Beendigung des Unterprogramms (sogenanntes Interrupt-Programm) wird wieder in Hauptprogramm weitergearbeitet. Peripherie-Geräte müssen diese Interrupt-Anforderung manchmal geben können, um sofort bedient zu werden. Beispiel: Es tritt ein Alarm-Zustand in einem Gerät auf, etwa die Spannungsversorgung fällt ab. Es kann dann über diese Leitung ein Interrupt angefordert werden. In dem Interruptprogramm werden dann z. B. alle Speicherinhalte in ein CMOS-RAM, das eine Batterie als Stromversorgung besitzt, gerettet. Das Hauptprogramm wird in Diesem Fall im allgemeinen nicht wieder sofort gestartet, sondern erst nach dem der Stromausfall beseitigt ist.

		-5V	o	1					
		+12V	o	2					
		-12V	o	3					
		+5V	o	4				S	
		+5V	o	5					
		0V	o	6				P	
		0V	o	7					
	|	D0	o	8				E	
	|	D1	o	9					
	|	D2	o	10				I	
	|	D3	o	11					
	|	D4	o	12				C	
	|	D5	o	13					
	|	D6	o	14		S		H	
	|	D7	o	15					
		-RD	o	16		B	I	E	
		-WR	o	17					
		-IORQ	o	18		C	O	R	
		-MREQ	o	19					
	|	A0	o	20					
	|	A1	o	21		-	-	-	
	|	A2	o	22					
	|	A3	o	23					
	|	A4	o	24		B	B	B	
	|	A5	o	25					
	|	A6	o	26		U	U	U	
	|	A7	o	27					
		-RESET	o	28		S	S	S	
		-M1	o	29					
		PHI	o	30					
		-RFSH	o	31					
		-INT	o	32					
		-WAIT	o	33					
	|	A8	o	34					
	|	A9	o	35					
	|	A10	o	36					
	|	A11	o	37					
	|	A12	o	38					
	|	A13	o	39					
	|	A14	o	40					
	|	A15	o	41					
		BANKEN	o	42					
		-BUSRQ	o	43					
		-BUSAK	o	44					
		PI	o	45					
		PO	o	46					
		-NMI	o	47					
			o	48					
			o	49					
			o	50					

Abb. 4.4.1 Busbelegung

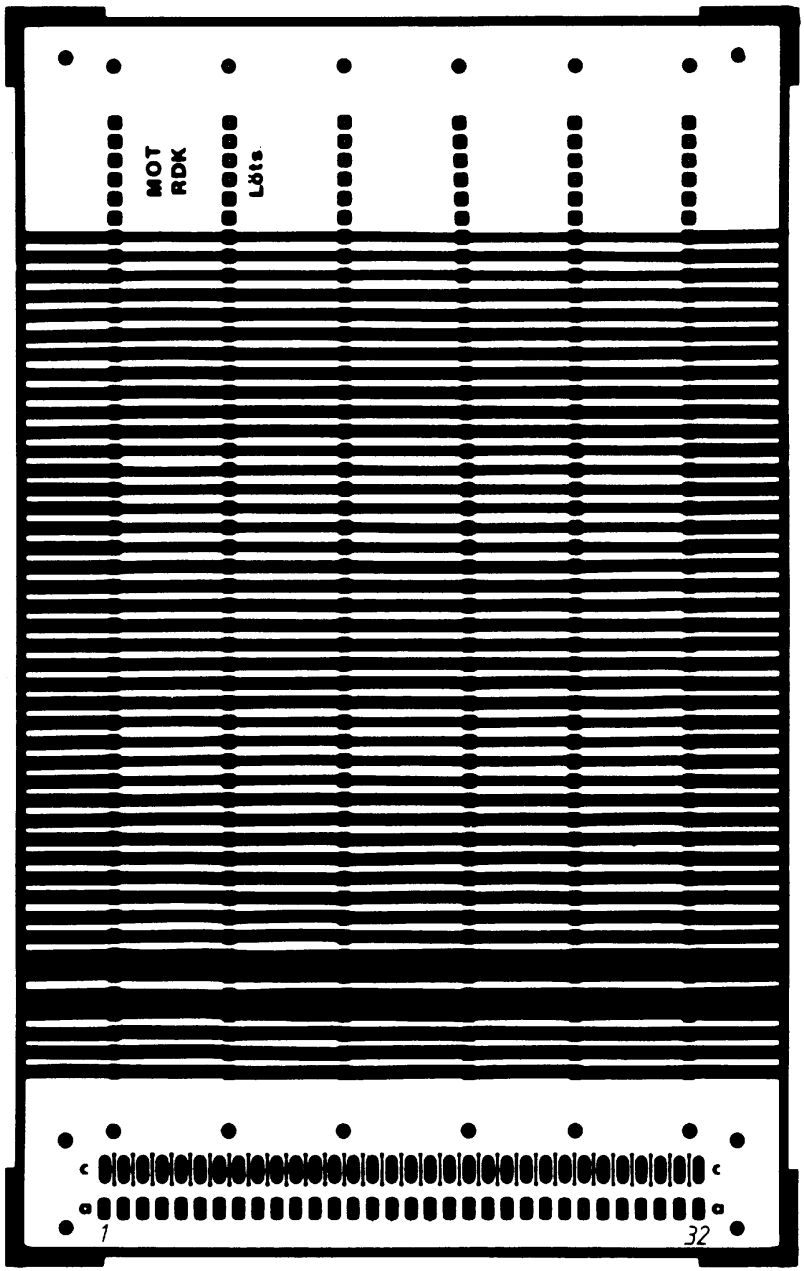


Abb. 4.4.2 Grundplatte für das Experimentiersystem

Die Leitung -WAIT dient langsamen Peripherie-Bausteinen, um die Zugriffszeit verlängern zu können. Die Leitung -WAIT sollte aber nicht dazu verwendet werden, um z. B. auf das Ende eines PROM-Programmiervorgangs zu warten und damit die CPU für lange Zeit (50ms) zu stoppen. In dieser Zeit kann die CPU andere Dinge tun und sie hat immer noch die Möglichkeit einmal kurz nachzufragen, ob der Progammiervorgang schon beendet ist.

Sollen Speicher an den Z80 angeschlossen werden, so werden weitere Signale benötigt. Dann ergibt sich der Speicher-IO-Bus. Es werden zunächst die restlichen Adressen A8 bis A15 benötigt. Dann folgt ein Signal mit dem Namen BANKEN, das nicht zu den Z80 Standard-Signalen gehört. Es hat die Aufgabe, den Speicher freizugeben, wenn es auf einem 1-Pegel liegt. Damit ist es möglich, mit einem zusätzlichen Dekoder mehr als 64K Speicher zu adressieren. Die Signale -BUSRQ und -BUSAK dienen dem Direkten-Speicher-Zugriff (DMA) von anderen Geräten als der CPU, die den Speicher benötigen. PI und PO sind reservierte Signale für eine spätere Entwicklung der Interrupt-Struktur und -NMI ist das CPU-Signal für einen Nicht-Maskierbaren Interrupt. Der normale Interrupt INT wird nur dann ausgeführt, wenn er durch einen Programm-Befehl erlaubt wurde. Der NMI-Eingang reagiert immer mit einer Programm-Unterbrechung und ruft ein Unterprogramm auf, das auf der Adresse 66H des Prozessor-Adreßraums liegt.

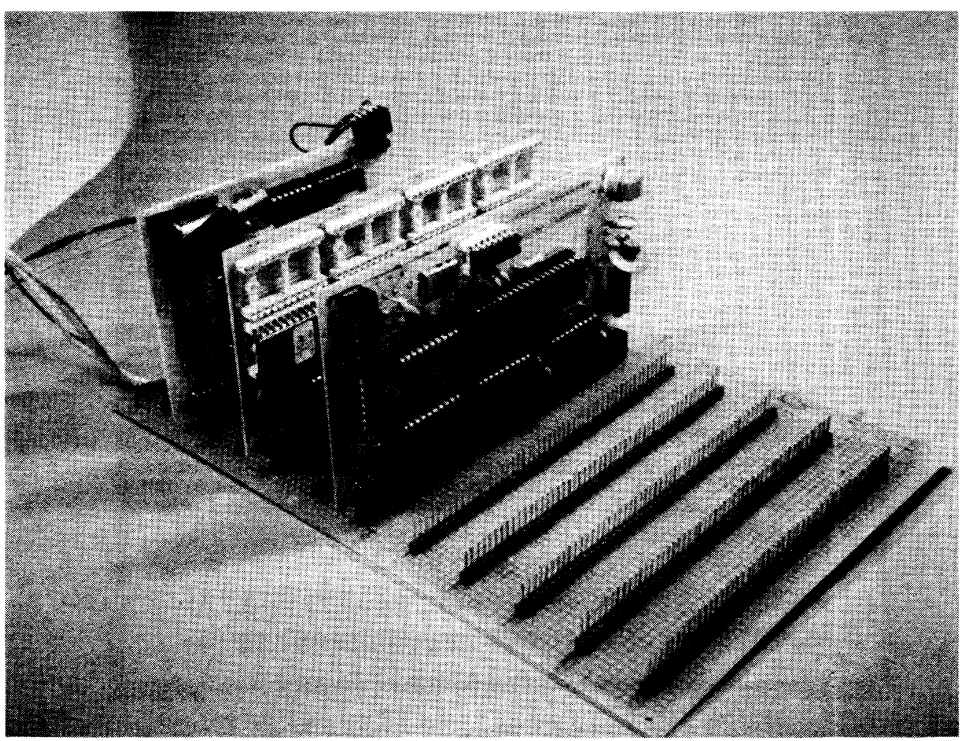


Abb. 4.1 Aufbau der Platinen auf einer Grundplatte

Die einzelnen Baugruppen des Experimentiersystems müssen auf eine Grundplatte aufgebaut werden. *Abb. 4.4.2* zeigt eine solche Grundplatte, wie sie auch, wie im übrigen alle hier vorgestellten Platinen im Handel erhältlich ist (siehe Bezugsquellenverzeichnis). Die Grundplatte hat ein Standard-Europaformat (160 x 100 mm). Sie kann damit in ein Einschubsystem mit gebräuchlichem Bus (z. B. ECB, ELZET, SMP) eingesteckt werden und die Verdrahtung auf diesem Bus kann ggf. durchgeführt werden. Normalerweise wird sie aber als reine Grundplatte verwendet, auf die wir nun nach und nach den Mikrocomputer aufbauen wollen.

4.5 Die SBC-Karte (Single Board Computer)

Die SBC-Karte enthält alles, was für einen vollständigen Mikrocomputer nötig ist, außer der Peripherie. Mit dieser Karte und einer weiteren Karte, der Karte SER läßt sich ein vollständiger Rechner aufbauen, mit dem mit Hilfe eines zusätzlichen Terminals (Datensichtgerät oder Teletype) kleine Programme entwickelt werden können. Für Leser, die kein Terminal besitzen, gibt es die Karte CRT und KEY, mit denen dieses Terminal selbst gebaut wird; es wird dann nur noch ein Bildschirm (Monitor oder TV-Gerät mit HF-Modulator) sowie eine Tastatur benötigt.

Der SBC enthält folgende Bauteile. Einmal natürlich die Z80-CPU, dann ein 4K EPROM, in dem sich ein kleines Monitor-Programm für die Programmeingabe befindet und einem 1K RAM-Speicher, der der Aufnahme des Benutzerprogramms bei der Programmentwicklung dient. Das Monitorprogramm wird im Software-Kapitel ausführlich beschrieben. Wir wollen uns hier nun systematisch mit dem Aufbau der SBC-Karte beschäftigen. *Abb. 4.5.1* zeigt die Gesamtschaltung der SBC-Karte. Die CPU wird zentral mit einem Steuertakt versorgt, der mit Hilfe eines Quarz-Generators erzeugt wird. Dazu dient das IC2 ein 7404. Als Quarz wird ein 2MHZ-Typ verwendet. Der Kondensator C2 mit 100 pF kann auch weggelassen werden, abhängig vom verwendeten Quarz-Typ. An Pin 8 des IC2 liegt der Takt an. Ein Pull-Up-Widerstand R4 „zieht" die Ausgangsamplitude des Oszillators auf 5V, so daß der Z80 den Takt auch weiterverwenden kann. Der Takt wird auch auf den Bus geführt. Der Z80 erhält von außen ein Signal -RESET, daß von einer Taste kommt, die den Prozessor rücksetzen soll. Die Taste befindet sich nicht mit auf der CPU-Karte, um Platz zu sparen. Es kann eine normale Taste verwendet werden, besser aber eine prellfreie, um die CPU möglichst sauber zu starten.

Die Eingänge -NMI, -INT, -WAIT und -BUSRQ sind auf +5V gelegt, da sie in der Minimalversion nicht benötigt werden. Der Ausgang -RFSH ist an den Bus geführt, da er für ein paar Peripherie-Baustein-Arten benötigt wird, wie auch das Signal –M1. Der Adreß- und Datenbus ist mit dem Speicherteil verbunden. Das 4K EPROM belegt die Adressen A0 bis A11 der RAM-Speicher belegt nur A0 bis A9. Anstelle des 4K EPROMS (2732) kann auch ein 2K EPROM (2716) verwendet werden; wenn eine Brücke J von 32 auf die Stellung 16 umgelötet wird. Dabei ist die Brücke für 32 schon auf die Platine standardmäßig vorhanden. Bei den beiden Speichern IC5 und IC6 handelt es sich um einen Spei-

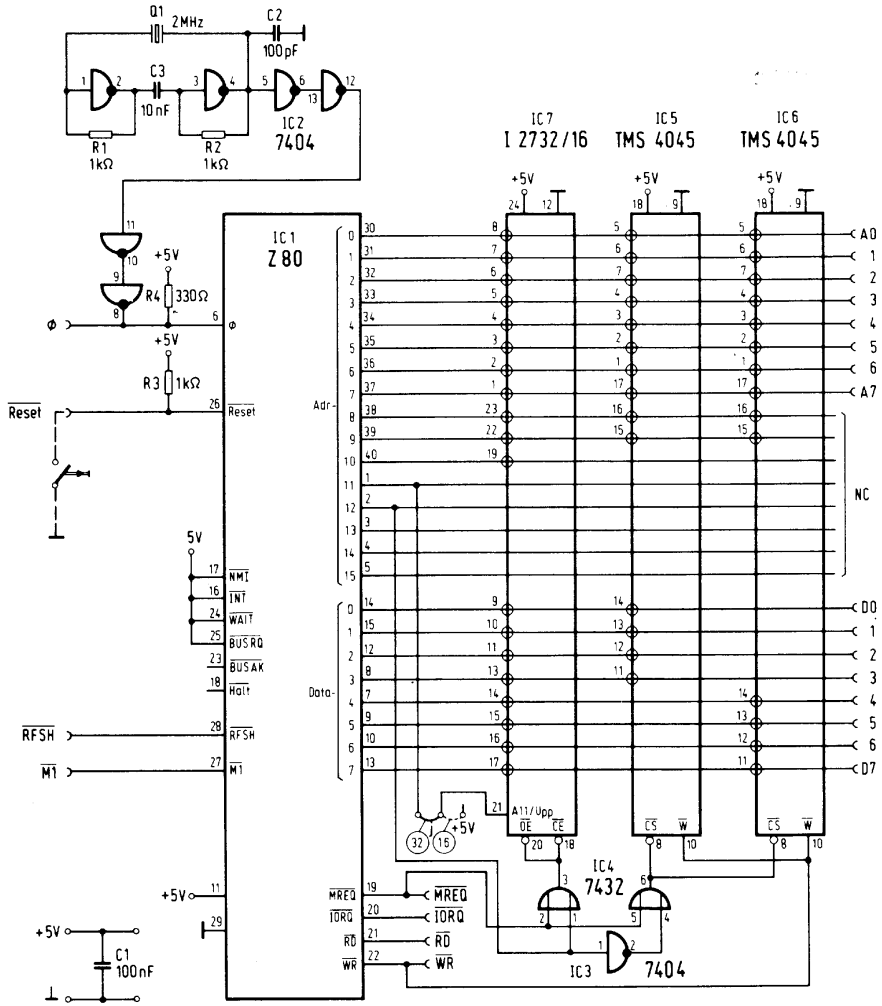


Abb. 4.5.1 Schaltung der SBC-Karte

cher mit einer 1K x 4 Organisation, daß heißt der Speicher besitzt eine Gesamtkapazität von 4096 Bits, aber mit 4 Ausgängen parallel, so daß zwei dieser Speicher für eine 1K x 8 Organisation benötigt werden. Anstelle des Typs TMS 4045 kann auch der Typ 2114 verwendet werden.

Die Speicher müssen, wie wir aus den vorherigen Abschnitten gelernt haben, auch noch selektiert werden. Dazu dient eine Logik, die aus den Bausteinen IC4 und IC3 besteht. Da hier nur Speicher adressiert werden sollten, sind die Signale -MREQ, -RD und -WR zu verwenden, um den Zugriff zu steuern. Die Z80 CPU fängt nach dem RESET bei der Speicheradresse 0 an, ein Programm auszuführen. Also muß man dafür sorgen, daß sich dort nach dem Stromeinschalten und RESET ein ROM-Teil befindet, in dem

schon ein Programm vorhanden ist. Würde die CPU einen RAM-Bereich vorfinden, so ist dessen Inhalt nicht definiert und die CPU würde wild irgendwelche Befehle ausführen. Wir müssen also dafür sorgen, daß von Adresse 0 an ROM (EPROM)-Bereich liegt. Dies geschieht mit dem Gatter IC4. An Pin 1 liegt die Adresse A12 an. Am Pin 2 des Oder-Gatters liegt das Signal -MREQ. Am Ausgang der Oder-Verknüpfung liegt nur dann ein 0-Signal an, wenn an beiden Eingängen zur gleichen Zeit 0-Signal anliegt, wenn also die Adresse A12 auf 0 liegt und -MREQ auch, also ein Speicherzugriff vorliegt. Der Ausgang des Oder-Getters wird aber an die beiden Selekt-Eingägen des EPROMs geleitet. Das EPROM wird bei einem 0-Signal freigegeben, also immer bei einem Speicherzugriff im Bereich 0 bis 4095. Dabei ist die Auswahl unabhängig von den Adressen A13, A14 und A15, da diese überhaupt nicht beschaltet sind. Der Speicher wird von dem anderen Oder-Gatter ausgewählt. Dabi erhält dieses Oder-Gatter an Pin 4 das inverse Signal der Adresse A12. Damit wird der RAM-Speicher immer dann freigegeben, wenn ein Speicherzugriff vorliegt (wegen -MREQ) und die Adreßleitung A12 ein 1-Signal trägt. Also wird der Bereich 4096 bis 4096+1023 angesprochen. Da hier die Adressen A13, A14, A15 und A10 und A11 nicht verwendet wurden, ist die Auswahl unabhängig von dem Zustand auf diesen Leitungen. Man sagt auch, der Speicher ist nicht eindeutig adressiert. Das bedeutet, daß gleiche Speicherzellen mit unterschiedlichen Adressen angesprochen werden können.

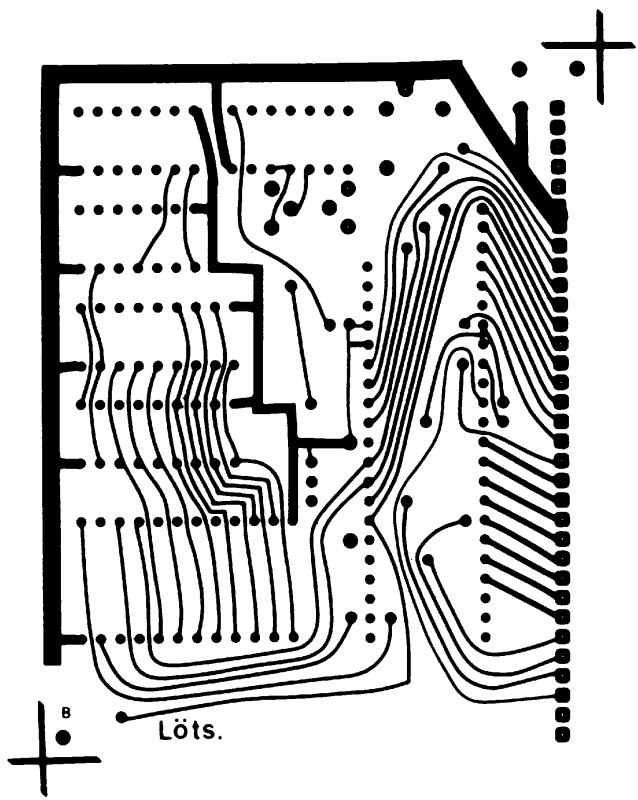


Abb. 4.5.2 Lötseite der SBC-Karte

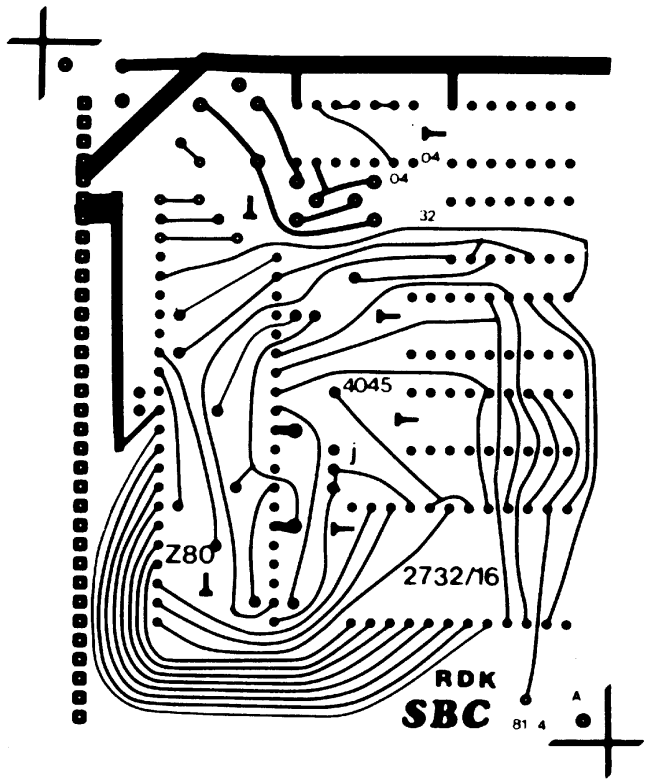


Abb. 4.5.3 Bestückungsseite der SBC-Karte

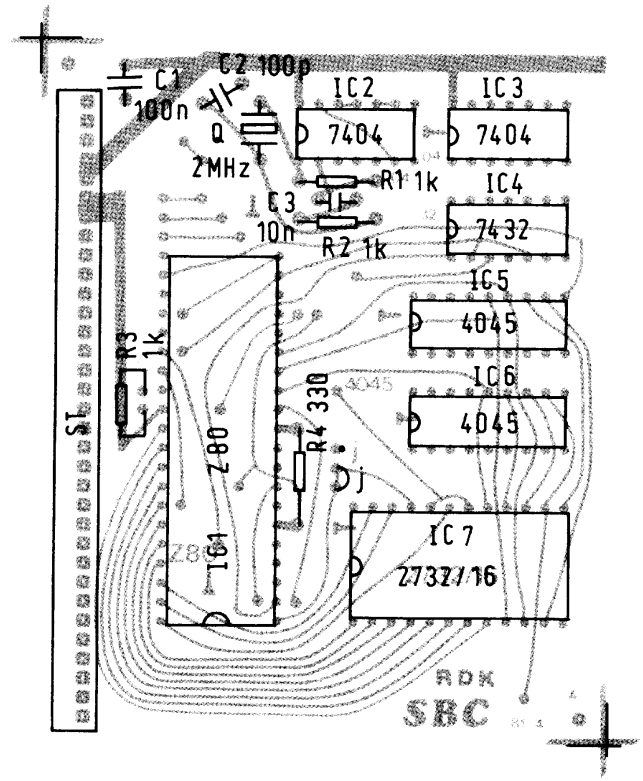


Abb. 4.5.4 Bestückungsplan der SBC-Karte

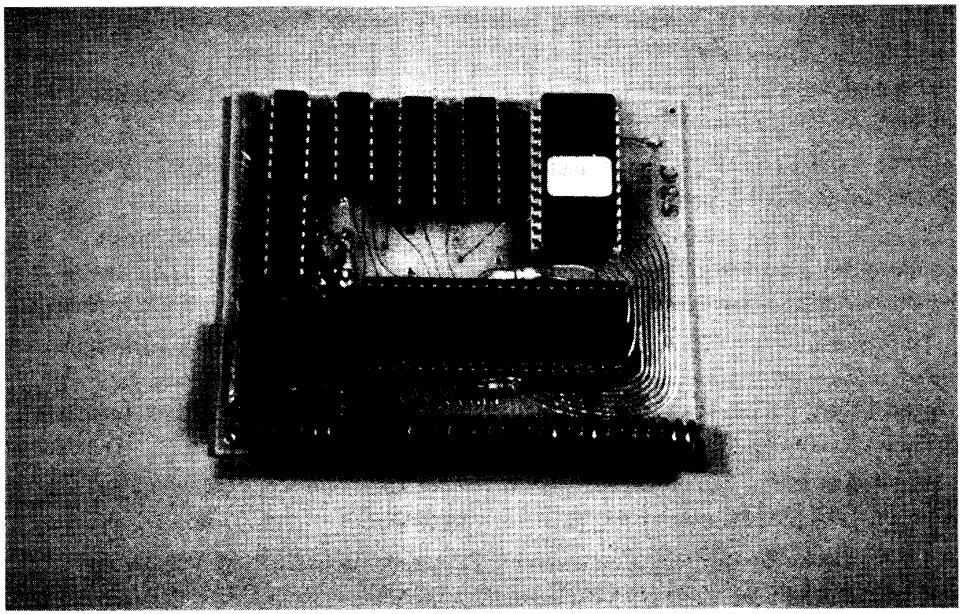


Abb. 4.2 SBC-Karte

Die Leitungen A0 bis A7 und D0 bis D7 sind an den gemeinsamen Bus geführt: ebenfalls die Signale -MREQ, -IORQ, -RD und -WR. Wir können nun mit den systematischen Zusammenbau der Platine beginnen. *Abb. 4.5.2* zeigt die Lötseite der Platine, *Abb. 4.5.3* zeigt die Bestückungsseite und *Abb. 4.5.4* zeigt den Bestückungsplan.

Gleich noch ein Hinweis. Bei allen Platinen unbedingt Sockel für die einzelnen ICs verwenden, da sonst eine Fehlersuche unmöglich ist.

1. Beginnen wir also mit der Bestückung, indem alle Sockel in die Platine eingelötet werden. Es ist darauf zu achten, daß keine Verbindungen zwischen Leiterbahnen durch Lötbatzen hergestellt werden. Es soll auch bei der Platinenoberseite unter den ICs auf Kurzschlüsse die bei der Herstellung der Platine ggf. entstanden sind, geachtet werden. Nach dem alle Sockel eingelötet sind erfolgt der erste Test.

2. Messen des Widerstandes zwischen der +5V Busleitung und der Masse. Der Widerstand muß unendlich sein, sonst liegt an irgend einer Stelle in Kurzschluß zwischen Masse und 5V vor.

3. Einlöten des Kondensators C1 100nF Keramische Scheibe. Dies ist ein Abblockkondensator, der Störungen auf den Versorgungsleitungen unterdrücken soll.

4. Einlöten der passiven Bauteile des Quarzgenerators: Quarz/C2, R1, R2 und C3.

5. Einlöten der passiven Bauteile für die CPU: R3 und R4. Die Brücke J bleibt wie sie auf der Platine schon eingestellt ist. – Nun sind alle Bauteile bis auf die ICs bestückt.

6. Anschluß an eine 5V Versorgung. Dabei ist auf richtige Polung zu achten (Vergleich mit der Busbelegung in Abb. 6.4.1).

7. Einschalten der Versorgungsspannung. Messen an Pin 14 von IC2, dort muß +5V anleigen. An Pin 7 muß OV anliegen.

8. Ausschalten der Versorgungsspannung. Das IC2 ein 7404 (kein 74LS04 verwenden) wird in den Sockel eingesetzt.

9. Einschalten der Versorgungsspannung. An Pin 8 von IC2 muß eine Rechteckspannung mit einer Frequenz von 2MHZ anliegen. Auf dem Oszilloscop ist die Breite der Periode einer Schwingung 500ns. Dieses Signal muß auch an Pin 6 der CPU anliegen.

10. Anschluß einer externen RESET-Taste. Es kann z. B. eine prellfreie Taste (siehe vorherige Kapitel) verwendet werden. Im Ruhezustand muß am Ausgang der Taste ein 1-Signal liegen. Für die Dauer der Betätigung ein 0-Signal. Der Ausgang dieser Taste wird an die Busleitung Nr. 28 angeschlossen.

11. Messen an Pin 26 der CPU. Bei Betätigen der Taste muß der Pegel von 1 auf 0 springen, um beim loslassen wieder auf 1 zurückgehen.

12. Nun kann die CPU eingesetzt werden. Zuvor aber unbedingt die Versorgungsspannung abschalten.

13. Um die CPU testen zu können, werden wir uns ein einfaches „Testprom" anfertigen, das nur aus einer Kurzschlußbrücke besteht. *Abb. 4.5.5.* zeigt die Schaltung. Die Datenleitungen werden auf 0 V gelegt. Dazu kann ein leerer 24-pol Sockel verwendet werden. Die Pin-Nummern sind in der Abbildung angegeben. Nach Einschalten der Versorgungsspannung und nach einem RESET ist die CPU aktiviert und ein erneuter Test kann erfolgen. Mit dem Oszilloscop sind die einzelnen Ausgänge der CPU zu prüfen. An -RD des ICs muß eine Pulsfolge anliegen, genauso wie bei -MREQ und -M1. Die Adreßausgänge A0 bis A15 müssen eine absteigende Frequenz erzeugen. Dabei liegt an A0 die höchste Frequenz. Wegen des Refresh-Zyklus sind Teile der Impulsfolge unterbrochen, dennoch muß auf dem Scop erkennbar sein, daß jeder nachfolgende Adreßausgang die halbe Frequenz des vorherigen (von A0 bis A15) beinhaltet. Damit ist das prinzipielle Funktionieren sichergestellt. Bei dem Testsockel wurde der NOP-Befehl des Z80 verwendet, der die Codierung 00 besitzt. Nun kann noch die Dekodierung ge-

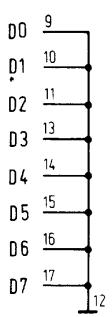


Abb. 4.5.5 Testbelegung für NOP-Befehl 00h

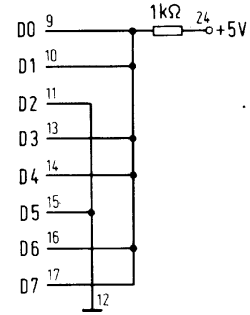


Abb. 4.5.6 IN-Befehl 0DBh

testet werden. Der Eingang -CS des EPROMs muß ebenfalls eine Pulsfolge erhalten, genauso wie die beiden -CS Eingänge der RAM-Bausteine (die noch nicht eingesetzt sind). Mit einem zweiten Kanal am Scop läßt sich überprüfen, daß immer nur entweder an EPROM oder am RAM ein negativer Selektier-Puls erscheint.

14. Auf die gleiche Weise läßt sich auch noch der Eingabe-Befehl testen. Abb. 4.5.6 zeigt die Belegung des Eprom zeigt die Belegung des Eprom-Steckers. Es wird hier ein Eingabe-Befehl angelegt der den Code DB besitzt. Ferner besitzt er noch eine Adresse als Parameter, die aber hier auch DB ist, also ein Befehl DB DB simuliert wird. Nach Start der CPU muß nun der Ausgang -IORQ pulsen.

15. Für die nachfolgenden Schritte ist es nötig, einige Test-EPROMs zu brennen. z. B. mit dem im vorherigen Kapitel beschriebenen Programmiergerät, oder die nachfolgenden Tests werden einfach ignoriert und der Test als Gesamttest mit dem fertigen Monitorprogramm durchgeführt, wie später noch gezeigt wird. Dazu noch kurz etwas zur sedezimalen Schreibweise (früher auch HEXADEZIMAL genannt). Die Codes bei Programmlistings sind bei uns immer sedezimal. Zur Eingabe des EPROM folgendes Beispiel:

Code: A7
wird im EPROM wie folgt eingebrannt:

Bitfolge:

1 0 1 0 0 1 1 1

mit der Zuordnung:

D7 D6 D5 D4 D3 D2 D1 D0

16. Die Speicherbausteine werden nun zusätzlich eingesetzt. Es folgt nun aber nochmals ein I/O-Test, diesmal aber mit EPROM. In dem EPROM muß ein kleines Programm stehen, wie es in *Abb. 4.5.7* gezeigt ist. Das Programm besteht aus zwei Befehlen, die aber 5 Bytes belegen. Zunächst wird ein Eingabe-Befehl gegeben. Der Befehl ist DB 00 und liest einen I/0-Port mit Adresse 0 in den Akkumulator. Dabei ist es für den Test nicht wichtig, ob der I/0-Port tatsächlich vorhanden ist oder, wie in unserem Fall, nicht vorhanden ist. Der nächste Befehl ist ein Sprungbefehl auf die Adresse 0 mit der Codierung C3 00 00. Wird diese Sequenz von der CPU ausgeführt, so ergeben sich dauernde Eingabe-Befehle. Es ergibt sich ein ähnliches Muster wie in Test 14, nur daß die -IORQ Pulse in einem größeren Abstand aufeinanderfolgen. Der Adreßbus sieht ebenfalls anders aus, da hier nicht der gesamte Speicherraum adressiert wird wie vorher, sondern nur die

```
                        ; Testprogramm  fuer Z80-CPU Karte SBC
                        ; Pulsen des Signals IORQ+RD

0000'   DB 00           schleife: in a,(0)          ;holen eines wertes nach a
0002'   C3 0000'                 jp schleife       ;wiederholung

                                  end
```

Abb. 4.5.7 Testschleife Eingabe

```
                        ; Testprogramm  fuer Z80-CPU Karte SBC
                        ; Pulsen des Signals IORQ+WR

0000'   D3 00           schleife: out (0),a         ;ausgabe eines wertes
0002'   C3 0000'                  jp schleife      ;wiederholung

                                  end
```

Abb. 4.5.8 Testschleife Ausgabe

```
                        ; Testprogramm  fuer Z80-CPU Karte SBC
                        ; Kombinierter Test  pulsgruppe auf IORQ
                        ; lesen und schreiben mit RD und WR

0000'   DB 00           schleife: in a,(0)          ;lesen v. wert'
0002'   D3 00                     out (0),a         ;ausgabe des  wertes
0004'   C3 0000'                  jp schleife      ;wiederholung

                                  end
```

Abb. 4.5.9 Kombinierte Ein-Ausgabe

ersten 5 Zellen. Bei diesem Test kann mit einem zweiten Kanal auf dem Scop geprüft werden, ob zur gleichen Zeit wie das -IORQ-Signal auf 0 ist, auch -RD auf 0 liegt.

17. *Abb. 4.5.8* zeigt das Programm zum Testen des Ausgabe-Befehls. Es muß dabei zur gleichen Zeit wie -IORQ auf 0 liegt -WR auf 0 liegen. Mit einem Kanal sieht man, daß -WR pulst, was beim vorherigen Programm nicht der Fall war.

18. *Abb. 4.5.9* zeigt ein Programm in dem sowohl eine Eingabe als auch Ausgabe erfolgt. Mit dem Scop betrachtet sieht man zwei aufeinander folgende -IORQ Pulse. *Abb. 4.5.10* zeigt das Impulsdiagramm. Mit einem zweiten Kanal läßt es sich überprüfen.

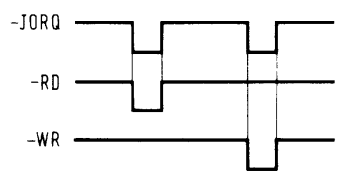


Abb. 4.5.10 Zeitdiagramm beim Testprogramm

```
                        ; Testprogramm  fuer Z80-CPU Karte SBC
                        ; pulsen MRQ und WR,RD bei allen
                        ; Speicheradressen

                                                    ;startwert unwichtig
0000'   7E              schleife: ld a,(hl)         ;data holen
0001'   77                        ld (hl),a         ;zurueckschreiben
0002'   23                        inc hl            ;next adresse
0003'   C3 0000'                  jp schleife      ;wiederholung

                                  end
```

Abb. 4.5.11 Testprogramm für Z80-CPU-Karte SBC

```
                        ; Testen einer Speicherzelle ob
                        ; Schreiblesezugriff ok ist

0000'   21 1000                 ld hl,1000h     ;adresse der zu testenden
                                                ;zelle
0003'   7E                      ld a,(hl)       ;holen eines zufallwertes
0004'   2F                      cpl             ;1er komplement
0005'   77                      ld (hl),a       ;zurueckschreiben
0006'   BE                      cp (hl)         ;lesen u. vergleichen
0007'   C2 0015'               jp nz,fail      ;nicht gleich dann fehler
000A'   2F                      cpl             ;gegenprobe
000B'   77                      ld (hl),a
000C'   BE                      cp (hl)         ;mit komplement wert
000D'   C2 0015'               jp nz,fail      ;doch nicht ok

0010'   DB 00           gut:    in a,(0)        ;messen IORQ pulse da
0012'   C3 0010'               jp gut          ;=gut fall

0015'   76              fail:   halt            ;HALT-signal = fehler

                                end
```

Abb. 4.5.12 Schreiblesezugriff

19. *Abb. 4.5.11* zeigt ein Programm für den Speicherzugriff. Mit dem ersten Befehl LD A, (HL) wird von einer Speicherzelle ein Wert in das Register A der CPU eingelesen. Dieser Wert wird mit dem zweiten Befehl LD (HL), A wieder in die Speicherzelle zurückgeschrieben. Die Adresse der Speicherzelle ist dabei durch den Inhalt des Registerpaars HL bestimmt (siehe auch Software-Kapitel). Anschließend wird das Registerpaar HL um eins erhöht. Dann folgt ein Sprung wieder auf den Anfang des Programms. Der Inhalt des Registerpaars ist nach dem Stromeinschalten undefiniert, doch stört es nicht, bei welcher Zelle der Zugriff beginnt. Nach einer kurzen Zeit wird der gesamte Speicher mit dieser Schleife adressiert. Mit dem Scop kann wieder der Adreßbus angesehen werden, der in Pulsgruppen, die mit aufsteigender Adresse die halbe Frequenz besitzen, alle Speicheradressen durchläuft. Dabei ist es hier nicht so kritisch, daß auch ein Schreibzugriff auf den EPROM-Teil erfolgt.

20. Zum wirklichen Testen der Speicher wird aber ein anderes Programm benötigt. *Abb. 4.5.12* zeigt das Programm. In diesem Programm wird nur eine einzige Zelle auf Adresse 1000H getestet. Dabei wird der Inhalt dieser Zeile geholt, dann das Einerkomplement gebildet und wieder zurückgeschrieben. Nun wird erneut aus der Zelle gelesen und verglichen, ob dieser Wert mit dem eingeschriebenen übereinstimmt. Falls nicht, wird nach FAIL gesprungen, sonst wird der Wert erneut komplementiert und zurückgeschrieben und es erfolgt wieder ein Vergleich. Stimmt es diesmal nicht, so wird wieder nach FAIL gesprungen, sonst nach GUT. Bei der Marke GUT wird eine Schleife durchlaufen, die dauernd Eingabe-Befehle ausführt. Damit kann am Ausgang -IORQ eine Pulsfolge gemessen werden. Wird FAIL angesprungen, so erfolgt die Ausführung des HALT-Befehls und mit dem Scop kann der Ausgang -HALT der CPU gemessen werden, der im Falle des Halt-Befehls auf den Wert 0 geht. Gehen also die RAMs (nur ein sehr grober Test) so muß -IORQ pulsen, gehen sie nicht so muß -HALT auf 0 liegen. Liegt weder

-HALT auf 0 noch pulst -IORQ, so liegt ein anderer Fehler vor, z. B. EPROM falsch gebrannt. Der Gegentest, falls -IORQ pulst, kann leicht durchgeführt werden, indem das RAM einmal herausgezogen wird (Spannung abschalten) und dann der Test wiederholt wird.

Damit ist die CPU im großen und ganzen vorgetestet und das Monitorprogramm im EPROM kann eingesteckt werden. Wenn die anderen Peripheriekarten nicht angeschlossen sind, so muß -IORQ an der CPU pulsen, da nach dem Start eine Peripherieeinheit (Serial Interface oder CRT und KEYBOARD) angesprochen wird. Der restliche Test wird daher in dem Kapitel zusammen mit der jeweiligen Peripherie-Einheit besprochen.

4.6 Vollausbau-CPU

Die CPU-Karte des vorherigen Abschnitts war für die Adressierung von IK Byte RAM und 4K Byte EPROM ausgelegt und konnte darüber hinaus auch nicht erweitert werden. Wer von Anfang an ein größeres System plant und den Mut besitzt, gleich damit anzufangen, dem sei hier eine CPU-Karte gezeigt, die bis auf 64K Byte Speicher aufgerüstet werden kann.

Die CPU-Karte enthält dabei keinerlei Speicher, sondern nur Puffer für Daten, Adreß- und Steuerbus. Als Speicher gibt es extra Karten. Durch die Pufferung kann der Bus viel stärker belastet werden und damit können mehr Karten angeschlossen werden als bei der SBC-Karte.

Abb. 4.6.1 zeigt die Schaltung. Der Quarz-Oszillator ist genauso aufgebaut wei bei dem SBC-Computer, nur daß noch zwei Flip-Flops nachgeschaltet sind. Die Flip-Flops sollen die Frequenz teilen, um unterschiedliche Quarze verwenden zu können.Mit der Brücke J1 kann die Frquenz der CPU eingestellt werden. Dabei wird in der vorverdrahteten Stellung die Frequenz durch 2 geteilt und bei einem 4MHZ Quarz ergibt sich ein Takt von 2MHZ für die CPU. Die CPU kann aber bei Verwendung einer -A Version auch mit 4MHZ betrieben werden, dazu wird ein 8MHZ Quarz beim Oszillator verwendet. Der Takt wird über einen Inverter I6 und I5 gepuffert, einmal zur CPU hin (Pullup-Widerstand R4 nicht vergessen) und dann zum externen Bus hin.

Die Reset-Logik befindet sich bei dieser Karte mit auf der CPU und besteht aus dem Monoflop MV1. Nach dem Stromeinschalten lädt sich der Kondensator C1 über den Widerstand R1 auf. Bei Erreichen des Triggerschwellwertes von MV1 wird es ausgelöst und erzeugt am Ausgang -Q einen kurzen negativen Puls, dessen Zeitdauer von C2 abhängt. Die Pulsdauer sollte relativ kurz gewählt werden, um auch die Verwendung von dynamischen Speichern zu gestatten, da diese bei einem zu langen Reset-Puls (in der Größenordnung 2ms) teilweise gelöscht werden könnten. Das Reset-Signal ist ebenfalls auf den Bus geführt. Eine Taste kann auf der CPU-Karte eingelötet werden und damit kann der RESET auch manuell ausgelöst werden.

Die Steuereingänge -WAIT, –BUSRQ, -NMI, -INT sind über die Widerstände R5, R6, R7, R8 auf ein 1-Signal gelegt. Sie sind aber auch auf den Bus geführt, so daß

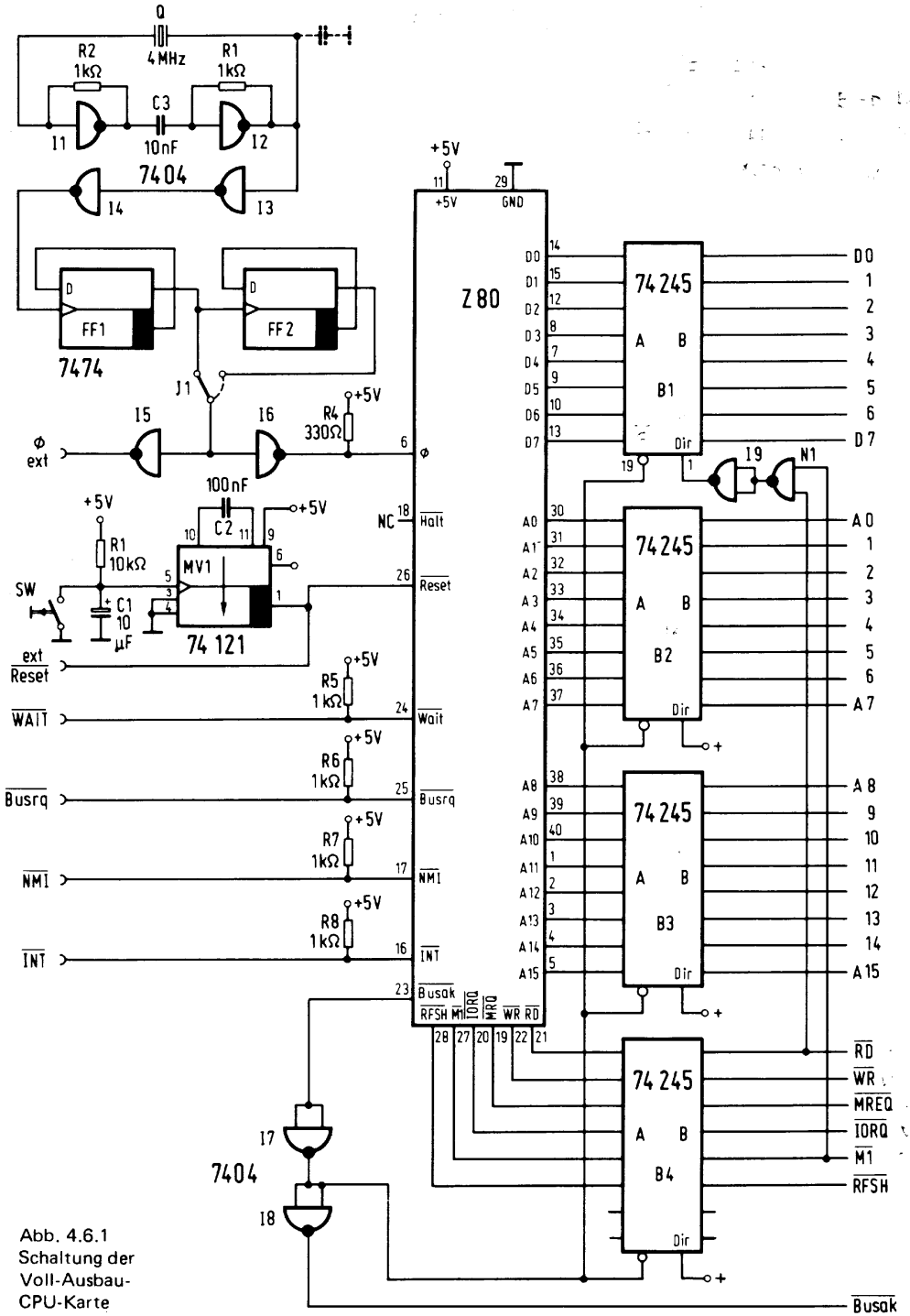


Abb. 4.6.1
Schaltung der
Voll-Ausbau-
CPU-Karte

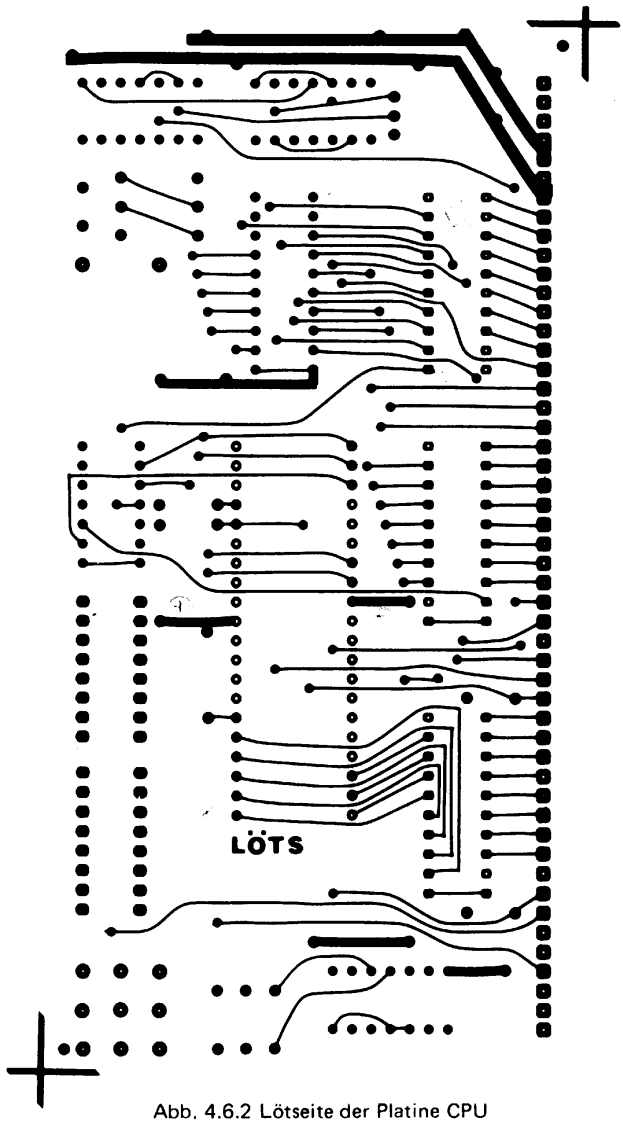


Abb. 4.6.2 Lötseite der Platine CPU

sie von außen aktiviert werden könnnen. Der Datenbus ist über den bidirektionalen Bustreiber B1 mit dem externen Datenbus verbunden. Der Adreßbus ist mit den Treibern B2 und B3 gepuffert und die Steuersignale -RD, -WR, -MREQ, -IORQ, -M1, -RFSH sind mit B4 gepuffert. Das Signal -BUSAK wird von der CPU als Rückmeldung bei einem DMA-Zugriff verwendet. Liegt es auf Low, so darf ein anderer Busteilnehmer auf den Bus zugreifen. Alle Ausgänge der CPU-Karte, außer Takt- und Reset-Signal müssen hochohmig, also im TRI-State-Zutand sein. Das -BUSAK-Signal wird dazu über einen Inverter I7 an die Freigabe-Eingänge aller Bus-Treiber geführt. Liegt der Pegel

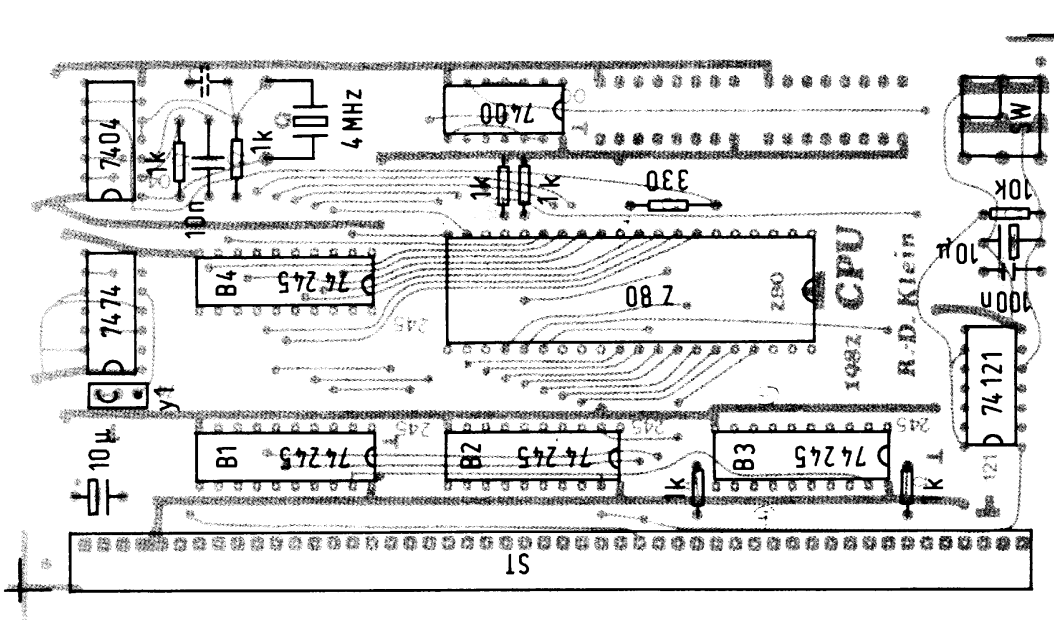


Abb. 4.6.4 Bestückungsplan der CPU-Karte

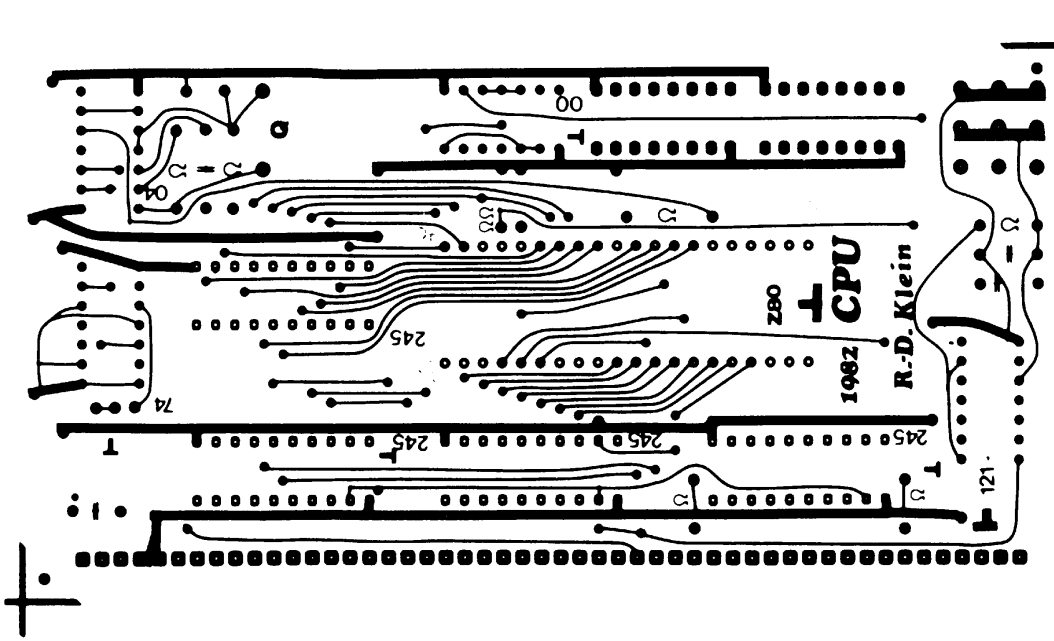


Abb. 4.6.3 Bestückungsseite der Platine CPU

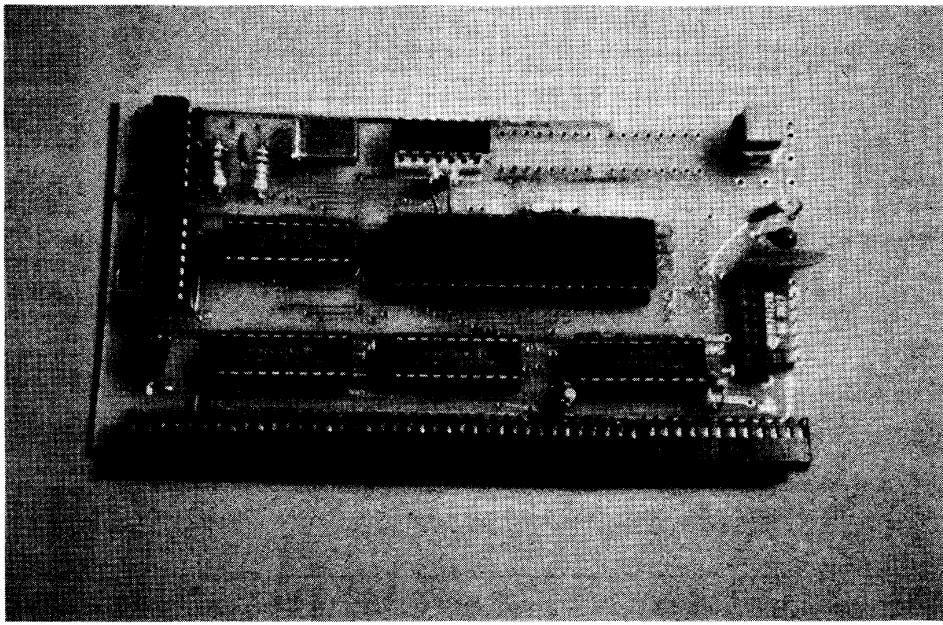


Abb. 4.3 CPU-Karte

von -BUSAK auf Low, so ist der hinter dem Inverter I7 auf High. Ein High (1-Signal) aber sperrt alle Bustreiber und führt sie in den TRI-State-Zutand. Als Bustreiber wurde einheitlich der Gattertyp 74LS245 verwendet. Da es sich um einen bidirektionalen Treiber handelt, wird die Richtung bei den Bausteinen B2, B3 und B4 von der CPU auf den Bus gschaltet und der Dir-Eingang auf ein 1-Signal gelegt. Bei B1 muß eine Richtungssteuerung vorgenommen werden. Dabei wird der Treiber im Normalfall von der CPU in Richtung BUS geschaltet, es gibt nur zwei Fälle in denen eine Umkehrung vorgenommen werden muß. Einmal bei einem allgemeinen Lese-Zugriff – der liegt immer dann vor, wenn das -Rd-Signal auf 0 liegt, dann gibt es einen Speicher oder IO-Lesezugriff. Dann gibt es noch den M1-Zyklus. Liegt das Signal auf 0, so muß ebenfalls in Richtung CPU geschaltet werden. Nun wird der Leser sicher denken, daß der letzte Schritt unnötig sei, da bei einem M1-Zyklus auch ein Lese-Zugriff vorliegt, doch ist dies nicht immer richtig, wie z. B. bei einem Interrupt-Antworte-Zyklus. Dort wird von der CPU ein Interrupt-Vektor gelesen und die Signale -M1 und -IORQ aktiviert. Da dies auch ein Lese-Zugriff ist, muß -M1 auch berücksichtigt werden.

Die Signale -M1 und -RD werden über das Nand-Gatter N1 und den Inverter I9 miteinander verknüpft und zum Richtungs-Eingang geführt. Der DIR-Eingang des Bustreibers B1 liegt immer dann auf 0 wenn -RD oder -M1 auf 0 liegen.

Abb. 4.6.2 zeigt die Lötseite der CPU-Platine und in *Abb. 4.6.3* zeigt die Bestückungsseite die CPU-Platine und in *Abb. 4.6.4* zeigt den Bestückungsplan.

Beim Bestücken werden in alle IC-Plätze Sockel eingelötet und dann alle passiven Bauteile eingelötet. Als Reset-Taste können zwei verschiedene Tasten-Formen eingesetzt

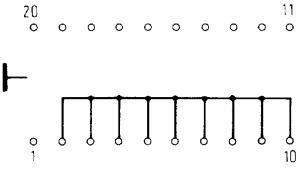


Abb. 4.6.5 Testsockel-Belegung

werden, zum Beispiel der Mini-Digit-Taster. Nun kann bereits mit dem Testen begonnen werden.

1. Messen zwischen 0V und +5V. Es darf kein Durchgang vorhanden sein. Wenn ja, so ist der Kurzschluß zu suchen.

2. Einsetzen des 7404-Bausteins, der den Quarz-Oszillator darstellt. Nach Anlegen der Versorgungsspannungmuß an Pin 8 des Gatters 7404 eine Frequenz von 4 MHZ anliegen, wenn ein 4MHZ-Quarz verwendet wurde (Standard bei 2MHZ CPU-Betrieb).

3. Einsetzen des 7474 (oder wahlweise 74LS74). Nun muß an Pin 6 der CPU ein Takt von 2MHZ anliegen. Die Brücke J1 ist dafür eingestellt.

4. Einsetzen des Bausteins 74121. Damit wird der Reset für die CPU erzeugt. Bei Betätigen des Tasters SW muß an Pin 26 der CPU ein kurzer negativer Puls erscheinen. Der Puls ist weniger als 1ms breit und daher mit dem Scop nur schwer zu sehen.

5. Messen der Pegel an Pin 16, 17, 25, 24 der CPU. Er muß auf +5V liegen. Nun kann die CPU eingesetzt werden.

6. Die Bustreiber sind noch nicht bestückt. Die Adreßleitungen der CPU müssen nun beliebige Signal enthalten. In den Sockel des Bustreibers B1 werden mit Hilfe eines DIL-Steckers (oder einen weiteren Sockels, als Stecker verwendet) Brücken nach Masse gelötet, wie *Abb. 4.6.5* zeigt. Damit wird wie schon beim Test der SBC-Karte gezeigt, ein NOP-Befehl an den Datenbus der CPU gelegt. Die Adreßleitungen müßten nun aufwärts zählen, mit der Ausnahme des Refresh-Zyklus. Die Tendenz ist aber gut zu sehen.

7. Nun können die restlichen Bauteile eingesetzt werden. Der Haupttest kann dann mit Hilfe einer Speicherkarte (siehe nächsten Abschnitt) durchgeführt werden.

4.7 RAM/ROM Karte

Für die Vollausbau-CPU wird ein externer Speicher benötigt. Wir wollen hier eine Karte behandeln, die den Ausbau auf 64K Speicher ermöglicht.

Die Karte kann dabei 16K RAM und/oder EPROM aufnehmen. *Abb. 4.7.1* zeigt die Schaltung. Als RAM-Baustein kann der Typ 6116 oder ein äquivalenter Typ eingesetzt werden. Er besitzt eine Organisation von 2K x 8 Bits. Da das EPROM mit der gleichen Anschlußbelegung arbeitet, kann hier gemischt auch das EPROM 2716 eingesetzt werden. So kann am Anfang (Adresse 0) ein Monitorprogramm stehen und danach RAM-Speicher. Die Karte adressiert 16K Speicher, somit werden für den Vollausbau auf 64K

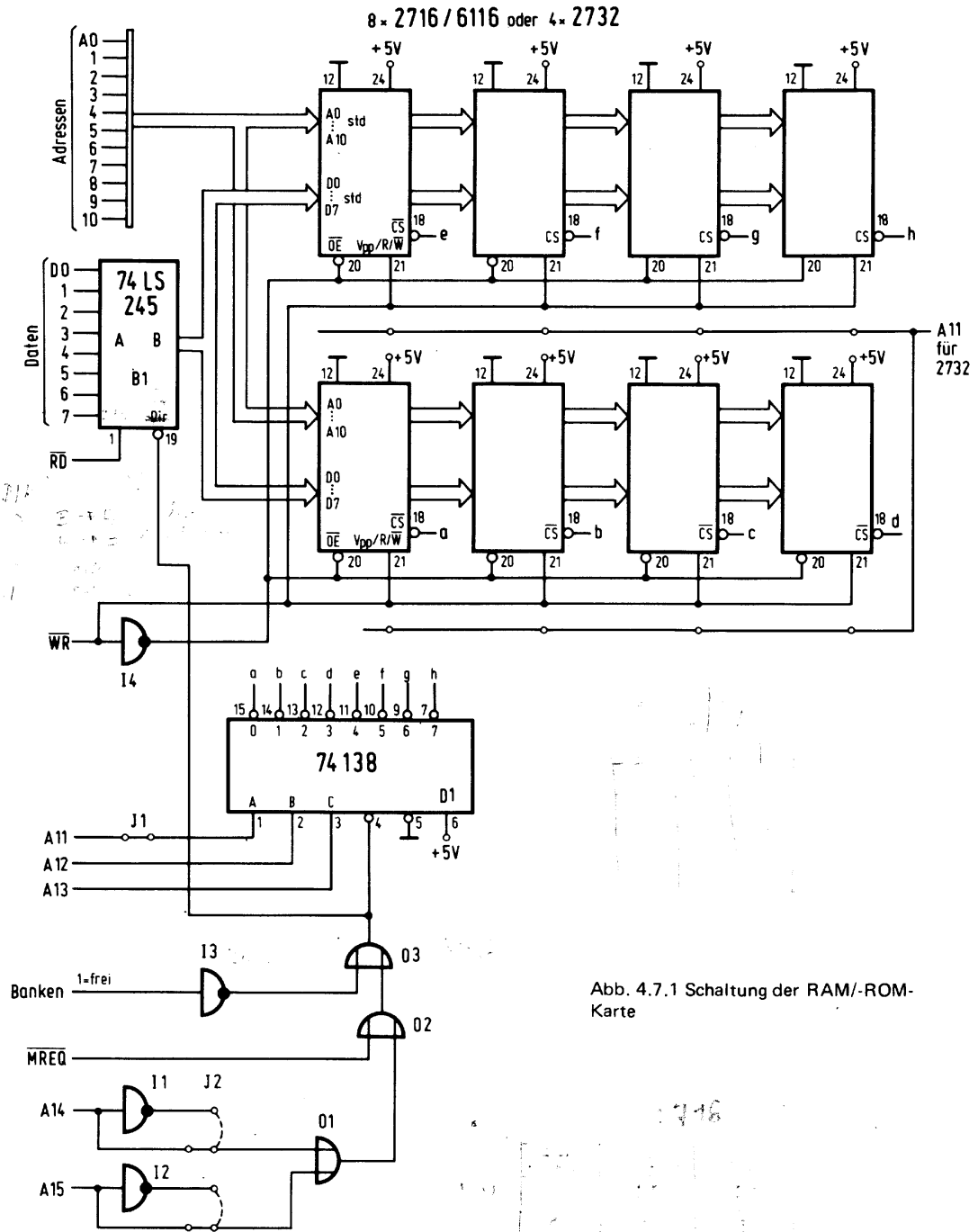


Abb. 4.7.1 Schaltung der RAM/-ROM-Karte

vier solche Schaltungen benötigt. Leider können ohne Änderung der Platine keine 2732-EPROMs verwendet werden, dennoch sind Brücken dafür vorgesehen.

Die Adressen A0 bis A10 sind an alle Speicherplätze geführt, denn sie sind bei beiden Speichertypen (2716/2732 und 6116) an die gleichen Anschlußstifte zu legen. Der Datenbus wird mit dem IC 74LS245 (B1) gepuffert und an den internen Datenbus geführt. Der Bustreiber erhält zwei Signale. Einmal muß bei diesem bidirektionalen Treiber die Richtung umgeschaltet werden können, dies geschieht mit dem Eingang DIR. Und ferner gibt es noch einen Freigabe-Eingang, der den Baustein vom hochohmigen Zustand in den normalen TTL-Arbeitszustand überführt. Die Richtung wird direkt mit dem Signal -RD der CPU gesteuert. Immer wenn dieses Signal auf einem 0-Pegel liegt, wird vom internen Datenbus der Speicher in Richtung CPU getrieben. Liegt das Signal auf 1, so wird in Richtung interner Bus getrieben, um z. B. einen Einschreibevorgang in die RAM-Baustein zu ermöglichen. Der Freigabe-Eingang kommt von der Adreßlogik.

Die Ram-Bausteine wie auch die EPROMs besitzen zwei Freigabe-Leitungen. Einmal -OE und dann noch -CS (oder auch -CE genannt). Mit dem Eingang -OE werden die Ausgabetreiber freigegeben und mit -CS wird der Baustein angewählt. Die Funktion ist ähnlich und nur bei den RAM-Baustein gibt es einen Unterschied. Mit -CS wird der Baustein bei einem Schreibvorgang freigegeben und dabei darf -OE nicht auf 0 gelegt werden, da sonst die Ausgabetreiber des Bausteins einen Kurzschluß verursachen würden. Dies ist aber nicht bei allen RAM-Bausteinen dieses Typs so, bei manchen Versionen ist intern eine Blockierlogik vorgesehen. Um aber alle RAM-Typen verwenden zu können, darf die Auswahl des Bausteins nur mit -CS erfolgen, hingegen ist -OE mit dem Schreibsignal über den Inverter 14 verknüpft, so daß bei einem Schreibzugriff der Pegel dieses Eingangs auf 1 liegt. Das Schreibsignal -WR selbst ist auf Pin 21 geführt, das den R/-W Eingang des RAMs darstellt. Bei Verwendung von EPROMs hat dieses Pin die Bedeutung des VPP-Eingangs, der beim Lesen auf +5V liegen muß; daher arbeiten EPROMs in dieser Schaltung ohne Änderung.

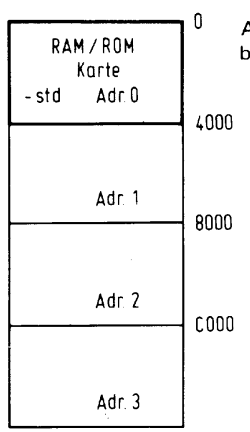


Abb. 4.7.2 Speicherbelegung

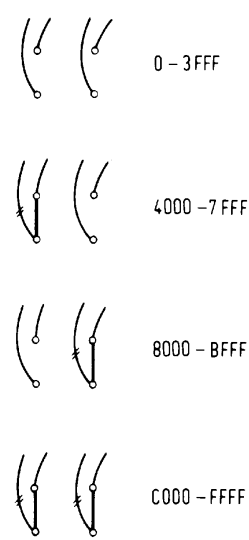


Abb. 4.7.3 Brücken für verschiedene Adreßbereiche

Die Auswahl eines der Speicherbausteine erfolgt mit dem IC 74LS138. Dazu werden an ihn die Adressen A11 bis A13 geführt. Der Freigabe-Eingang des Bausteins ist mit dem Freigabe-Eingang des Bustreibers B1 verbunden und kommt vom Ausgang des Oder-Gatters 03. Dieser Ausgang liegt immer dann auf 0, wenn die Karte adreßmäßig selektiert wurde und ein -MREQ Signal vorliegt, also ein Speicherzugriff durchgeführt wurde. Die Karte besitzt eine Adresse, die mit der Brücke J1 eingestellt wird. Dazu sind die beiden Inverter I1 und I2 vorgesehen.

Abb. 4.7.2 zeigt den Speicherbereich. Die Standard-Einstellung der Brücke bewirkt, daß die Karte im Bereich 0 bis 3FFFh (h steht für HEX oder Sedezimal) angesprochen

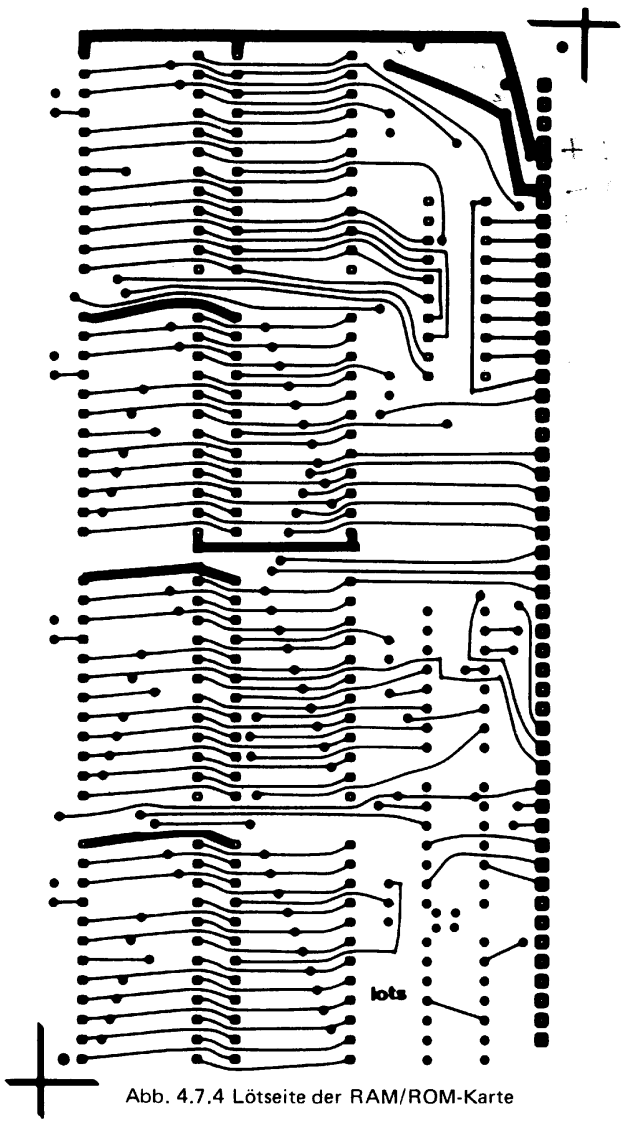


Abb. 4.7.4 Lötseite der RAM/ROM-Karte

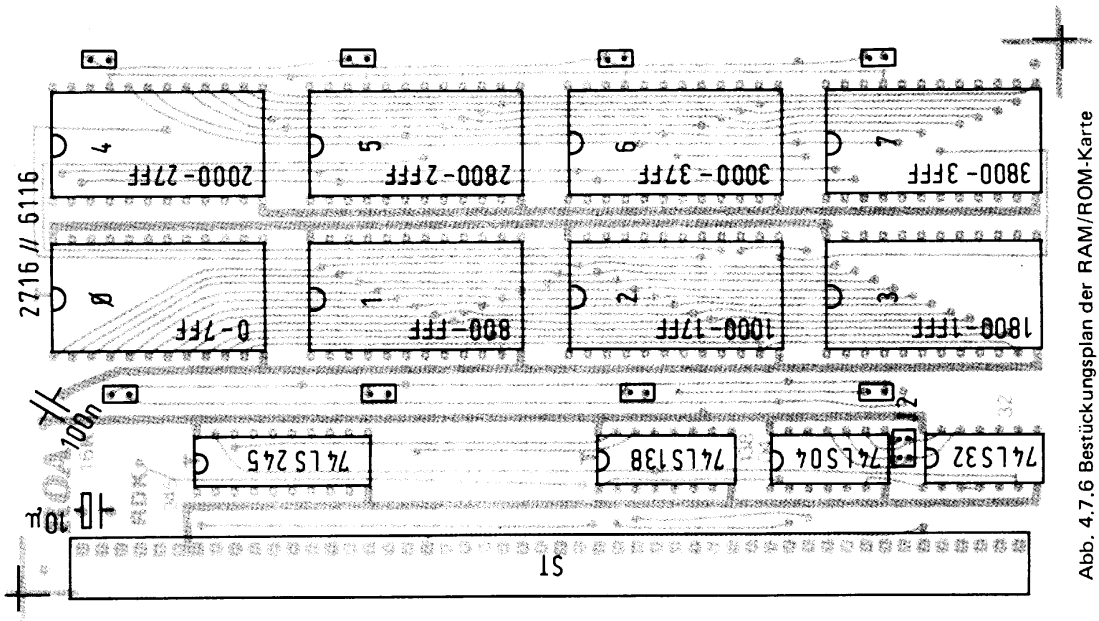


Abb. 4.7.6 Bestückungsplan der RAM/ROM-Karte

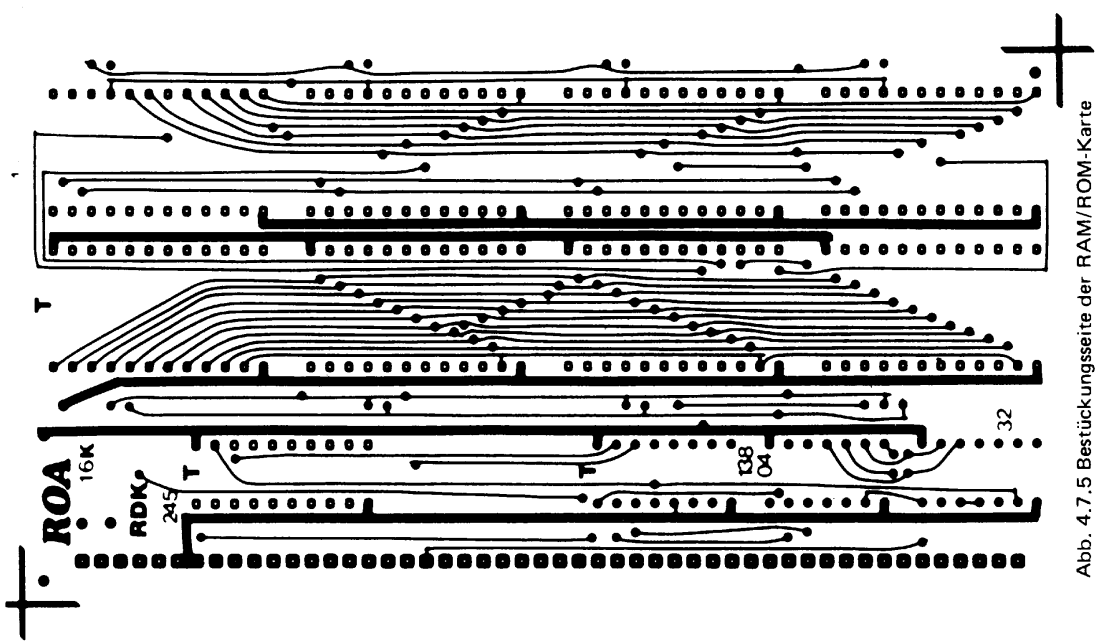


Abb. 4.7.5 Bestückungsseite der RAM/ROM-Karte

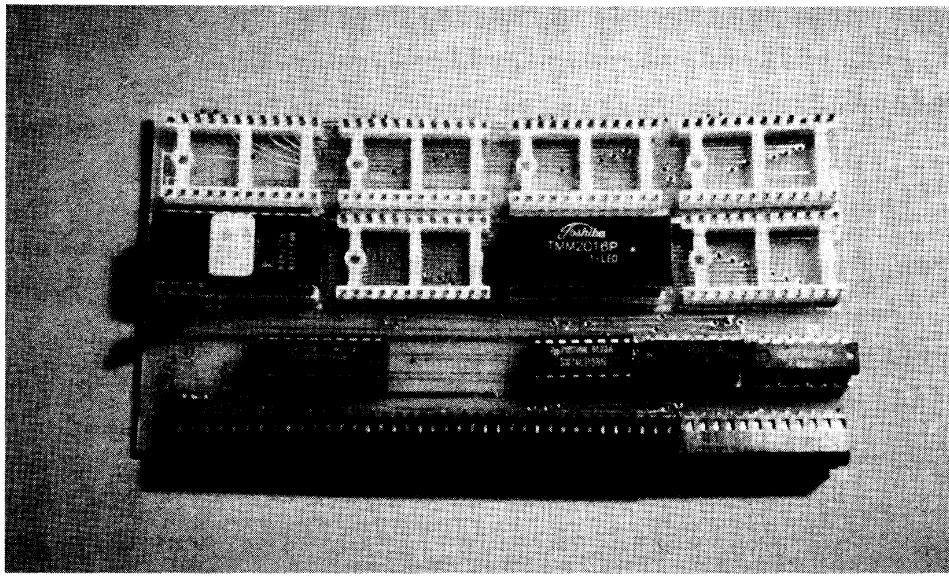


Abb. 4.4 RAM/ROM-Karte

wird. Wird die Brücke bei I1 durchtrennt und umgelötet, so kommt der Speicher im Bereich 4000h bis 7FFFh zu liegen. Wird anstelle von I1, I2 in die Leitung zu 01 gesetzt, so liegt der Bereich von 8000h bis BFFFh. Werden beide Inverter dazwischen geschaltet, liegt der Speicher zwischen C000h und FFFh. Abb. 4.7.3 zeigt nochmals eine Darstellung aller Kombinationen und deren Lage auf der Platine. Um auch mehr als 64K Speicher im Computer verwenden zu können, gibt es hier noch einen Eingang mit den Namen BANKEN. Liegt der Eingang auf 1, so ist die Karte freigegeben, liegt er auf 0, so ist die Karte gesperrt.

Soll zum Beispiel der gesamte Hauptspeicher mit RAM gefüllt werden, also 64K RAM verwendet werden, so ist es nötig zumindest beim Start des Rechners irgendwo einen EPROM-Bereich zu haben, um den Rechner überhaupt zur Ausführung eines Programms zu bringen. Wie das möglich ist, wird im nächsten Abschnitt genauer gezeigt. Nun zu der Platine. *Abb. 4.7.4* zeigt die Lötseite der Karte, *Abb. 4.7.5* die Bestückungsseite. In *Abb. 4.7.6* ist der Bestückungsplan abgedruckt. Dort ist auch die Lage der Adreßbereiche ersichtlich.

Beim Aufbau der Platine wird nach bewährten Verfahren wieder mit dem Bestücken der Sockel begonnen. Alle ICs müssen in Sockel untergebracht werden, um überhaupt vernünftig Fehler suchen zu können. Dann wird ein Tantal-Kondensator von 10 μF und ein keramischer Kondensator mit 100 nF eingelötet. Nun kann der Test beginnen. Dazu werden die ICs 74LS138, 74LS04 und 74LS32 eingesteckt. Der Bustreiber wird nicht eingesteckt.

1. CPU und die RAM Karte werden über den Bus miteinander verbunden. Der Sockel aus Abb. 4.7.1 wird wieder benötigt. Er liefert den NOP-Befehl an die CPU und diese zählt damit permanent. Der Sockel wird in die Fassung des 74LS245 gesteckt. *Abb. 4.7.7*

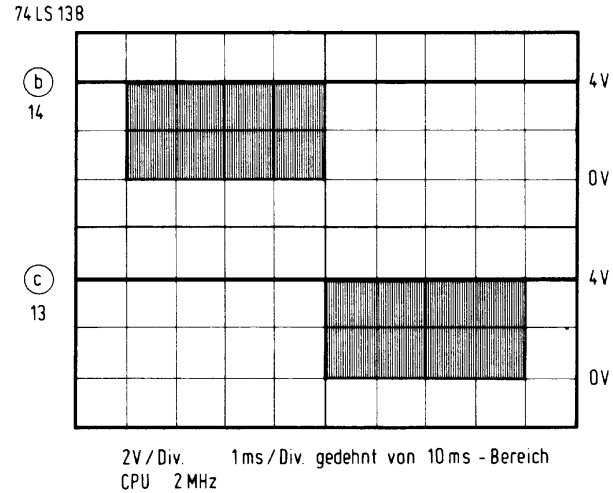


Abb. 4.7.7 Oszillogramm RAM/ROM-Karte

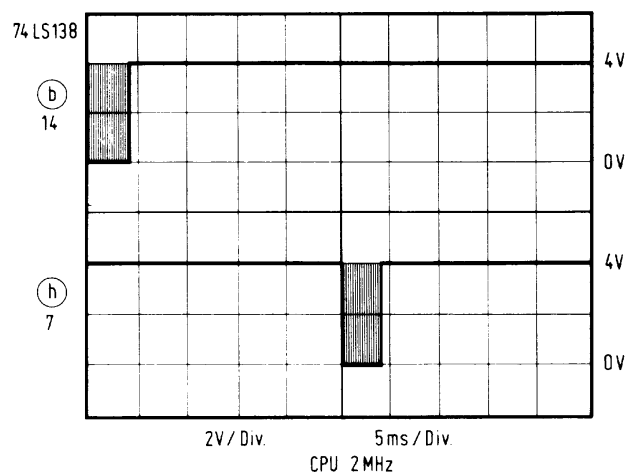


Abb. 4.7.8 Oszillogramm RAM/ROM-Karte

zeigt ein Oszillogramm. Pin 14 des ICs 74LS138 wurde mit Kanal 1 und Pin 13 mit Kanal 2 verbunden. Die einzelnen Ausgänge 0 bis 7 adressieren jeweils einen Teilbereich des Speichers und werden daher jeweils für 2048 Zugriffe angesprochen, dann wird der nächste Ausgang aktiviert. An Pin 15 ist dieses Verhalten nicht so schön zu erkennen, so dort auch ein Zugriff bei einem REFRESH-Zyklus statt findet. Es zeigt sich daher dort ein kontinuierliches Impulsmuster. In *Abb. 4.7.8* ist ein weiteres Impulsdiagramm dazu abgebildet. Diesmal bei den Pins 14 und 7 des Dekoders 74LS138.

2. Der Rest der Speicherkarte kann nun nach dem Einsetzen des Bustreibers 74LS245 und der Speicher genauso getestet werden, wie im Abschnitt der SBC-Karte mit den Testprogrammen dargestellt wurde. Sobald das Monitorprogramm läuft, können die Speicher mit Hilfe des Monitors getestet werden. Siehe dazu auch die entsprechende Software-Kapitel.

4.8 Bankselekt-Karte mit RAM/ROM

Der Z80-Prozessor kann „nur" 64K Speicher adressieren,da er einen 16-Bit breiten Adreßbus besitzt. Sollen mehr als 64K Speicher verwendet werden, so muß eine Zusatzlogik dafür sorgen, daß weitere Adreßleitungen zur Verfügung stehen. Es gibt aber noch einen anderen Grund um eine solche Banklogik zu verwenden. Die Z80-CPU beginnt nach einem RESET bei der Adresse 0 mit der Ausführung des Programms. Dort muß also ein ROM-Bereich sein, da sofort ein ausführbares Programm vorhanden sein muß. Schön ist es aber, im Bereich 0 RAM zu haben um z. B. Programme dort austesten zu können, denn im 0-Bereich befinden sich ein paar ausgezeichnete Adressen, wie 0, 8, 10 h..38 h, 66h, die eine besondere Bedeutung haben. Ebenfalls gibt es sehr viele käufliche Programme (z. B. Betriebssystem CP/M) das ein RAM im unteren Bereich fordert. Daher wird eine BANK-SELEKT-Logik verwendet. Damit ist es möglich, beim Stromeinschalten in den Bereich 0 ein EPROM-Gebiet zu legen und durch einen besonderen Befehl, nämlich einfach ein Zugriff auf einen IO-Port kann dann der EPROM-Bereich gegen RAM ausgetauscht werden. Dabei wird zuvor per Programm der EPROM-Bereich in einen nicht benutzten RAM-Bereich kopiert und dort dann beim Umschalten weitergearbeitet.

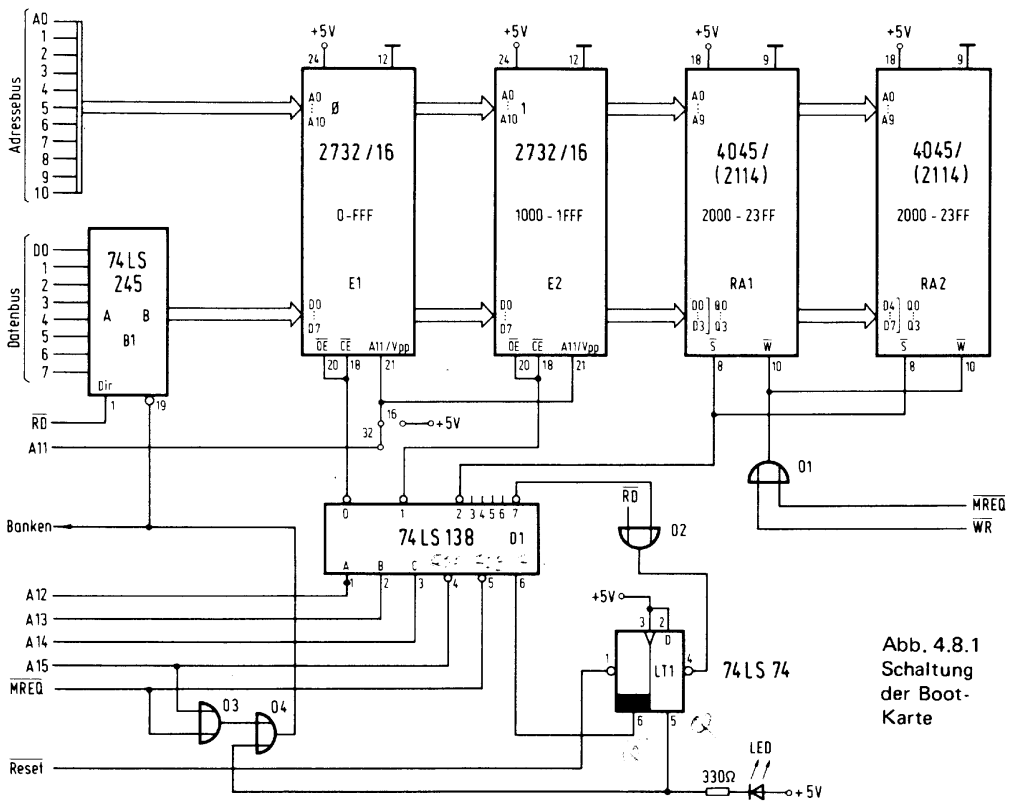


Abb. 4.8.1 Schaltung der Boot-Karte

Eine einfache BANK-SELEKT-Logik, mit der erreicht werden kann, von Adresse 0 weg RAM zu haben, wird im folgenden besprochen. *Abb. 4.8.1* zeigt die Schaltung. Im Schaltbild sind zwei EPROMs mit je 4K Bytes Speicher vorhanden, sowie ein RAM-Speicher mit 1K, der zusätzlich verwendet werden kann um die Karte auch ohne eine weitere Speicherkarte betreiben zu können. Die Auswahl einer weiteren Speicherkarte geschieht über das Signal BANKEN. Es wird an alle anderen Speicherkarten geführt, die für einen Adreßbereich von 0 bis FFFFh dekodiert sein können. Die Leitung BANKEN liegt immer dann auf einem 1-Signal, wenn die Karten freigegeben sind. Liegt es auf 0, so werden alle am Bus liegenden restlichen Speicherkarten desaktiviert. Damit ist BANKEN praktisch eine neue Adreßleitung A16.

Das Signal BANKEN liegt immer dann auf 0, wenn A15 auf 0 liegt, -MREQ auf 0 liegt und außerdem der Ausgang Q des Flip-Flops LT1. Das Flip-Flop LT1 ist der Kern der BANK-SELEKT-Logik. Es wird durch das Signal -RESET beim Stromeinschalten oder nach einem manuellen Reset auf einen definierten Zustand gesetzt und zwar so, daß der Ausgang Q auf 0 liegt und damit die BANK-Logik aktiv ist. Damit wird immer im unteren Bereich 0 bis 7FFFh die Speicherlogik auf der Karte angesprochen und der außenstehende Bereich durch das Signal BANKEN gesperrt. Das Flip-Flop kann nun auch zurück gesetzt werden, nämlich dann, wenn der EPROM-Bereich von der CPU nicht mehr benötigt wird. Das geschieht durch einen Lesezugriff auf irgend eine Zelle im Adreßbereich 7000h bis 7FFFh. Dazu ist der Ausgang 7 des Adreßdekoders D1 an den einen Eingang von 02 geführt, das zusätzlich mit dem Signal -RD verknüpft ist. Erscheint am Ausgand des Oder-Gatters 02 ein negativer Puls, so wird das Flip-Flop über den Eingang RESET (Pin-4) geführt und der Ausgang Q nimmt den Wert 1 an.

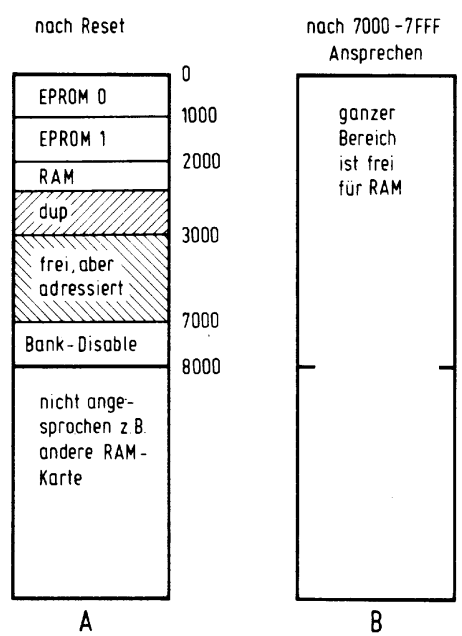


Abb. 4.8.2 Speicherbereich

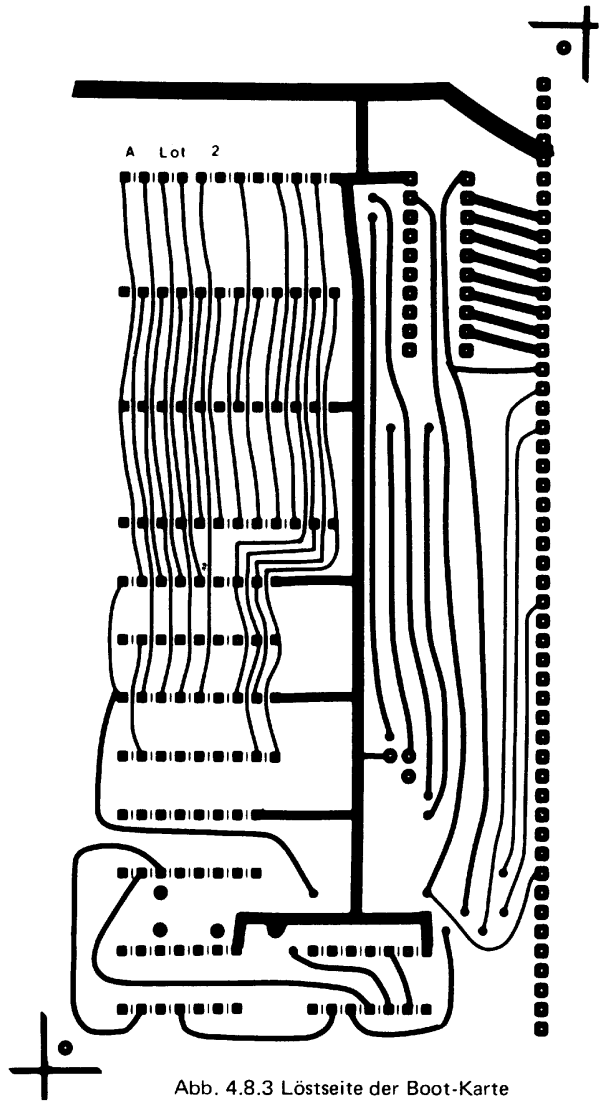


Abb. 4.8.3 Löstseite der Boot-Karte

Über den Ausgang -Q wird der Dekoder für alle Zeiten gesperrt, und auch das Flip-Flop läßt sich nur über einen RESET zurücksetzen. Der EPROM Breich wird aber nur am Anfang gebraucht, um die CPU überhaupt nach dem Start mit einem Programm zu versorgen. Der Monitor kann dann z. B. auch in diesem ROM stehen und muß dann natürlich vor dem Umschalten in einem RAM-Bereich außerhalb des Bank-Logik-Bereichs (8000-FFFFh) kopiert werden. Es kann auch in einem anderen Bereich ebenfalls ein EPROM zur Verfügung stehen.

Abb. 4.8.2 zeigt die Speicherbelegung. Nach dem RESET gilt die Belegung nach A und nach einem Lese-Zugriff auf den Bereich 7000h bis 8000h gilt die Belegung nach B. Der

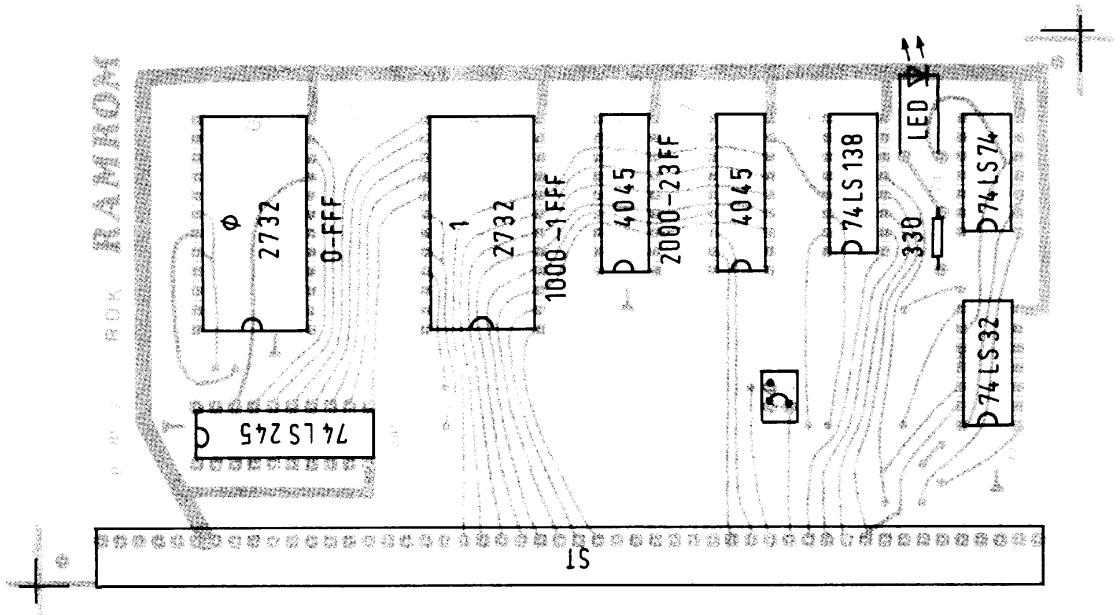


Abb. 4.8.5 Bestückungsplan der Boot-Karte

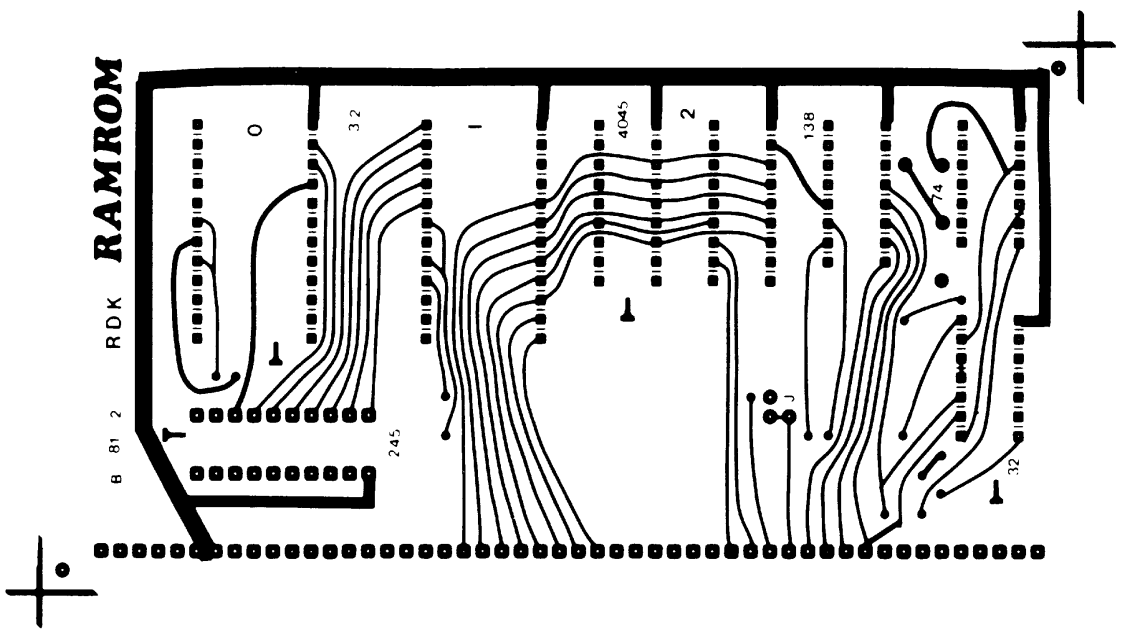


Abb. 4.8.4 Bestückungsseite der Boot-Karte

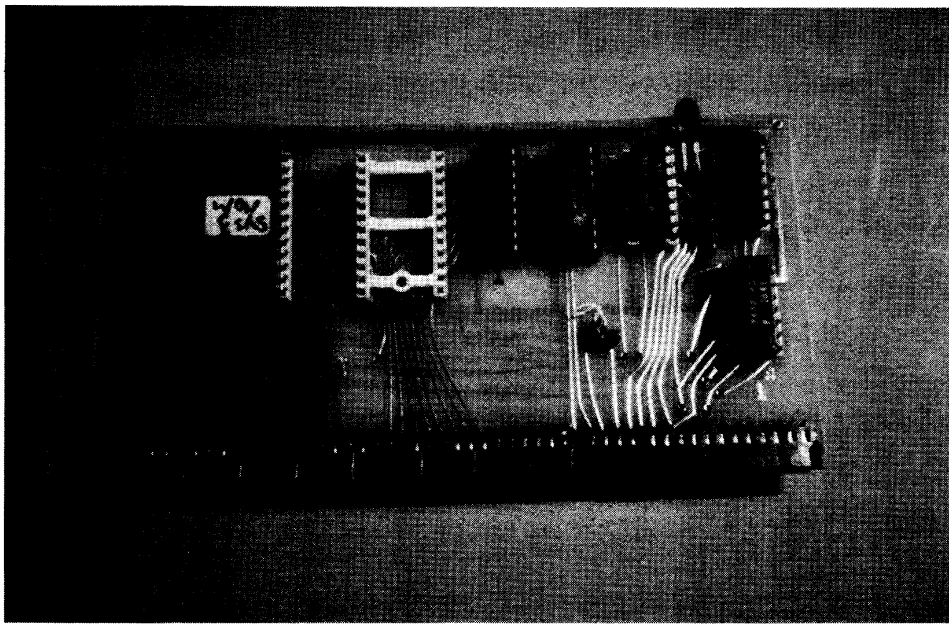


Abb. 4.5 Boot-Karte

RAM-Speicher liegt dabei von Adresse 2000h bis 23FFh. Da die Adressierung nicht eindeutig ist, wird der Bereich bis zu der Adresse 2FFF zyklisch wiederholt, daß heißt auf Adresse 2400h, 2800h und 2C00h liegt dieselbe Speicherzelle wie auf Adresse 2000h. Wird also z. B. auf Adresse 2000h der Wert 55h geschrieben, so ist der Wert auf Adresse 2400h ebenfalls 55h. Dies stört aber nicht weiter und bewirkt eine Schaltungsvereinfachung.

Nun zum Schaltungsaufbau. *Abb. 4.8.3* zigt die Lötseite der Platine, *Abb. 4.8.4* zeigt die Bestückungsseite und *Abb. 4.8.5* zeigt den Bestückungsplan.

Zunächst wird an allen IC-Stellen ein Sockel eingelötet. Dann wird der Widerstand für die LED eingelötet und schließlich die LED selbst.

Test der Schaltung

1. Betrieb mit der CPU-Karte und der unbestückten RAMROM-Karte mit der Bootlogik. Messen aller Versorgungsspannungen an der Karte.

2. Der Betriebstest wird etwas komplizierter, da sich der NOP-Text nicht anwenden läßt, würde die CPU nur NOP-Befehle erhalten, so würde in kurzer Zeit der Bereich 7000h erreicht und bei einem Lesezugriff dorthin wird bekanntlich der Speicherbereich ausgeblendet und man kann nicht mehr weiter testen. Daher wird eine andere Sequenz benötigt.

```
                        ;*********************************
                        ;* Test Boot-Karte RAMROM        *
                        ;*********************************

0000'   C3 0000'        start:  jp start        ;endlos
```

Abb. 4.8.6 Testprogramm Boot-Karte

Es wird in einem EPROM ein kurzes Programm nach *Abb. 4.8.6* abgelegt. Dies ist ein Sprungbefehl auf sich selbst. Dazu wird der Bustreiber B1, Dekoder D1 sowie LT1 und das Oder-Gatter in die Sockel bestückt. Nach einem RESET des Prozessors muß das Prom angesprochen werden und es ergibt sich ein Oszillogramm nach *Abb. 4.8.7*. Dort ist das Selekt-Signal des EPROMs und der M1-Cyklus dargestellt. Beim M1-Signal wird der Opcode (c3H) des Sprungbefehls geholt. Dann folgt ein Refresh-Cyklus, der hier zwar keine Bedeutung besitzt, aber auch zu einem Zugriff auf das EPROM führt. Mit zwei weiteren Selektvorgängen wird der Operand des Sprungbefehls, also die Adresse 00 00 geholt. Der Ausgang 0 von LT1 muß dabei auf einem 0-Signal liegen und an dessen Takteingang dürfen keine Pulse anliegen. Sollte es nicht funktionieren, so ist zu überprüfen, ob bei einem RESET-Vorgang am Pin 1 des Flip-Flop der RESET-Puls ankommt.

3. Nun können die restlichen Sockel überprüft werden. Dazu zeigt *Abb. 4.8.8* ein weiteres Programm. Es liest zyklisch aus der Speicherzelle 1000h um die Dekodierung des EPROMS 1 zu testen und ferner schreibt es in die Zelle 2000h abwechselnd eine 0 und liest, und schreibt ein 0ffh und liest. Damit kann mit dem Scop auch das RAM überprüft werden. Als RAM-Bausteine kann der Typ 4045 (Texas) oder 2114 (Intel) verwendet werden. *Abb. 4.8.9* zeigt das Oszillogramm für den EPROM-Test. Das Selekt-Signal des zweiten EPROMS (EPROM 1) ist im oberen Bereich abgebildet. Unten ist das M1-Signal dargestellt. Es treten jetzt 8 M1-Signale auf, bevor sich das Ganze wiederholt, da das Programm auch aus acht Befehlen besteht, bevor eine Programmwiederholung erfolgt. Bei

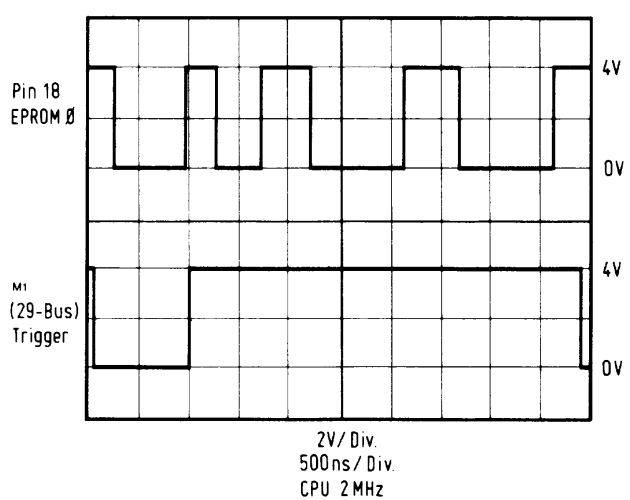


Abb. 4.8.7 Oscillogramm Boot-Karte

```
                        ;*********************************
                        ;* Test Boot-Karte RAMROM 2      *
                        ;*********************************

0000'                   start:
0000'   3A 1000                ld a,(1000h)    ;lesen EPROM 1
0003'   3E 00                  ld a,0          ;0 laden
0005'   32 2000                ld (2000h),a    ;0 nach speicher
0008'   3A 2000                ld a,(2000h)    ;wieder laden
000B'   3E FF                  ld a,0ffh       ;alles ff
000D'   32 2000                ld (2000h),a    ;ff nach speicher
0010'   3A 2000                ld a,(2000h)    ;und wieder laden
0013'   C3 0000'               jp start        ;alles wiederholen
```

Abb. 4.8.8 Testprogramm Boot-Karte

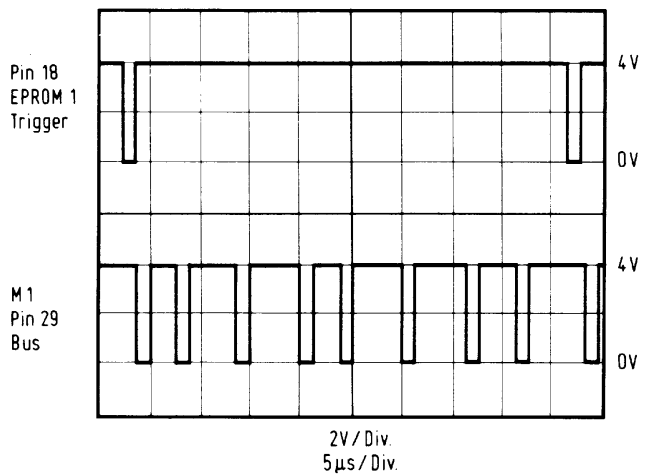


Abb. 4.8.9
Oszillogramm
Boot-Karte

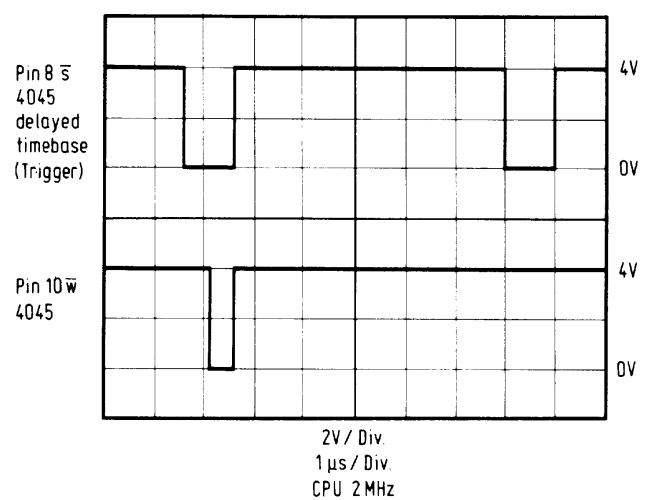


Abb. 4.8.10
Oszillogramm
Boot-Karte

der Eingabe des Programms in ein EPROM ist zu beachten, daß in unseren Listings bei Adressen immer die lesbare Form gewählt ist, also erst der höherwertige Teil abgebildet und dann der niederwertige. Im Speicher erfolgt die Ablage aber UMGEKEHRT. Also erst niederwertige Adresse und dann höherwertiger Teil. Beispiel:

Steht im Listing

C3 1233

so muß im Speicher

C3 23 12 stehen.

Adressen erkennt man am Listing immer daran, das sie als 16 Bit-Größe dargestellt werden und zusammengeschrieben sind, also 1223. Steht dagegen 23 12 so ist der Code genauso abzulegen wie er dasteht.

In *Abb. 4.8.10* ist das erste Oszillogramm für den RAM-Test abgebildet. Es wird mit dem ersten Kanal an Pin 8 des RAMs gemessen und mit dem zweiten Kanal an Pin 10. Um die Zeitverhältnisse zu verdeutlichen, ist hier mit einer Ausschnittsvergrößerung gearbeitet worden und nur ein Schreib-Lesevorgang dargestellt. Man erkennt, daß das -W-Signal etwas später erscheint als das Selektsignal. Insgesamt gibt es zwei Schreibvorgänge im gesamten Ablauf. *Abb. 4.8.11* zeigt ein weiteres Oszillogramm. Dort wurde nun eine Datenleitung (hier D3) zusammen mit dem Selekt-Signal der RAMs abgebildet. Man erkennt, daß beim Einschreiben des Wertes 1 kurz ein Buskonflikt vorliegt, da der Pegel der Datenleitung für die Hälfte der Zeit auf halbem Niveau liegt und erst in der letzten Phase auf 1 geht. Das kommt daher, weil das -W-Signal erst später erscheint. Der Buskonflikt schadet den RAMs aber nicht, da sie für diesen Betriebsfall ausgelegt sind. Die anderen Datenleitungen sehen ähnlich aus, doch zwischen den Zugriffen auf das RAM ergeben sich Unterschiede. Am externen Datenbus stehen die Signale besser aus. Dort ist der Bus-

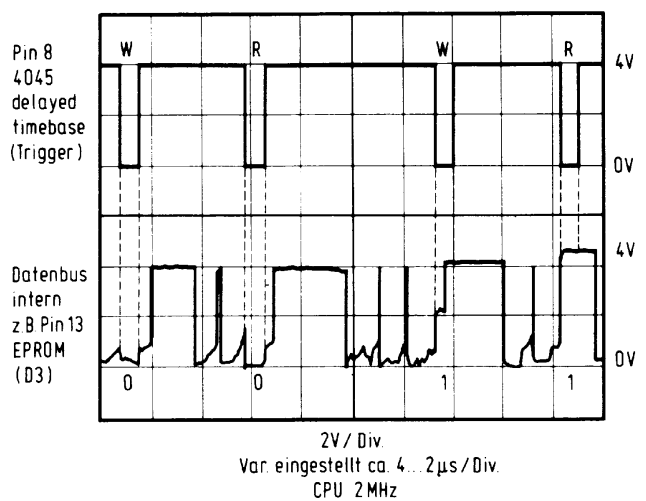


Abb. 4.8.11 Oszillogramm Boot-Karte

```
                                ;**********************************
                                ;* Test Boot-Karte RAMROM         *
                                ;* Test der Bootlogik             *
                                ;**********************************

0000'                           start:
0000'   21 FFFF                        ld hl,0ffffh    ;erst mal warten
0003'   00                      loop:   nop             ;verzoegerung
0004'   00                              nop
0005'   00                              nop
0006'   2B                              dec hl          ;hl veringern
0007'   7D                              ld a,l          ;pruefe ob hl=0
0008'   B4                              or h            ;beide l,h=0
0009'   C2 0003'                       jp nz,loop      ;nein dann nochmals
000C'   3A 7000                        ld a,(7000h)    ;BANK AUS
000F'   76                              halt            ;nicht mehr wirksam
```

Abb. 4.8.12 Testprogramm Boot-Karte

konflikt nicht erkennbar. Zum vollständigen Test müssen alle internen und externen Datenleitungen betrachtet werden.

4. Test der Boot-Logik. Dazu ist in *Abb. 4.8.12* ein kurzes Programm abgedruckt. Es hat die Aufgabe, nach einer kurzen Verzögerung den Bereich auszublenden. Dazu ist am Anfang eine Warteschleife vorhanden. Nach dem RESET leuchtet die LED zunächst für ca. 2 Sekunden und erlischt dann, wenn die Schaltung richtig arbeitet.

Damit ist die Boot-Logik-Karte getestet.

4.9 Fragen zum Kapitel 4

1. Warum ist es nicht sinnvoll, große Abläufe mit der in Abb. 4.1.2 gezeigten Schaltungsart zu realisieren?
2. Warum sind in Abb. 4.1.4 die Kathoden der Dioden mit den Ausgängen des Dekoders verbunden und nicht die Anoden?
3. Wozu wird der Ladeeingang in Abb. 4.1.6 benötigt?
4. Könnte das Befehlsformat auch anders aussehen als in Abb. 4.1.7 Vorschläge falls ja.
5. Wieviele Ausgänge müßte der Dekoder in Abb. 4.2.1 bei 10 Adresseneingängen besitzen?
6. Warum können die Flip-Flops LT1 bis LT4 in Abb. 4.2.2 nicht weggelassen werden und was würde dann passieren?
7. Warum liegen die Eingänge E1 bis E7 in Abb. 4.2.3 auf Masse?
8. Was passiert, wenn der Programmierpuls beim EPROM-Rechner länger als 50ms ist?
9. Was versteht man unter der Zugriffszeit bei RAMS? Siehe auch Abb. 4.2.9.

10. Stellt die Blockschaltung nach Abb. 4.3.1 einen brauchbaren Computer dar?

11. Der Z80 besitzt 16 Adreßleitungen. Wieviel Speicher kann er damit rein rechnerisch adressieren?

12. Wozu gibt es beim Z80 die Unterscheidung zwischen einem Speicherzugriff und einem Peripherie-Zugriff und durch welche beiden Signale können sie von einer externen Peripherie unterschieden werden.

13. Wieviele Peripherie-Einheiten kann der Z80 adressieren?

14. Wozu dient der -WAIT-Eingang des Z80s?

15. Warum genügt es nicht nur die Adreßleitung A10 einmal nichtinvertiert und einmal invertiert an die Selektleitung des EPROMs und des RAMs in Abb. 4.5.1 anzuschließen? Warum muß es noch mit -MREQ verknüpft werden?

16. Wozu dient der Widerstand R3?

17. Warum muß das Signal -WR nicht noch mit -MREQ verknüpft werden, bevor es an den -W-Eingang des RAMs gelangt?

18. In Abb. 4.6.1 wird der Dir-Eingang des Bustreibers B1 durch eine Verknüpfung der beiden Signale -RD und -M1 gesteuert, warum genügt es nicht nur eines von beiden Signalen zu verwenden?

19. Wozu dient der Eingang BANKEN in Abb. 4.7.1.

20. Warum ist es nötig, Speicherbanken anzulegen?

5 Peripherie-Geräte

Bisher hatten wir uns nur um den Prozessor und seine unmittelbare Umgebung wie RAM und ROM gekümmert. Der Computer muß aber auch mit seiner Außenwelt in Verbindung treten können, um sinnvolle Aufgaben verrichten zu können. Dazu wird eine Schnittstelle zwischen Computer und Außenwelt benötigt. *Abb. 5.1* verdeutlicht dies. Als Schnittstelle kann bei einem Computer ein Datensichtgerät, Drucker, oder früher Lochkarten, Lochstreifen etc. dienen. Doch auch schon Datensichtgeräte und Drucker können zur Außenwelt gezählt werden. Dann kann man im Prinzip zwei verschiedene Schnittstellen unterscheiden!

1. Parallele Schnittstelle. Der Computer ist über eine Reihe von Leitungen mit der Außenwelt verbunden. Jede dieser Leitungen kann eine andere Bedeutung haben oder einer Gruppe zugeordnet sein, wie in dem Beispiel der Verkehrsampel-Steuerung aus dem vorherigen Kapitel. *Abb. 5.2* zeigt eine schematische Darstellung.

2. Serielle Schnittstelle. Es gibt eine Sonderform der Verbindung mit der Außenwelt, die als serielle Schnittstelle bezeichnet wird. Dabei werden die Daten seriell, wie der Name ja schon sagt, über Leitungen übertragen. *Abb. 5.3* zeigt ein Beispiel. Eine Datenleitung dient als Verbindung vom Computer zur Außenwelt und eine andere dient der Verbindung von der Außenwelt in Richtung Computer. Diese Schnittstelle wird immer dann verwendet, wenn eine nicht allzu hohe Übertragungsrate verlangt wird, oder eine große Entfernung überbrückt werden soll. Eine niedrige Datenrate ist zum Beispiel bei einem Datensichtgerät ausreichend. Der Mensch gibt dabei Befehle über eine Tastatur an den Computer, dabei kann nicht sehr schnell eingegeben werden. Daher genügt eine niedere Datenrate. Wenn der Computer eine Ausgabe auf den Bildschirm eines Datensichtgerätes macht, so muß der Mensch dies auch noch lesen können und daher reicht es ebenfalls, eine niedere Datenrate zu verwenden. Praktisch alle Datensichtgeräte haben daher eine serielle Verbindungsmöglichkeit. Bei Druckern sieht das anders aus. Hier soll möglichst schnell ausgegeben werden. Doch liegt die Begrenzung in der Druckmechanik selbst. Bei langsamen Druckern (10 Zeichen pro Sekunde bis 200 pro Sekunde) und schnelleren (200 bis 600 Zeichen pro Sekunde) kann immer noch eine serielle Schnittstelle verwen-

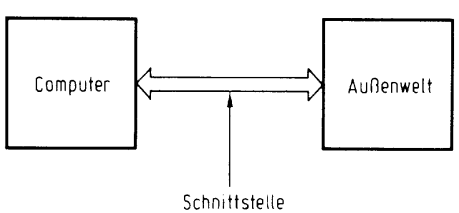


Abb. 5.1 Schnittstelle Computer-Außenwelt

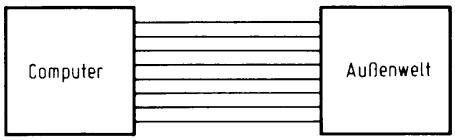


Abb. 5.2 Parallele Verbindung

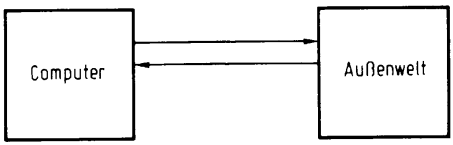


Abb. 5.3 Serielle Verbindung

det werden. Bei Schnell- und Schnellstdruckern wird aber die parallele Schnittstelle bevorzugt. Ein Laserdrucker z. B. kann bis zu 200 Druckseiten pro Minute ausstoßen, dort sind besondere schnelle Datenkanäle nötig.

5.1 Serielles Interface

Unser Computer besitzt noch kein Interface. Daher soll als nächstes eine Karte mit einem seriellen Interface besprochen werden. Danach ist es möglich, ein handelsübliches Datensichtgerät oder eine Datensichtgerätkarte an den Computer anzuschließen, um mit ihm zu arbeiten. In einem weiteren Abschnitt wird aber auch eine solche Karte beschrieben, so daß, falls kein Datensichtgerät beschafft werden kann, das serielle Interface zunächst nicht benötigt wird.

Zunächst etwas über die Grundlagen der seriellen Übertragung. Wir haben gelernt, daß die Daten seriell, daß heißt zeitlich aufeinanderfolgend über eine Leitung übertragen werden. Wie aber geschieht dies nun genau:

Es gibt dafür in der Praxis eine Vielzahl von Möglichkeiten; wollen aber nur die gebräuchlichste verwenden.

Soll ein 8 Bit-Datenwort übertragen werden, so wird dies in einzelne Bits zerlegt, die dann eins nach dem anderen auf die serielle Leitung geschaltet werden. Beim Empfänger müssen die einzelnen Bits wieder zu einem Datenwort zusammengesetzt werden. Dazu ist es aber nötig zu wissen, wann das erste Bit übertragen wurde und wie lange jeweils ein Bit auf der Leitung ist. Dafür gibt es ein Standardformat. In *Abb. 5.1.1* ist das Format abgebildet. Übertragen wird mit einer festen Bitrate, die auch als BAUDRATE bezeichnet wird. Es ist dies eine Frequenz und errechnet sich bei einer Breite t der einzelnen Bits zu f = 1 / t. Haben die einzelnen Bits eine Breite von 3.3 ms so ergibt sich eine Baudrate von 300 Baud. Um dem Empfänger mitzuteilen, wann die Übertragung stattfindet, gibt es am Anfang ein sogenanntes Startbit. Dieses Startbit trägt keine Datenin-

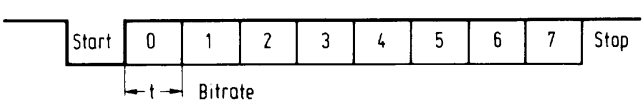


Abb. 5.1.1 Format eines seriellen Bitstromes

formation und dient nur dazu, dem Empfänger den Beginn einer Datenübertragung mitzuteilen. Danach folgen die Datenbits von 0 bis 7 in unserem Fall. Begonnen wird mit dem niederwertigsten Datenbit. Nach dem Daten folgen Stop-Bits, um einen Abstand zwischen dieser Übertragung und einer eventuell gleich anschließenden Übertragung zu garantieren. Die Anzahl der Datenbits reicht von 5 Bits, bei alten Fernschreibern, bis zu 8 Bits, um eine Textübertragung mit dem ASCII-Codesatz zu ermöglichen. Es kann nach den Datenbits auch noch ein Paritätsbit folgen, das zusätzlich übertragen wird. Dieses Paritätsbit ist die Quersumme über die Datenbits: Damit kann beim Empfänger geprüft werden, ob ein Fehler bei der Übertragung vorlag.

Beispiel GERADE (even) Parität bei 7 Datenbits + 1 Paritätsbit

1 0 1 1 0 1 1 1

Oder bei UNGERADER (odd) Parität

1 1 0 1 0 1 0 1

Gerade oder ungerade Parität wird nach Konvention verwendet und muß im Sender und Empfänger übereinstimmen. Werden 8 Datenbits und Parität übertragen, so sind 9 Bits über die Leitung zu transportieren, wobei ein Startbit und ein oder zwei Stop-Bits hinzukommen. Es können ein oder zwei Stop-Bits ebenfalls per Konvention gewählt werden. Die Gesamtübertragungsrate errechnet sich dann wie folgt:

f/((1 *START + n*DATEN + k*PARITÄT + 1*STOP))

Beispiel:
8 Datenbits keine Parität 1 Stop Bit, 1200 BAUD

1200/(1+8+0+1) Zeichen/sek

= 120 Zeichen/sek

Es können dann 120 Zeichen pro Sekunde übertragen werden.

Wie kann z. B. der Sender aussehen. *Abb. 5.1.2* zeigt ein Schema. Es wird z. B. ein 11 Bit-Schieberegister verwendet. Das Schieberegister erhält über den Takteingang direkt die Baudrate, also bei 1200 BAUD entsprechend 1200 Hz. Die zu übertragenden Dten werden an die Paralleleingänge B bis I angeschlossen, hier sollen 8 Datenbits, keine Parität und zwei Stop-Bits verwendet werden. Das Start-Bit wird durch eine Massenverbindung an den Eingang A erzeugt und die beiden Stop-Bits werden durch J, K and +5V erzeugt.

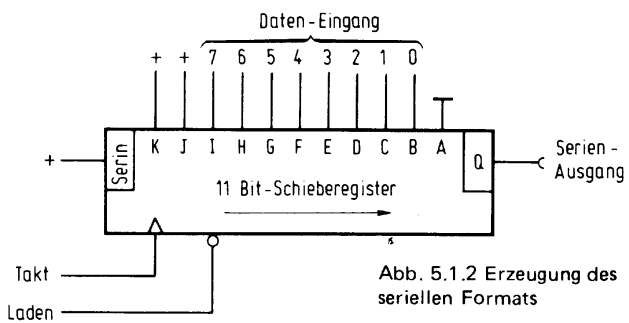


Abb. 5.1.2 Erzeugung des seriellen Formats

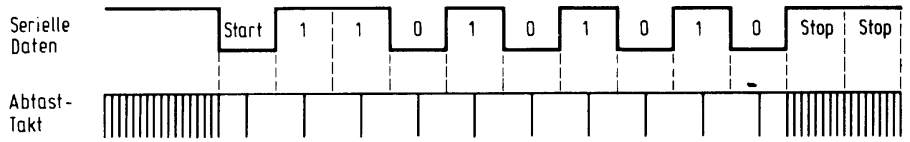


Abb. 5.1.3 Abtastung des Signals beim Empfänger

Wird nun ein Ladepuls an den Lade-Eingang des Schieberegisters gelegt, so beginnt die Übertragung. Nach beendeter Übertragung befinden sich im Schieberegister lauter 1-Werte, die über den Eingang Ser des Schieberegister laufend nachgeschoben werden. Soll ein neues Datenwort übertragen werden, so muß mit einem externen Zähler festgestellt werden, ob die Übertragung des vorherigen Datenwortes schon beendet ist. Bei einem Ladevorgang des Schieberegisters kann dieser externe Zähler z. B. anfangen, bis auf 11 zu zählen. Ist der Wert erreicht, so kann ein neuer Ladevorgang stattfinden, der auch den Zähler wieder rücksetzen müßte.

Wie kann das serielle Signal wieder zurückgewonnen werden? Dazu *Abb. 5.1.3*. Im oberen Bildteil ist das serielle Signal aufgetragen unten der sogenannte Abtasttakt. Mit dem Abtasttakt wird jeweils der Wert auf der Datenleitung in ein internes Register übernommen. Der Abtasttakt kann z. B. der 16-fache Wert der Baudrate sein. Nun beginnt der Vorgang: Sobald das Startsignal erscheint, also eine 0 auf der Leitung liegt, wird der Abtasttakt umgeschaltet, als erstes wird nun nach der Zeit t/2 eine erneute Abtastung durchgeführt, ist dort die Leitung immer noch auf 0, so liegt tatsächlich ein Startsignal vor. Wenn nicht, wird alles ignoriert und die Abtastung mit t/16 fortgesetzt. Ist aber das Startsignal da, so kann nach der Zeit t erneut abgetastet werden und das erste Datenbit kann in ein Schieberegister geschoben werden. Die restlichen Datenbits, oder ggf. Paritätsbits werden ebenfalls in das Schieberegister eingetragen. Nun erfolgen die Stop-Bits. Dort wird geprüft, ob die Leitung auf 1 liegt, tut sie das nicht, sondern liegt sie noch auf 0, so wurde ein Übertragungsfehler erkannt, der auch den Namen FRAMING ERROR besitzt. Denn er tritt auch auf, wenn z. B. im Sender die Anzahl der Datenbits + Paritätsbits größer als die der im Empfänger eingestellten ist. Das Übertragungsverfahren wird auch als asynchrones Übertragungsverfahren bezeichnet, da Sende- und Empfangstakt starr miteinander verbunden sind. Es gibt auch ein synchrones Übertragungsverfahren, daß aber die zusätzliche Übertragung der Taktfrequenz benötigt, die entweder über eine getrennte Leitung oder durch ein Modulationsverfahren mit den Daten übertragen werden muß. Mit synchroner Übertragung können im allgemeinen höhere Baudraten verwendet werden, doch ist das Verfahren aufwendiger und für unsere Zwecke nicht von Nutzen.

Zum Glück müssen wir aber Sender und Empfänger nicht selbst aus einzelnen Gattern zusammenbauen, denn es gibt fertig integrierte Bausteine, die all dies beinhalten. Da ist eine Vielzahl dieser Bausteine mit unterschiedlichen Eigenschaften. Aus dieser Vielzahl heraus verwenden wir einen besonders komfortablen, den Baustein 6551. *Abb. 5.1.4* zeigt die gesamte Schaltung des seriellen Interface. Hier wird nun erstmals eine Einheit nicht über den Speicheradressenraum angesprochen, sondern über die Peripherieadressen. adressiert.

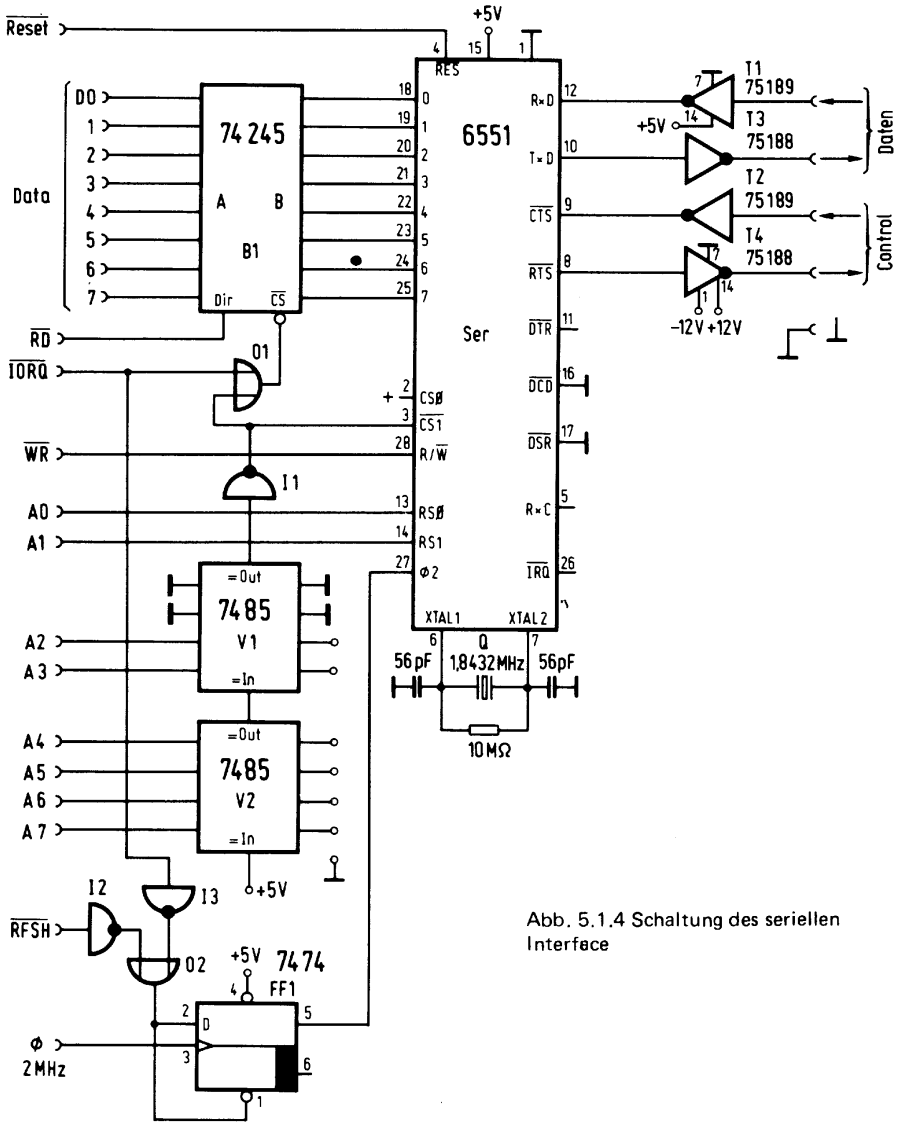


Abb. 5.1.4 Schaltung des seriellen Interface

Der Datenbus ist mit dem bidirektionalen Bustreiber 74LS245 gepuffert. Der Richtungseingang Dir ist dabei wie auch bei den Speicherschaltungen mit -RD verbunden, denn bei einem Lesezugriff soll die Datenrichtung von der Karte auf den Bus zeigen. Der Baustein B1 wird immer dann freigegeben, wenn einmal das Signal -IORQ vorliegt, da es sich um einen Peripheriezugriff handeln soll; zum Andern muß die Karte adressiert sein. Die Auswahl des Adreßbereichs geschieht mit Hilfe des Vergleiches 74LS85 (V1 und V2). Der Baustein 6551 benötigt selbst zwei Adreßleitungen für vier interne Register, so daß die

Adressen A0 und A1 schon reserviert sind. Die Adressen A2 bis A7 können dagegen zur Adressierung hinzugezogen werden. A8 bis A15 sind hier ohne Bedeutung, da bei der Adressierung von Peripheriegeräten beim Z80 nur 256 Adressen verwendet werden. A2 bis A7 gelangen daher an die Vergleicher. An den gegenüberliegenden Eingängen der Vergleicher kann mit Hilfe von Brücken eine Vergleichsadresse eingestellt werden. Wird keine Brücke nach Masse eingesetzt, so reagiert die Karte auf die Adressen FC, FD, FE und FF. Diese Adressen verwenden wir auch im Monitorprogramm. Am Gleich-Ausgang von V1 erscheint immer dann ein 1-Signal wenn die Adressen von A2 bis A7 mit der Vergleichsadresse übereinstimmen. Dieses ist sowohl bei Peripherie- als auch bei Speicherzugriffen der Fall. Daher wird über 01 die Verknüpfung mit -IORQ durchgeführt; nun reagiert die Schaltung nur noch auf Peripherieadressen. Der Ausgang hinter I1 wird aber direkt an den Eingang -CS1 des seriellen Bausteins geführt und die Verknüpfung mit -IORQ geschieht dort indirekt über die Leitung PHI2 (Pin 26). Das Signal -WR wird direkt an den Eingang R/-W des Bausteins 6551 geführt und gibt an, wann ein Schreibzugriff vorliegt. Das Signal PHI2 ist leider nicht direkt mit einem Signal des Z80-Busses verbindbar, da der Baustein ursprünglich für 6502-Systeme entworfen wurde. Eine kleine Anpaßschaltung bestehend aus FF1, 01, I22 und I33 erzeugt das passende Signal. Über I3 wird auch das -IORQ-Signal wieder in die Schaltung eingeführt. Das Signal -RFSH wird hier nicht zum Refresh von Speichern benötigt, erfüllt aber eine ähnliche Aufgabe. Es liegt permanent vor und zwar immer zwischen IO-Zugriffen. Durch die Schaltung zur Erzeugung von PHI2 wird ein nahezu kontinuierlicher Takt für den Baustein 6551 erzeugt, der aber bei Zugriffen auf den Baustein das Signal -IORQ trägt.

Der Serielle Baustein benötigt einen eigenen Quarz, der die Frequenz 1.8432 MHZ erzeugen muß. Diese krumme Frequenz wird benötigt, um intern standardisierte Baudraten erzeugen zu können. Die Baudrate ist im übrigen programmierbar.

In unserer Schaltung kommen neben dem seriellen Ausgang TxD und dem seriellen Eingang RxD auch noch weitere Ein- und Ausgänge vor. Der Eingang -CTS kann eine Übertragung verhindern, wenn er auf einem 1-Signal liegt (Pin 9). Damit ist es möglich, außen ein langsames Gerät anzuschließen, das in der Lage ist, die Übertragung für kurze Zeit anzuhalten, um die eingetroffenen Daten z. B. zu verarbeiten. Dies trifft z. B. bei manchen Datensichtgeräten zu, bei der Durchführung des Bildschirmlöschbefehls. Umgekehrt kann der Baustein über die Leitung -RTS verlangen, daß die Übertragung bei dem anderen Gerät gestoppt wird. Die Leitung wird dazu mit dem anderen Gerät an dem -CTS-Eingang verbunden. Zwischen den Geräten liegen aber noch die Bausteine T1, T2, T3 und T4. Sie haben die Aufgabe den Pegel von 0V bis 5V auf ein Niveau +12V bis -12V anzuheben. Auf diesen Spannungspegel hat man sich geeinigt und er garantiert die Übertragung auch über große Entfernungen (km Bereich). Die Bausteine 75188 (1488) und 75189 (1489) übernehmen die Pegelwandlung. Der Sender 75188 wird dazu mit +/-12V versorgt. Beim Empfänger ist dies nicht nötig. Das Signal -RTS kann per Software aktiviert werden. Der Baustein besitzt noch eine Reihe weiterer Ein- und Ausgänge, wie -DTR, -DCD, -DSR die aber für uns hier nicht weiter von Bedeutung sind.

Der Ein-Ausgang RxC kann eine externe Baudrate aufnehmen, oder dient als Ausgang der internen, je nach Programmierung des seriellen Bausteins. Der Ausgang -IRQ ist ein Interrupt-Ausgang, den wir jedoch ebenfalls nicht verwenden werden.

RS1	RS0	Schreiben	Lesen
0	0	Sende Daten	Empfangsdaten
0	1	Software Reset	Status Register
1	0	Command Register	
1	1	Control Register	

Abb. 5.1.5 Registerbelegung beim Baustein 6551

Der Baustein besitzt wie schon erwähnt mehrere interne Register. *Abb. 5.1.5* zeigt ihre Bedeutung. Wird das Register 0 angesprochen, so wird das Datenregister aktiviert. Register 0 bedeutet in unserem Fall aber die Adresse 0FCh, wenn die Brücken an Vergleicher offen sind. Wird in das Register 0 etwas geschrieben, so werden die Daten in das Senderegister übertragen und seriell über die Leitung geschickt. Beim Empfang kommen die Daten im

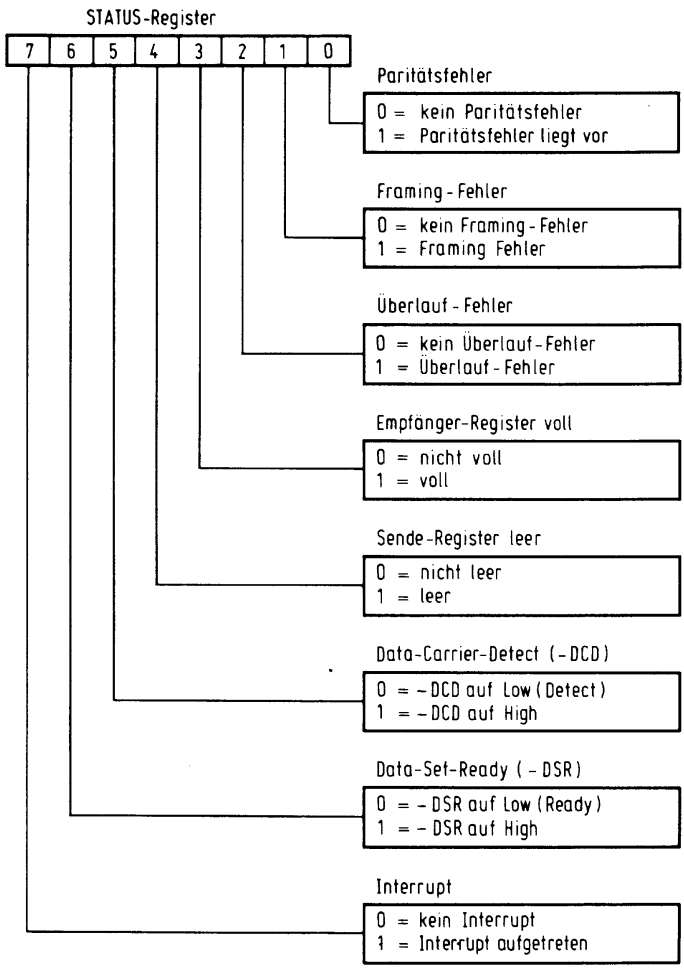


Abb. 5.1.6 Belegung des Status-Register

Empfangsregister an, das ebenfalls über die Adresse 0FCh erreichbar ist, diesmal aber bei einem Lese-Zugriff. Nun genügen diese Register allein natürlich nicht, um einen Datenübertragung durchführen zu können. Zum einen müssen wir wissen, ob die abgesandten Daten vollständig aus dem Senderegister übertagen wurden, um einen nächsten Datenwert eingeben zu können und dann müssen wir wissen, ob Daten vom Empfangsteil angekommen sind. Dies ist mit Hilfe des Status-Registers möglich, das über die Adresse 0FDh (Register 1) ausgelesen werden kann. *Abb. 5.1.6* zeigt die Bedeutung der einzelnen Bits im Status-Register. Für uns sind zunächst die beiden Bits 4 und 3 interessant, die anderen beschreiben den Zustand bei Fehlern oder den Zustand von den Spezialleitungen wie -DSR und -DCD und Interruptmeldungen. Bit 4 ist genau dann auf 1, wenn das Senderegister leer ist und ein neues Zeichen in ds Register 0 geschrieben werden kann, welches dann über die Leitung übertragen wird. Bit 3 des Status-Registers ist genau dann auf 1, wenn ein Datenwort empfangen wurde und in Register 0 bereitsteht. Wird Register 0 ausgelesen, so geht dieses Bit solange wieder auf 0 zurück, bis ein neues Datenwort empfangen wurde. Wird in Register 1 geschrieben, so wird der serielle Port 6551 rückgesetzt, ähnlich als ob er einen physikalischen Reset über die Leitung -RESET empfangen hätte. Dabei ist es egal, welcher Wert in Register 1 geschrieben wurde. Die restlichen Register dienen der Voreinstellung von Parametern und Betriebsart des Bausteins. *Abb. 5.1.7* zeigt die Bedeutung des Bits des Control-Registers 3 (Adresse OFFh bei uns). Dort wird die Baudrate programmiert, die Wortbreite eingestellt und die Anzahl der Stop-Bits be-

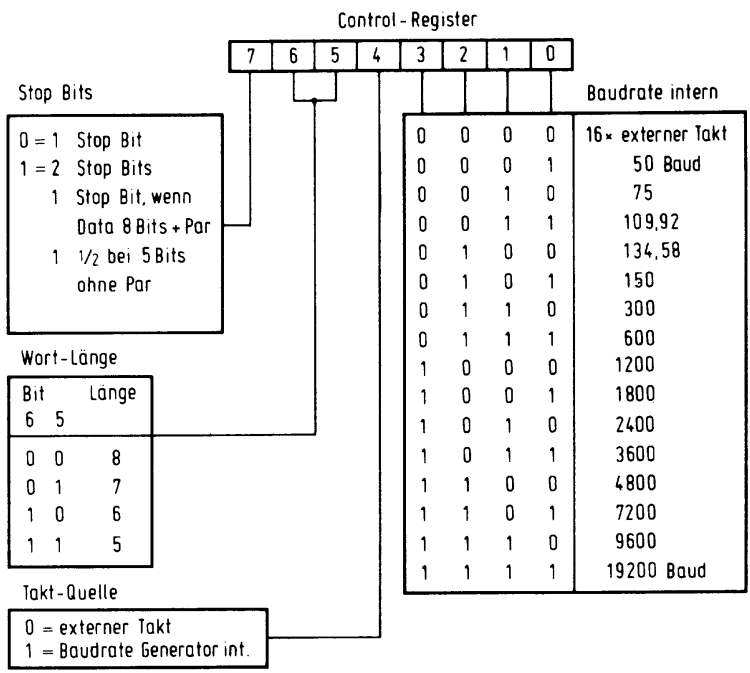


Abb. 5.1.7 Control-Register

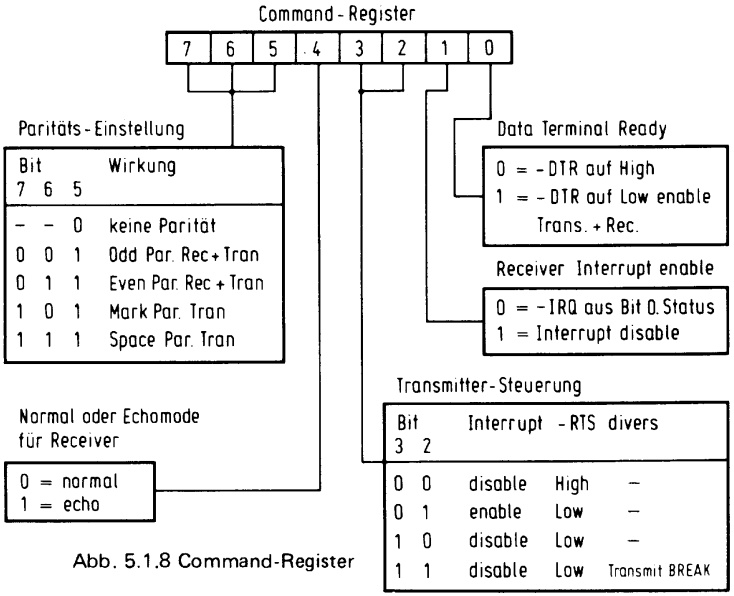


Abb. 5.1.8 Command-Register

Dezimal	Hex	ASCII	dez	hex	asc	dez	hex	asc	dez	hex	asc
0	00	NUL	32	20		64	40	@	96	60	'
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1 XON	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3 XOFF	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	↑	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	

Abb. 5.1.9 Der ASCII-Satz

stimmt. Wird eine Baudrate von 9600 Baud verwendet und 8 Bits mit einem Stop-Bit übertragen, so ergibt sich das Befehlsbyte: 00011110b oder 1Eh.

Im Command-Register *Abb. 5.1.8* kann bestimmt werden ob ein Paritätbit verwendet wird und wenn ja welche Art es sein soll. Die Begriffe Even und Odd haben wir schon kennengelernt, Mark bedeutet das ein fester Wert 1 angenommen wird und Space, daß der Wert 0 verwendet wrid. Dies ist keine echte Parität, sondern es wird nur das eine Bit bei der Übertragung mit verwendet. Die restlichen Bits dienen der Einstellung von -RTS, von Interrupts und der Freigabe der Übertragungskanäle. Wir verwenden keine Parität und -RTS und -DTR liegen auf Low. Damit ergibt sich als Steuerwort 00001011b oder 0Bh. Es wird außerdem der Normal-Mode mit Bit 4 eingestellt, da der Echo-Mode hier nicht verwertbar ist.

Wir hatten vorher schon einmal kurz den Begriff ASCII verwendet, hier soll erklärt werden was darunter verstanden wird. *Abb. 5.1.9* zeigt eine Umrechnungstabelle. Jedem darstellbaren Zeichen des ASCII-Satzes (nach ISO-Norm, DIN 66003) ist ein Wert zuge-

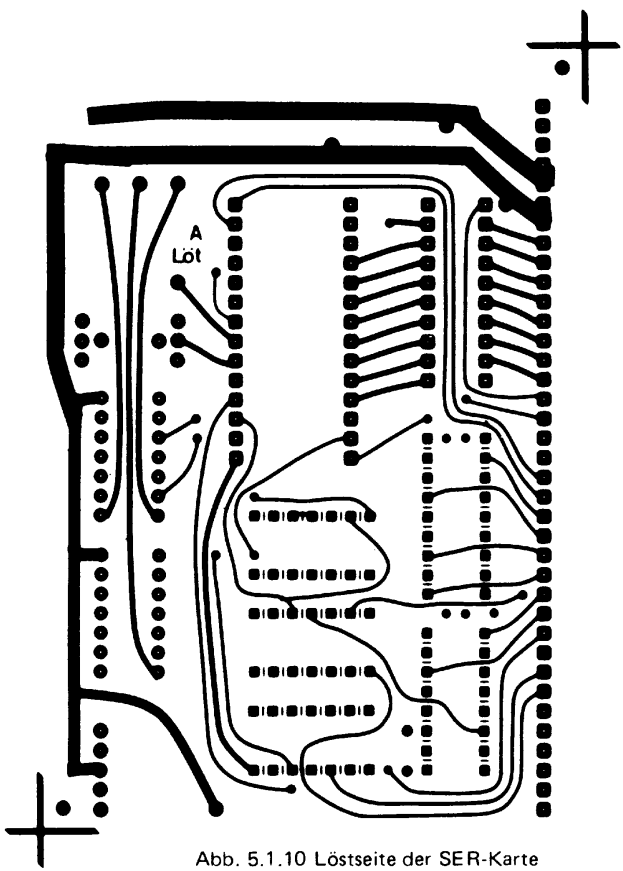


Abb. 5.1.10 Löstseite der SER-Karte

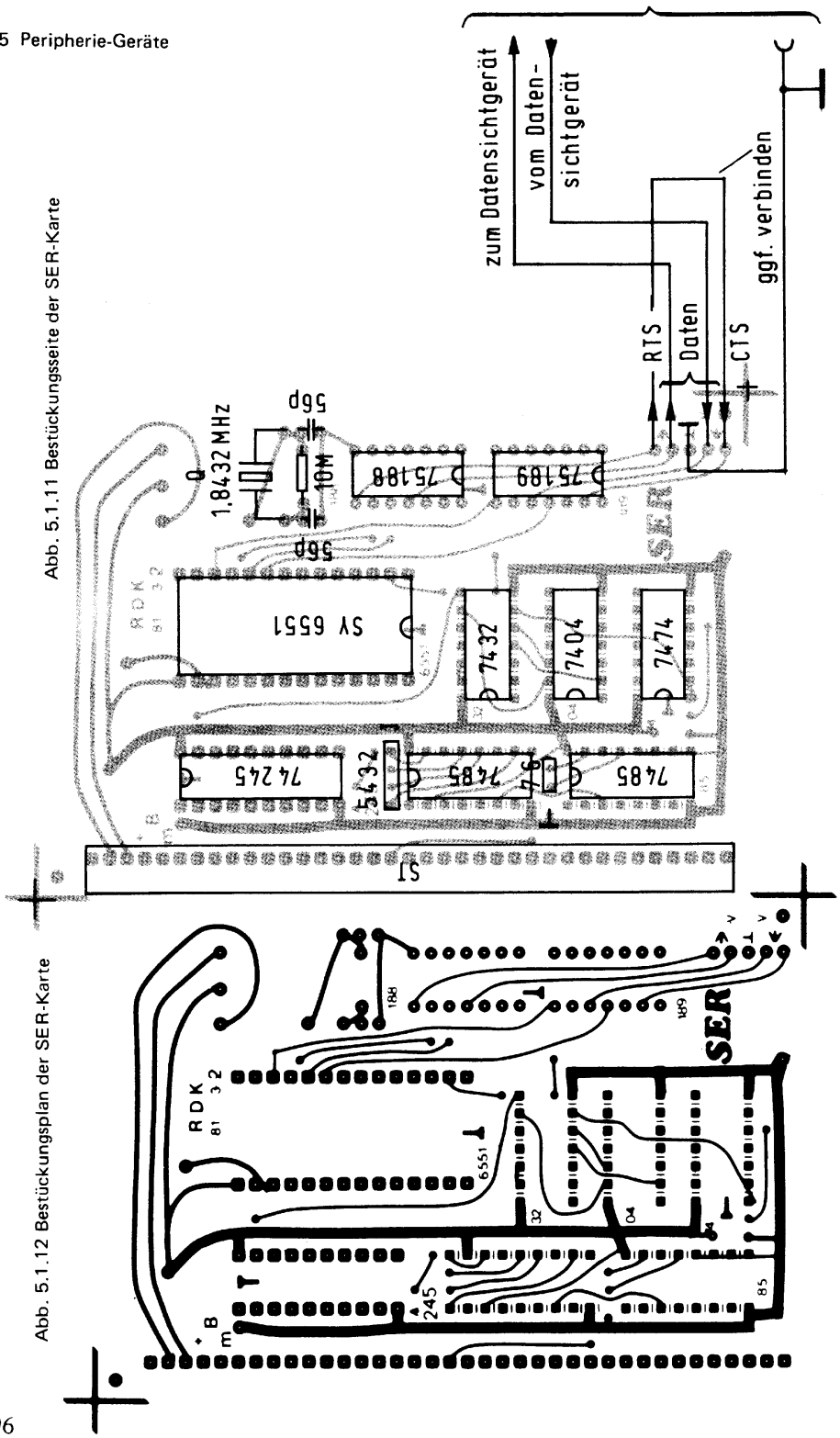


Abb. 5.1.11 Bestückungsseite der SER-Karte

Abb. 5.1.12 Bestückungsplan der SER-Karte

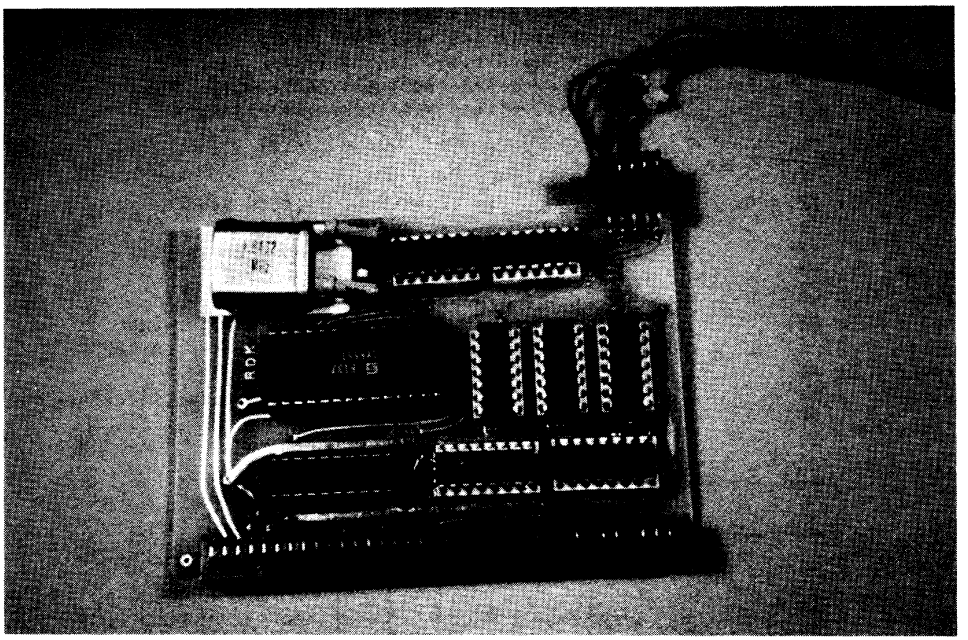


Abb. 5.1 SER-Karte

ordnet, der der Code dieses Zeichens darstellt. In der Tabelle ist der entsprechende HEX-Wert ebenfalls mit abgebildet. Die meisten Datensichtgeräte und Computerschnittstellen gerade bei Mikrocomputern richten sich nach diesem Code.

Es kann jetzt mit dem Bestücken der Leiterplatte begonnen werden. *Abb. 5.1.10* zeigt die Lötseite der Platine. *Abb. 5.1.11* zeigt die Bestückungsseite und *Abb. 5.1.12* zeigt den Bestückungsplan.

Wie üblich werden zunächst nur alle Sockel eingelötet, die ICs bleiben draußen. Ebenfalls werden die passiven Bauteile, z. B. Quarz, Widerstände und Kondensatoren eingelötet. Dann kann der Test beginnen.

1. Messen der Versorgungsspannungen an den ICs. Als erstes, messen aller 5V Spannungen an den ICs. Dann messen der +/-12V Spannung am IC 75188. +12V liegt dabei an Pin 14 und -12V an Pin 1.

2. Die Karte kann nun entweder zusammen mit dem SBC-Computer oder der CPU-Karte + RAM/ROM oder Boot-Karte betrieben werden. Für den Test genügt es, das Monitorprogramm zu verwenden, oder es werden die nachfolgenden Programmstücke zum Testen verwendet. Es werden nun alle ICs eingesetzt. Eine Verbindung des CTS-Eingangs der Karte mit dem RTS Ausgang der Karte wird hergestellt, um später zu garantieren, daß -CTS am IC auf 0 V liegt.

3. *Abb. 5.1.13* zeigt ein kurzes Testprogramm. Es definiert alle Parameter des Serienports und durchläuft eine Eingabeschleife. Damit entspricht es dem Verhalten nach dem

```
                        ;**********************************
                        ;* Serielles Interface 6551       *
                        ;* Dekoder und Timing Test        *
                        ;* Programmierung als Empfaenger  *
                        ;**********************************

0000'                   start:
0000'   3E 1E                   ld a,1eh        ;9600 Baud
0002'   D3 FF                   out (0ffh),a
0004'   3E 0B                   ld a,0bh        ;no par enable
0006'   D3 FE                   out (0feh),a
0008'   DB FD           loop:   in a,(0fdh)     ;status port
000A'   E6 08                   and 00001000b   ;receiver ready
000C'   28 FA                   jr z,loop       ;nein dann warten
000E'   DB FC                   in a,(0fch)     ;ja einlesen
0010'   18 F6                   jr loop         ;hier alles wiederholen
```

Abb. 5.1.13 Testprogramm für SER

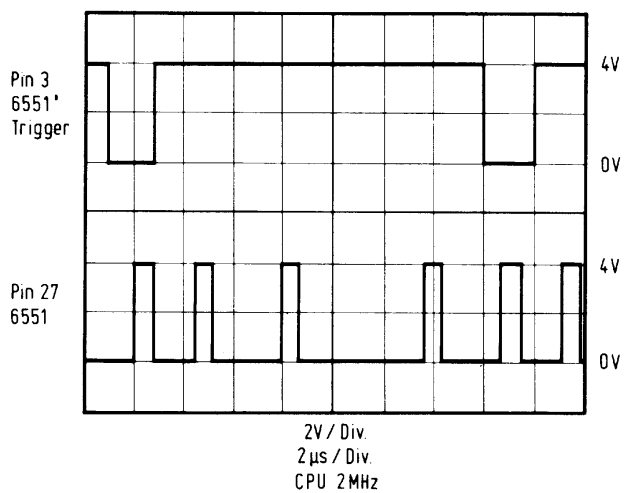


Abb. 5.1.14 Oszillogramm der Karte SER

Start des Monitorprogramms. Der Serienbaustein wird auf 9600 Baud initialisiert. Ein Test kann nun darin bestehen, ein Datensichtgerät mit der gleichen Baudrate und einem V-24-Anschluß (so wird die +/-12V Schnittstelle auch bezeichnet) mit dem Rechner zu verbinden. Dabei muß das Datensichtgerät 9600 Baud, 8 Datenbits und 1 Stop-Bit besitzen. Nach dem RESET der Schaltung muß der Monitor sich auf der Console melden. Wird eine Taste benötigt, zum Beispiel das Leerzeichen, so muß der Monitor darauf reagieren, dann ist der gesamte Test beendet. Steht kein Datensichtgerät zur Verfügung oder trat ein Fehler auf, so wird nun im folgenden systematisch getestet.

4. Nach dem Start des Testprogramms muß an PIN 5 des 6551 eine Frequenz von 153600 Hz anliegen (Periode 6.51 ys). Dies ist die 16fache Baudrate, da für die Empfängersteuerung eine höhere Taktrate benötigt wird, wie schon gezeigt.

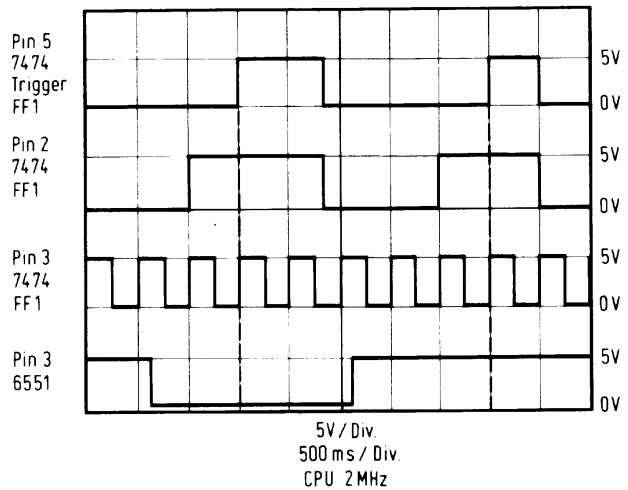


Abb. 5.1.15 Oszillogramm der Karte SER

```
                        ;*******************************
                        ;* Serielles Interface 6551    *
                        ;* Test Sender Teil V24        *
                        ;*******************************

0000'                   start1:
0000'   3E 1E                   ld a,1eh          ;9600 Baud
0002'   D3 FF                   out (0ffh),a
0004'   3E 0B                   ld a,0bh          ;no par enable
0006'   D3 FE                   out (0feh),a
0008'   0E 6A                   ld c,01101010b    ; testmuster
000A'   DB FD           loop1:  in a,(0fdh)       ;status test
000C'   E6 10                   and 10h           ;transmitter ready
000E'   28 FA                   jr z,loop1
0010'   79                      ld a,c
0011'   D3 FC                   out (0fch),a      ;auf datenport
0013'   18 F5                   jr loop1          ;und wiederholen
```

Abb. 5.1.16 Testprogramm der Karte SER

5. *Abb. 5.1.14* zeigt das Oszillogramm das den Dekodierungsvorgang darstellt. Der 6551 wird zyklisch durch unser Testprogramm angesprochen.

6. Eine Reihe von Impulsen sind in *Abb. 5.1.15* dargestellt. Dies ist die Erzeugung des Signals PHI2.

7. Nun kann der Senderteil getestet werden. *Abb. 5.1.16* zeigt das Testprogramm, das diemsal eine Ausgabeschleife darstellt, die kontinuierlich den Wert 6Ah auf die Datenleitung gibt, oder das Zeichen „j" gemäß seiner ASCII-Bedeutung. In *Abb. 5.1.17* ist ein Oszillogramm dargestellt, das direkt am Ausgang des Übertragungstreibers T3 abgenommen wurde. Zu beachten ist, daß das Signal -CTS direkt am 6551 Pin 9, den Wert 0 annehmen muß, denn sonst ist der Baustein blockiert und eine Datenausgabe erfolgt nicht.

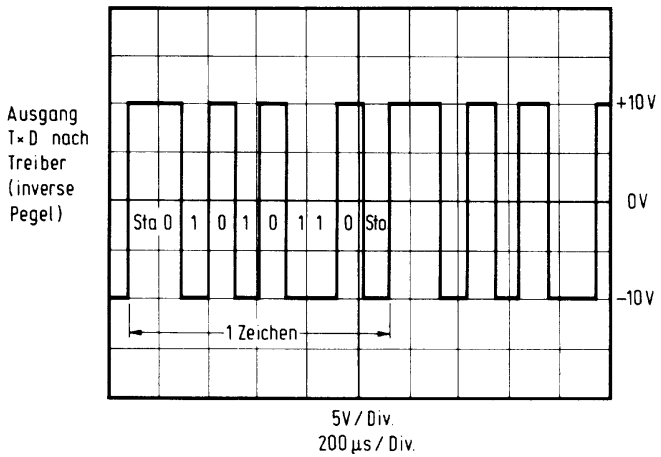


Abb. 5.1.17 Oszillogramm der Karte SER

```
                        ;*********************************
                        ;* Serielles Interface 6551      *
                        ;* Test Sender+Empfaenger        *
                        ;*********************************

0000'                   start2:
0000'  3E 1E                    ld a,1eh        ;9600 Baud
0002'  D3 FF                    out (0ffh),a
0004'  3E 0B                    ld a,0bh        ;no par enable
0006'  D3 FE                    out (0feh),a
0008'  0E 6A                    ld c,01101010b  ; testmuster
000A'  DB FD            loop2:  in a,(0fdh)     ;status test
000C'  E6 10                    and 10h         ;transmitter ready
000E'  28 FA                    jr z,loop2
0010'  79                       ld a,c
0011'  D3 FC                    out (0fch),a    ;auf datenport
0013'  DB FD            loop3:  in a,(0fdh)     ;nun empfang abwarten
0015'  E6 08                    and 8h
0017'  28 FA                    jr z,loop3      ;warten bis empfangen ist
0019'  DB FC                    in a,(0fch)     ;und wert holen
001B'  4F                       ld c,a          ;nun neuer Wert
001C'  18 EC                    jr loop2        ;muss indentisch alten sein
```

Abb. 5.1.18 Testprogramm der Karte SER

Der Eingang des Treibers CTS wurde aber mit dem Ausgang von RTS verbunden. Daher ist der Pegel ok. Wird der Eingang offen gelassen, so erscheint am Ausgang des Treibers ein 1-Signal und der Serien-Port wird blockiert. Hier ist ferner zu beachten, das die Treiber die Signale umkehren, daher rührt auch das inverse Bild in Abb. 5.1.17. Am 6551 Pin 10 ist es genau invertiert und natürlich nur mit einer 5V (4V) Amplitude vorhanden.

8. Auf einem Datensichtgeräteschirm müssen lauter Zeichen – „j" erscheinen. Wichtig ist, daß das Datensichtgerät vor dem Stromeinschalten mit dem Rechner verbunden wurde, denn sonst kann eine fehlerhafte Synchronisation auch zu falschen Zeichen auf dem Schirm führen.

9. In *Abb. 5.1.18* ist ein weiteres Programm abgebildet. Um es zu verwenden, wird entweder ein Datensichtgerät angeschlossen, oder die Verbindung vom Eingang des Seriellen Interface zum Ausgang des seriellen Interface hergestellt. Wird nach Herstellung der Rückkopplung der RESET ausgelöst, so müssen fortlaufend 6Ah-Werte über die Leitung gesendet werden. Ist ein Datensichtgerät vorhanden, so erscheint nur ein „j" am Bildschirm. Danach wird jedes Zeichen, das auf der Tastatur eingegeben wird, auch wieder auf den Bildschirm ausgegeben. Zu beachten ist beim Anschluß des Datensichtgeräts auch hier wieder, daß der Ausgang unseres Serien-Interface mit dem entsprechenden Daten-Eingang des Datensichtgerätes verbunden wird und umgekehrt. Ist man sich dabei nicht ganz sicher, so kann eine Messung mit einem Oszilloscop Aufschluß darüber geben. Dabei muß im Ruhezustand am Datenausgang des Sichtgerätes ein -12V-Pegel liegen und bei Betätigung einer Taste ein kurzes Datenmuster erscheinen. Wird der so gefundene Ausgang mit dem entsprechenden Eingang des Datensichtgerätes verbunden, muß beim Drücken einer Taste auf dem Bildschirm ein Zeichen erscheinen. Geschieht dies nicht, so kann es sein, daß das Datensichtgerät ebenfalls einen CTS-Eingang besitzt und dann genügt es, diesen Eingang mit dem RTS-Ausgang des Datensichtgerätes zu verbinden. Funktioniert dieser Test, so kann der Eingang CTS des Datensichtgerätes mit unserem Ausgang RTS verbunden werden und umgekehrt unser Eingang CTS mit dessen Ausgang RTS.

5.2 Tastatur-Anschluß

Für diejenigen, denen kein komplettes Datensichtgerät zur Verfügung steht, wird nun der Selbstbau eines solchen beschrieben. Wir beginnen dabei mit dem einfachsten Teil, der Tastatur.

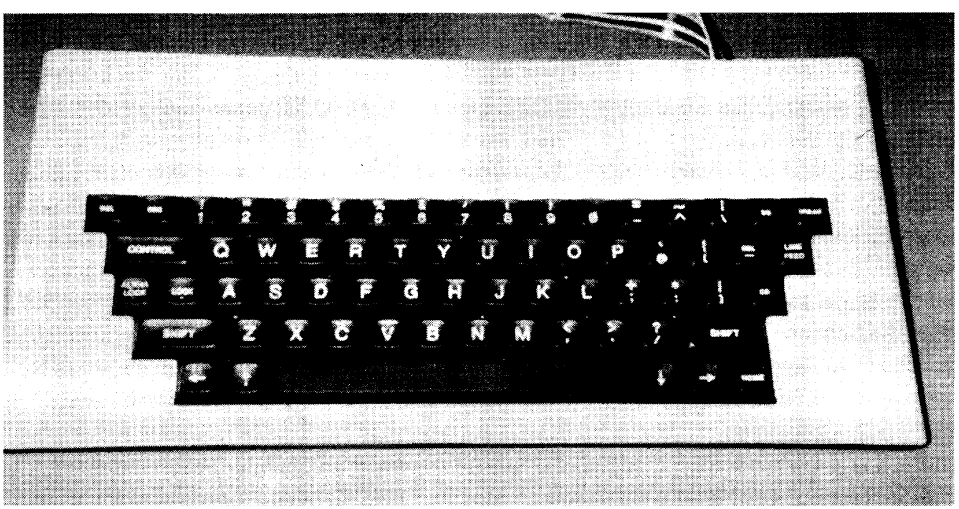


Abb. 5.2 Low-Cost-Tastatur von CHERRY

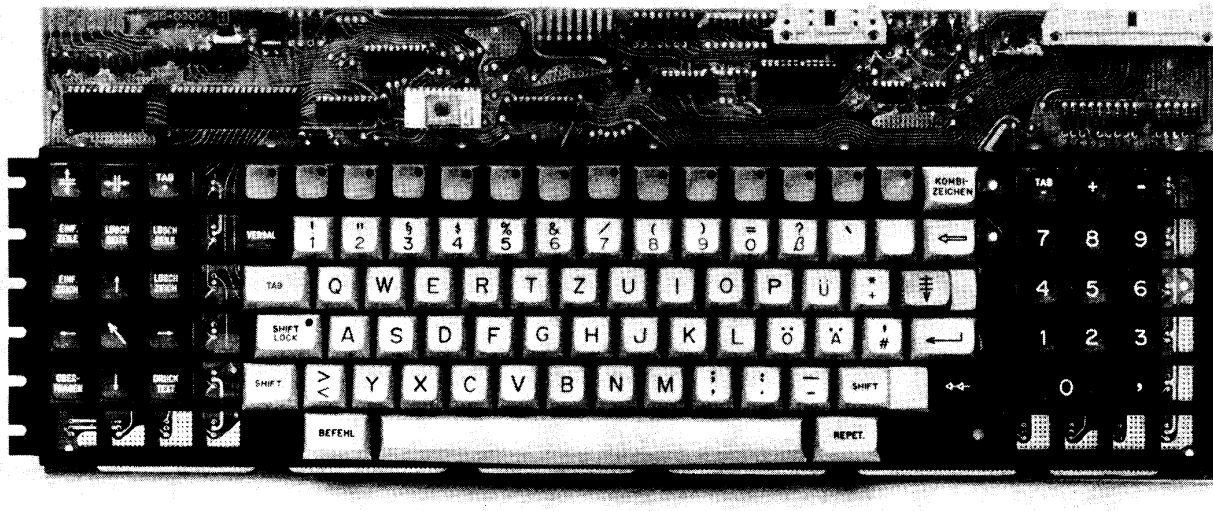


Abb. 5.3 komfortable Tastatur

Es lohnt sich nicht, die Tastatur selbst zu bauen, da dafür im Handel preiswerte Bausätze und auch fertige Tastaturen vorhanden sind. *Abb. 5-2* zeigt eine Low-Cost-Tastatur von der Firma Cherry die zu einem Preis von ca. 200 DM erhältlich ist. Eine komfortablere Tastatur ist in *Abb. 5-3* gezeigt, sie stammt von der Firma Honeywell GmbH und stellt so das andere Extrem dar. Beim Kauf einer Tastatur sind ein paar Dinge zu beachten. Zum Einen benötigen wir eine Standard-ASCII-Tastatur, denn es werden vereinzelt auch EBCDIC oder ähnlich andere codierte Tastaturen angeboten. Zum Andern benötigen wir einen Parallelausgang, was aber die gebräuchlichste Form ist.

Die Low-Cost-Tastatur von Cherry z. B. erfüllt alle diese Anforderungen. Ein weiterer Punkt ist das Vorhandensein von ein paar wichtigen Steuertasten. Die Tasten CTRL und CR müssen vorhanden sein. Mit CTRL ist es möglich, durch Betätigen einer Buchstabentaste ein Steuerzeichen des Bereich 0..1Fh auszugeben. Diese Steuerzeichen werden zum Teil benötigt, um z. B. Befehle wie Cursor Links, Cursor rechts, Bildschirmlöschen o. ä. durchzuführen. Ein getrenntes numerisches Tastenfeld ist zwar schön jedoch nicht unbedingt erforderlich. Ferner sollte man sich vor Tastaturen hüten, die keine Kleinbuchstaben ausgeben können. Wir werden zum Anschluß eine Platine mit einem Parallel-Interface verwenden. *Abb. 5.2.1* zeigt die Schaltung. Ein Buffer B2 dient als Eingabeport. Ferner ist auf der Karte auch noch ein weiterer Port zur Abfrage von Schalterstellungen vorhanden. Über die Schalter kann zum Beispiel die gewünschte Betriebsart für den Sichtschirm sowie die Baudrate für ein serielles Interface eingestellt werden. Ist die Karte nicht da, so sollte durch Pull-Up-Widerstände auf dem Datenbus gesichert sein, das der Wert OFFh eingelesen wird. Dann kann die Monitorsoftware automatisch das serielle Interface verwenden. Ist die Karte vorhanden, so merkt das Monitor-Programm anhand der Schalterstellung, ob mit der Tastatur gearbeitet werden soll oder nicht. Die Schalterstellungen sind im Monitorlisting des Softwareteils beschrieben. Die Datenleitungen der Tastatur werden an die

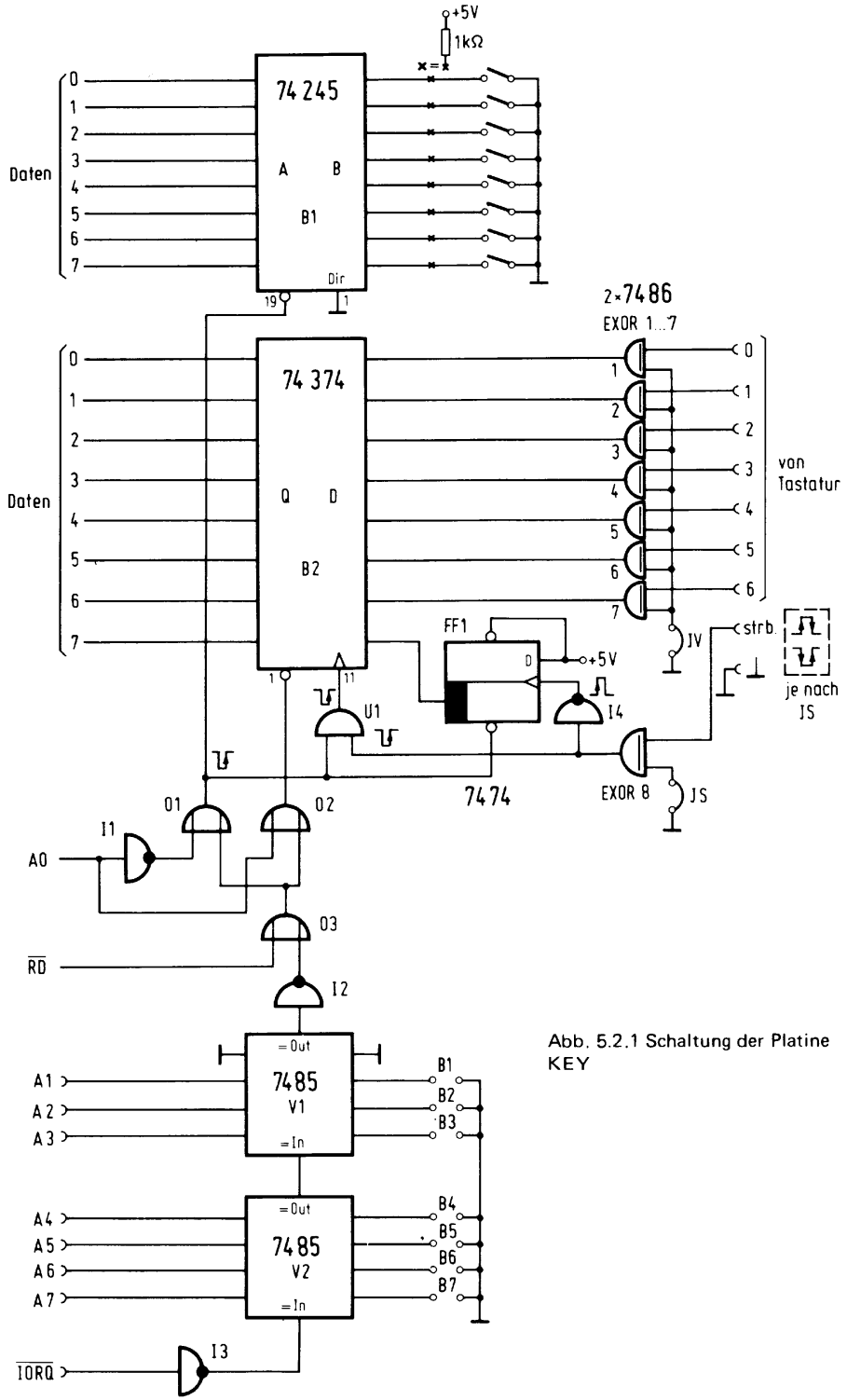


Abb. 5.2.1 Schaltung der Platine KEY

Anschlüsse 0 bis 6 geführt. Es sind nur 7 Leitungen verwendet, da der ASCII-Satz nur 128 Zeichen belegt. Manche Tastaturen besitzen auch nach einem achten Ausgang, der dann aber eine Paritätsleitung darstellt, die wir hier nicht benötigen. Sie kann offen bleiben.

Die Daten gelangen von den Platineneingängen auf die Exclusiv-Oder-Gatter Ex1 bis Ex7. Über die Brücke JV kann der andere Eingang der Exclusiv-Oder-Gatter wahlweise mit 1 oder 0 belegt werden. Liegt ein 0-Signal am anderen Eingang wirken die Gatter als 1 zu 1 Übertrager ohne den logischen Wert zu verändern. Sind sie auf 1, so wirken die Exclusiv-Oder-Gatter wie ein Inverter. Damit ist es möglich, auch Tastaturen zu verwenden, die einen Ausgang mit negativer Logik verwenden. Am Treiber B2 muß in jedem Fall mit positiver Logik der ASCII-Wert eines eingegebenen Zeichens anstehen. Nun liefert die Tastatur einen kurzen Impuls, der die Betätigung einer Taste anzeigt. Dieser Puls kann je nach Ausführung positiv oder negativ sein. Um den logischen Wert umdrehen zu können, ist das Exclusiv-Oder-Gatter Ex8 vorhanden. Mit der Brücke JS kann auch hier die richtige Polarität eingestellt werden. Über das Oder-Gatter 02 kann der Inhalt des Treiber-Latches B2 auf den Datenbus geschaltet werden. Dazu wird am Vergleicher V1 und V2 eine Portadresse mit den Brücken B1 bis B7 eingestellt. Als Adressen wählen wir 68h und 69h. Dazu wird bei den Brücken folgendes Bitmuster eingestellt:

B7	B6	B5	B4	B3	B2	B1
ein	–	–	ein	–	ein	ein

– bedeutet, die Brücke bleibt offen.

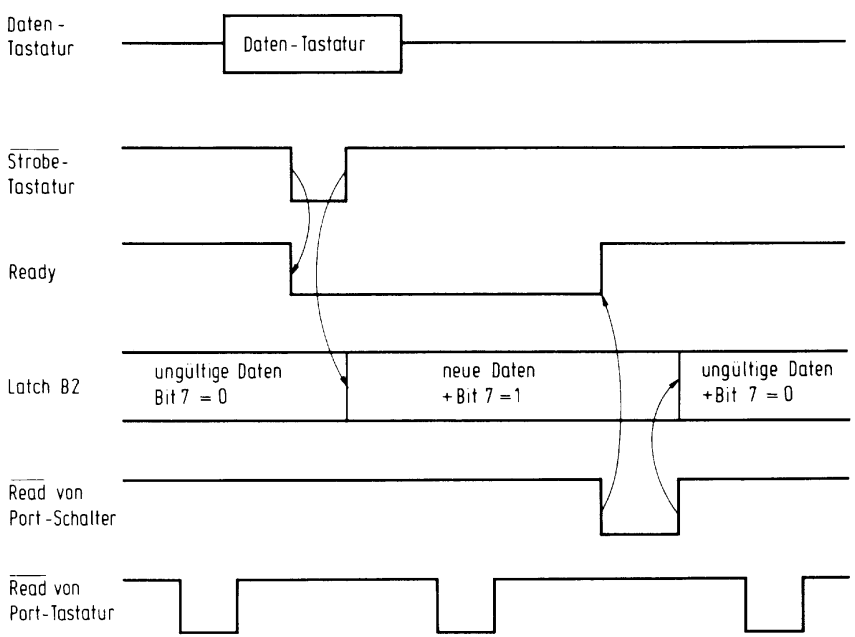


Abb. 5.2.2 Zeitlicher Ablauf bei einer Tastatureingabe

Die Tastatur-Daten können bei Adresse 68h abgerufen werden und die Positionen der Dil-Schalter sind bei Adresse 69h verfügbar.

Abb. 5.2.2 zeigt den Zeitablauf bei einem Eingabevorgang. Über die Adresse wird der Tastaturport ständig abgefragt. Das Bit 7 in den Eingabedaten gibt an, ob eine Eingabe der Tastatur erfolgt ist. Ist das Bit 7 auf dem Wert 1, so liegt keine Eingabe vor. Erfolgt nun das Strobesignal, so wird mit der fallenden Flanke ein 1-Signal in das Flip-Flop FF1 geschoben und am Q-Quer Ausgang des Flip-Flops erscheint eine 0. Dieser Wert wird zusammen mit den noch anstehenden Tastatur-Daten mit der steigenden Flanke des Strobe-Signals in den Zwischenspeicher in B2 übernommen. Bei einer erneuten Abfrage des Tastatur-Ports ist das Bit 7 nun auf 0 gesetzt und dies bedeutet, daß die Daten gültig sind. Sie können dann weiterverarbeitet werden. Nun kann der Zustand des Flip-Flops wieder gelöscht werden und eine nachfolgende Abfrage zeigt, daß die Daten übernommen werden. Dies geschieht durch Einlesen des Dil-Schalter-Ports auf Adresse 69h. Mit der fallenden Flanke des Lese-Pulses wird das Flip-Flop FF1 wieder gelöscht und mit der steigenden Flanke wird der Zustand des Q-Quer-Ausgangs in den Zwischenspeicher B2 übernommen. Der Zustand der Daten ist dabei nicht bestimmt, da nicht alle Tastaturen die Daten be-

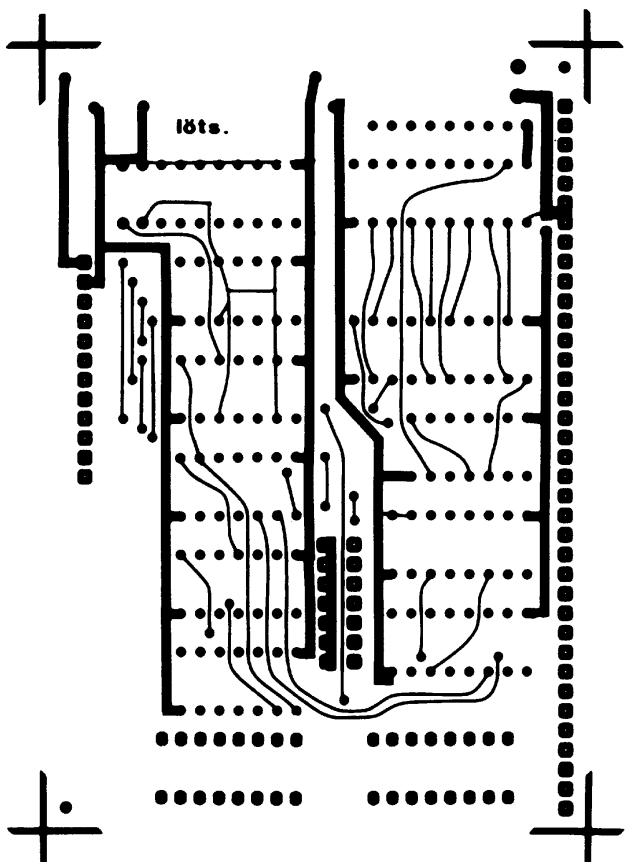


Abb. 5.2.3 Lötseite der Platine KEY

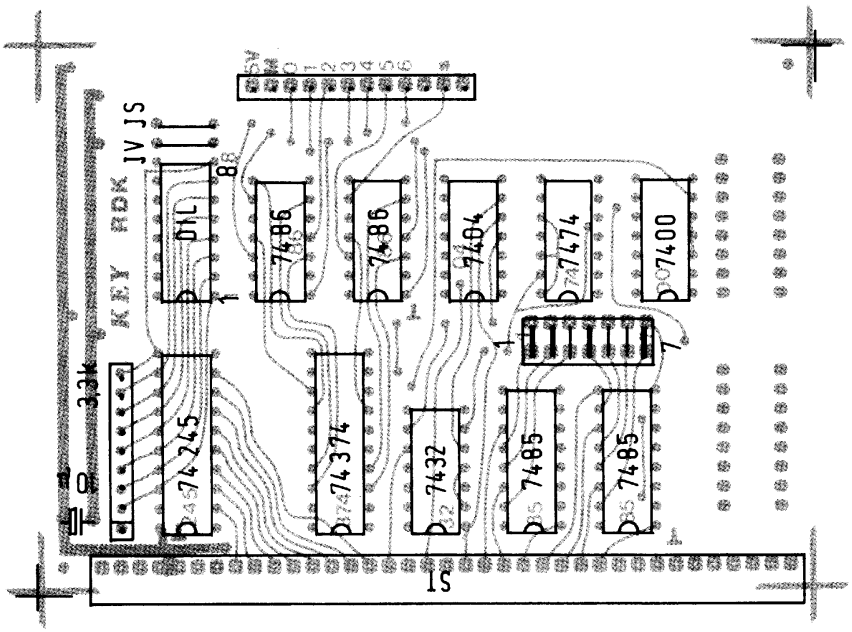


Abb. 5.2.5 Bestückungsplan der Platine KEY

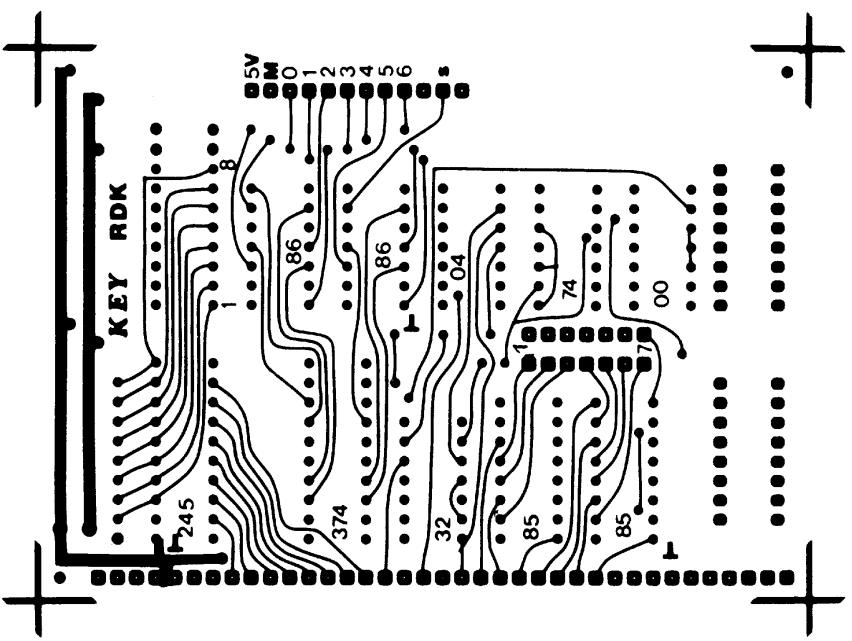


Abb. 5.2.4 Bestückungsseite der Platine KEY

liebig lang anstehen lassen, doch ist nun Bit 7 auf 1 und bei einer erneuten Abfrage des Tastatur-Ports auf Adresse 68h ergibt sich, daß die Daten ungültig sind.

Es ist nicht vorteilhaft, den Löschvorgang gleich durch Auslesen des Tastatur-Ports vorzunehmen, denn manchmal will man die Tastaturdaten vorabfragen, um festzustellen, ob überhaupt Daten da sind und erst in einem anderen Programm sollen dann die Daten übernommen werden. Dies entspricht auch mehr dem Verhalten des seriellen Ports, bei dem durch Abfragen des Statusports noch nicht das Daten-Da-Bit gelöscht wird, sondern erst beim Einlesen der Daten. Bei uns ist allerdings Status und Datenport identisch. Daher brauchen wir einen Extra-Port zum Löschen.

Abb. 5.2.3 zeigt die Lötseite der Platine und in *Abb. 5.2.4* ist die Bestückungsseite abgebildet. *Abb. 5.2.5* zeigt den Bestückungsplan der Platine.

Beim Aufbau der Platine wird wieder nach dem Standard-Verfahren vorgegangen. Zunächst wird alles mit Sockeln bestückt. Dann wird der Kondensator eingelötet und schließlich die Brücken B1 bis B7.

1. Anschluß einer Tastatur. Dazu müssen gemäß der Tastaturanleitung, die der jeweiligen Tastatur beiliegt, die benötigten Signale Daten 0 bis 6 und der Strobeausgang gefunden werden. Sie werden über eine lange Leitung mit der Tastaturplatte verbunden. Auf der Platine sind die Signale beschriftet. Die Tastatur benötigt ebenfalls eine Versorgungsspannung. Meißt nur 0V bis +5V (z. B. Low-Cost Cherry-Tastatur). Wird zusätzlich eine -12V Versorgung gebraucht, wie das bei alten Tastaturen gebräuchlich war, so muß die -12V Versorgungsspannung auch noch auf die Steckerleiste an der Platine geführt werden. Dazu verwendet man am Besten den der 5V-Leitung gegenüberliegenden Anschluß.

2. Test der Tastatur. Mit einem Scop wird die Polarität des ankommenden Strobe-Signals geprüft. Kommt ein negativer Puls an, so wird die Brücke JS eingefügt, wenn nicht, wird sie weggelassen. Bei den Daten wird ähnlich verfahren. Mit einer positiven Logik wird die Brücke JV eingelötet. Direkt hinter den 74LS86 Bausteinen muß eine der positiven Logik entsprechendes Signal anstehen.

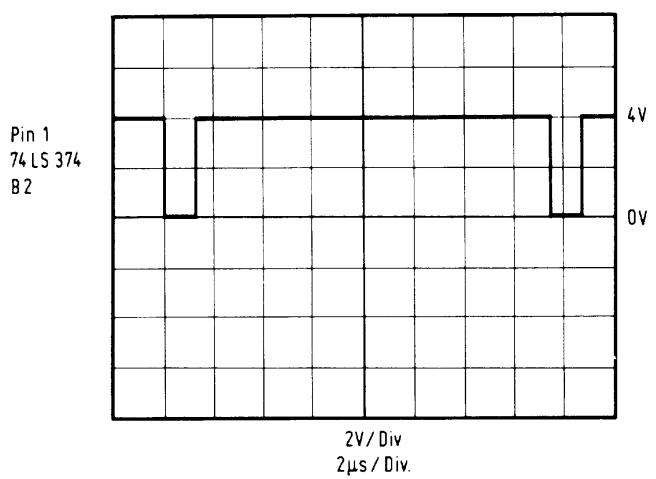


Abb. 5.2.6
Oszillogramm der
Karte KEY

```
                        ;*****************************
                        ;* Tastatur Testprogramm     *
                        ;*****************************

0000'   DB 68           Start:  in a,(68h)      ;lese Tastaturport
0002'   DB 69                  in a,(69h)      ;lese Dil-Schalter
0004'   18 FA                  jr start
```

Abb. 5.2.7 Testprogramm der Karte KEY

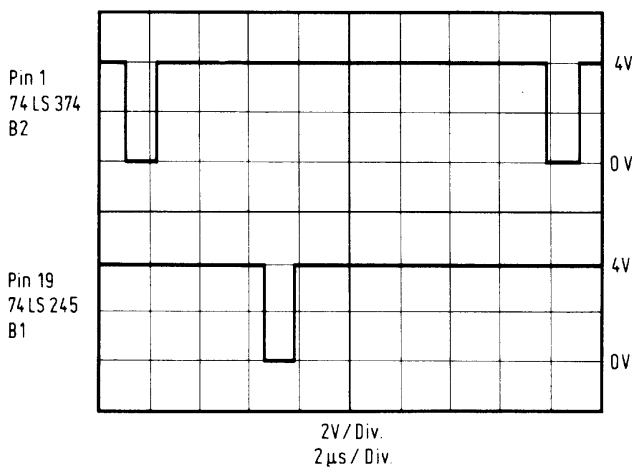


Abb. 5.2.8 Oscillogramm der Karte KEY

3. Einsatz der restlichen Bausteine. Der Dilschalter wird nun auf den Betrieb mit Datensichtgerät und Tastatur eingestellt:

DIL	1	2	3	4	5	6	7	8
	on	on	on	off	off	on	on	on.

4. Nun muß an Pin 1 des Schaltkreises 74LS374 ein Pulsmuster nach *Abb. 5.2.6* entstehen, wenn das Monitorprogramm im EPROM des SBC-Computers vorhanden ist. Der Test muß auch ohne CRT-Controller-Karte arbeiten.

5. Ein weiterer Test geschieht mit einem Testprom nach *Abb. 5.2.7* das beide Port bezüglich der Adressierung. *Abb. 5.2.8* zeigt das dazugehörige Ozsillogramm.

Der Resttest kann nur zusammen mit der CRT-Platine oder einem Serienport erfolgen. Steht das serielle Interface zur Verfügung, so genügt es beim Dil-Schalter den CRT-Mode auszuschalten. Dann erfolgen alle Datenausgaben über das serielle Interface, während die Eingabe von der Tastatur erfolgt.

Der Dil-Schalter steht dann wie folgt:

DIL	1	2	3	4	5	6	7	8 (Achtung Bits 0...7)
	on	on	on	off	off	on	on	off

Damit ist der Aufbau der KEY-Karte abgeschlossen.

5.3 Aufbau eines Datensichtgerätes

Mit Hilfe einer CRT-Karte kann zusammen mit der KEY-Karte des vorherigen Abschnitts ein vollständiges Datensichtgerät aufgebaut werden. Dieses Datensichtgerät ist dann über den Z80-Bus mit der CPU-Karte verbunden und über die Tastatur können Befehle eingegeben werden. Wird zusätzlich eine Seriell-Interface-Karte verwendet, so läßt sich die ganze Anordnung auch als selbständiges Terminal verwenden und kann über das serielle Interface z. B. mit einem anderen Computer verbunden werden. Die Daten von diesem Computer werden dann auf dem Bildschirm angezeigt und Eingaben von der Tastatur gelangen zum Computer über die serielle Leitung. Das Programm für diese Terminal-Betriebsweise befindet sich bereits in unserem Monitorprogramm und kann mit dem Befehl „T" gestartet werden (siehe auch den entsprechenden Softwareabschnitt).

Nun zu den Grundlagen der Betriebsweise von Datensichtgeräten. Wir werden mit einem Verfahren arbeiten, das auch normale Fernsehgeräte verwenden. Bei einem Fernseher gibt es 625 Zeilen. Dabei wird diese Auflösung durch das sogenannte Zeilensprungverfahren erzeugt. Es wird dazu mit einer Bildwiederholrate von 50 Hz jeweils ein Halbbild auf dem Bildschirm dargestellt. In *Abb. 5.3.1* und *Abb. 5.3.2* ist der Strahlverlauf bei zwei Halbbildern dargestellt. Es werden zunächst beim ersten Halbbild alle ungeraden Zei-

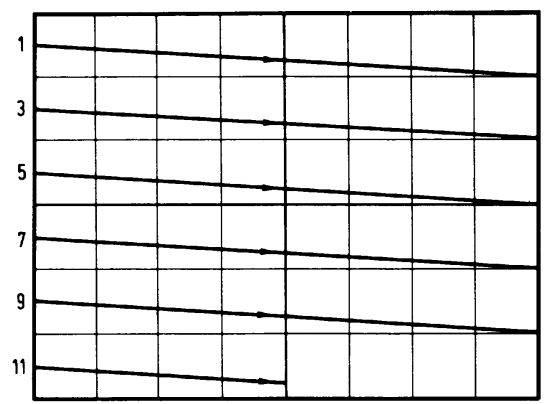


Abb. 5.3.1 Vorgang beim Zeilensprungverfahren ungerade Zahl

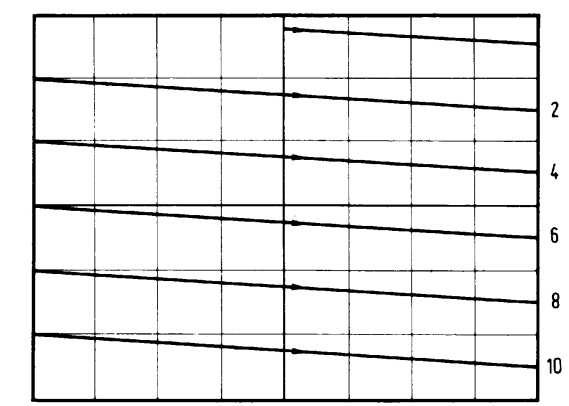


Abb. 5.3.2 Vorgang beim Zeilensprungverfahren gerade Zahl

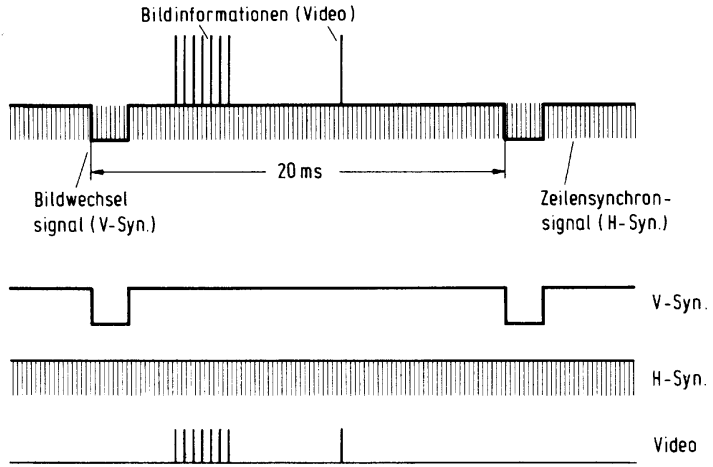


Abb. 5.3.3
Videosignale

len erfaßt, beim zweiten Bilddurchlauf, nach 20ms werden alle geraden Zeilen durchlaufen und danach wiederholt sich der Vorgang. Dadurch wird erreicht, daß das Bild nicht so stark flimmert, da mit 50Hz ein ganzer Bildaufbau durchgeführt wird, obwohl die eigentliche Rate nur 25 Hz beträgt. Da wir eine so hohe Auflösung nicht benötigen arbeitet unser CRT nicht mit dem Zeilenvorsprungverahren, sondern wir beschreiben nur jeweils die ungeraden Zeilen. Dies ist mit jedem Fernseher möglich und gibt den Vorteil, daß das Bild nicht mehr flimmert.

Um dieses Raster zu erzeugen, gibt es ein Signal, das alle Informationen enthält und BAS-Signal genannt wird. Der TV-Schirm oder Sichtgeräte-Monitor benötigt im Prinzip drei Signale. Einmal das Vertikal-Synchronsignal, dann das Horizontal-Synchronsignal und schließlich noch die eigentliche Video-Information. Ein Bildschirm besitzt intern zwei Oszillatoren. Der eine sorgt für die horizontale Ablenkung und der andere für die vertikale Ablenkung. Nun müssen diese beiden Oszillatoren auf das externe Bildsignal synchronisiert werden. Dies geschieht mit den beiden Synchronsignalen. Das Vertikal-Synchronsignal steuert den Vertikal-Oszillator im Bildschirm und das Horizontal-Synchronsignal steuert entsprechend den Horizontal-Oszillator. Damit nich alle drei Signale über getrennte Leitungen, oder beim TV-Gerät über getrennte Sendefrequenzen übertragen werden müssen, wird eine Mischung der drei Signale erzeugt, und es entsteht das BAS-Signal *Abb. 5.3.3.* Das Bildwechsel oder Vertikal-Synchronsignal ist zusammen mit dem Zeilen- oder Horizontalsynchronsignal gemischt und die Bildinformation befindet sich oberhalb dieser Synchronsignale. Im Bildchirmgerät gibt es eine Schaltung, die aus diesem Gemisch wieder die einzelnen Signale herauszieht.

Die Amplitude des Videosignals bestimmt die Helligkeit des Bildes. In *Abb. 5.3.4* ist eine Ausschnittsvergrößerung des Signals, bei der eine Zeile dargestellt ist. Eine Zeile hat eine Dauer von 64 ys. Die Schwarzschulter beim Synchronsignal bestimmt durch ihren Pegel den Wert bei dem der Bildschirm dunkel erscheint.

Wir wollen nun Buchstaben auf dem Bildschirm erzeugen. Dazu müssen die Buchstaben in ein Punktraster zerlegt, und Zeile für Zeile auf den Bildschirm gebracht werden.

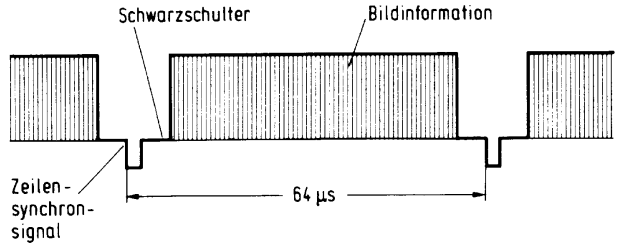


Abb. 5.3.4
Zeilensignale

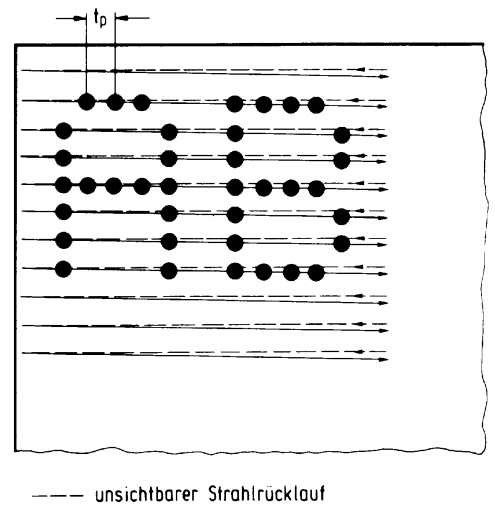


Abb. 5.3.5
Zeichenraster

Abb. 5.3.5 zeigt das Verfahren anhand des Buchstabens „A" und „B". Der Buchstabe wurde nicht nur in Zeilen zerlegt, sondern auch in Spalten. Dies hat folgenden Grund: Die Zeichen müssen in der zerlegten Form in einem Speicher irgendwo abgelegt sein, um falls sie angezeigt werden sollen, auch in dieser Form zur Verfügung zu stehen. Für ein Zeichen das in ein 8*8 Punktraster zerlegt ist, wird ein Speicherplatz von 8 Bytes benötigt. Soll der volle ASCII-Satz dargestellt werden, so werden (Steuerzeichen ausgenommen) 96*8 Bytes benötigt. Wird ein Zeichen mit mehr Punkten dargestellt, so wird auch entsprechend mehr Speicherplatz benötigt. Bei einer Auflösung von 16*16 Punkten pro Zeichen werden 96*32 Bytes benötigt. Die höhere Auflösung bedingt aber auch noch eine höhere Ausgabefrequenz. Dazu ein Beispiel: Wir wollen ein Zeichen inclusiv Abstand zum nächsten Zeichen mit 8 Punkten in horizontaler Richtung darstellen. Als Bildbreite haben wir 40ys Zeit, da der Rest der 64 ys meist nicht sichtbar ist und auch für den Strahlrücklauf benötigt wird. Wir wollen 64 Zeichen pro Zeile unterbringen, also wird pro Punkt (40ys/(64*8)) = 0.078125 ys Zeit verfügbar sein (tp im Bild). Dies entspricht einer Frequenz von 12.8 MHz. Damit tritt auch noch ein anderes Problem auf, die Bandbreite des verwendeten Sichtschirms. Wird ein normaler Fernseher verwendet, so beträgt die Bandbreite ca. 5 bis 6 MHz, bei Farbgeräten noch weniger. Die Zeichen sehen daher dort etwas verwaschen aus. *Abb. 5.3.6* zeigt, wie der Anschluß eines Fernsehers prinzipiell über einen

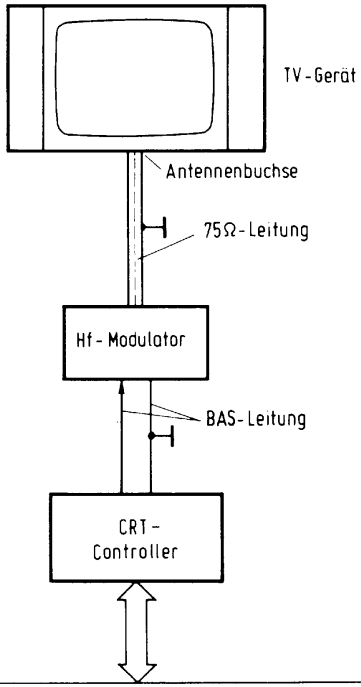


Abb. 5.3.6 Blockschaltbild eines CRT-Gerätes mit HF-Modulator

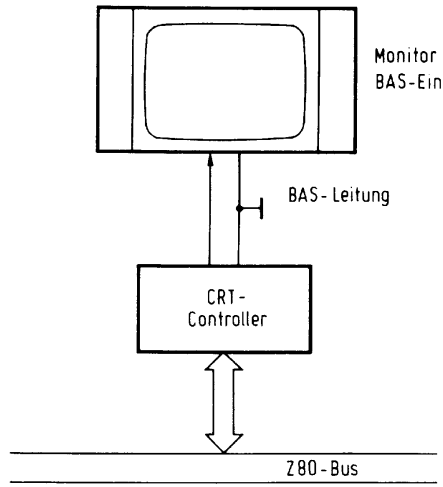


Abb. 5.3.7 Blockschaltbild eines CRT-Gerätes ohne HF-Modulator

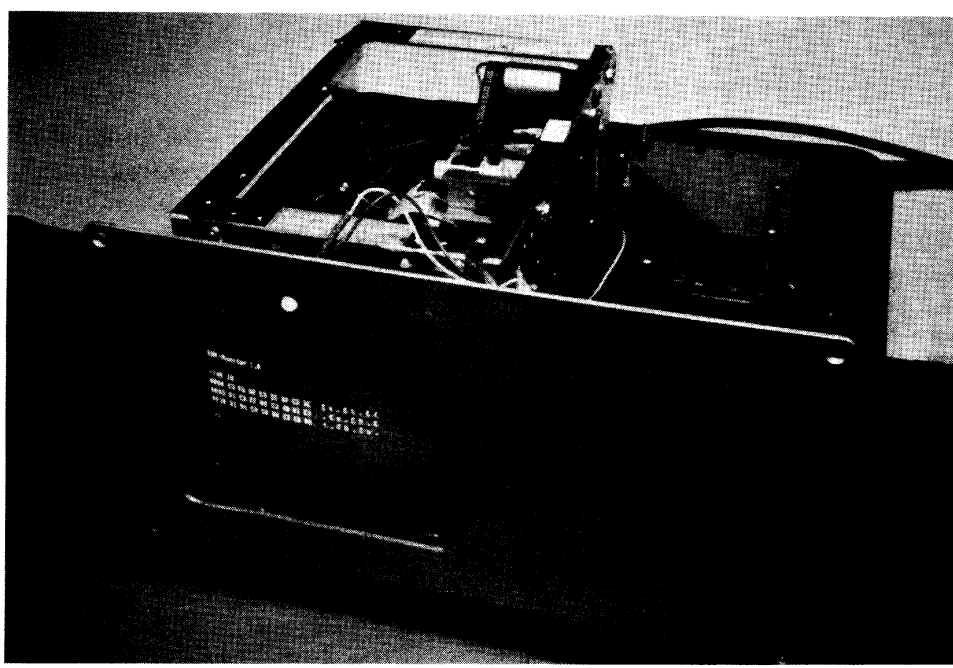


Abb. 5.4 Kleiner Monitor

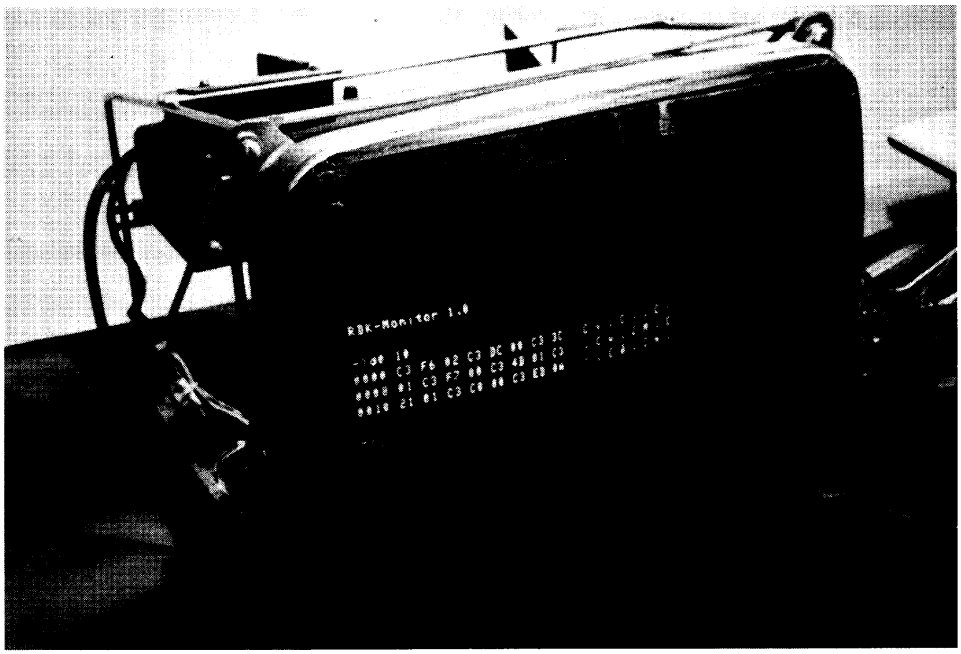


Abb. 5.5 Großer Monitorschirm

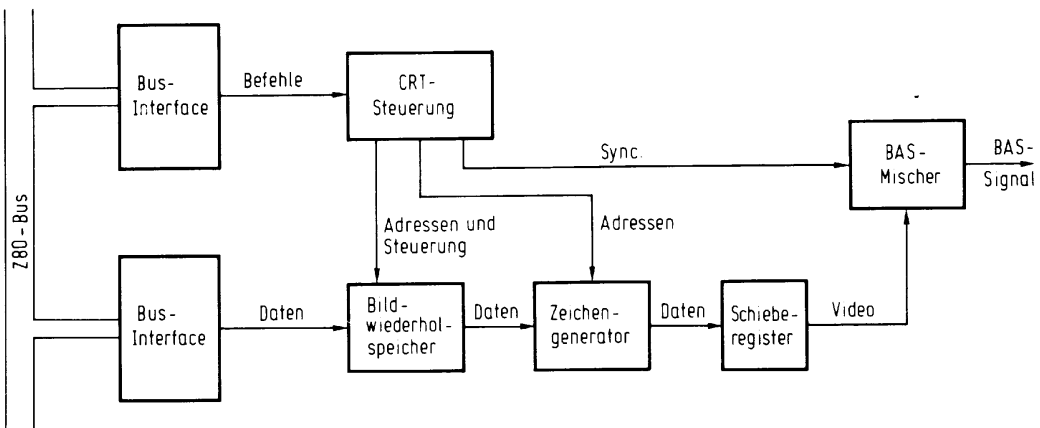


Abb. 5.3.8 Blockschaltung der Crt-Karte

Hf-Modulator möglich ist. Solche HF-Modulatoren gibt es fertig im Handel und sie verlangen meist, aber nicht immer, ein BAS-Signal am Eingang.

Heutzutage ist aber ein guter Video-Monitor mit BAS-Eingang und einer Bandbreite ab 18 MHz auch nicht teuer als ein Fernsehgerät (Preis ca. ab 300 DM) und bietet ein wesent lich besseres Bild. Allen, die sich intensiver mit Mikrorechnern beschäftigen wollen, sei daher der Kauf eines solchen Monitors angeraten. *Abb. 5.3.7* zeigt den Anschluß an einen Video-Monitor. Beim Kauf sollte man jedoch darauf achten, daß der Monitor einen BAS-

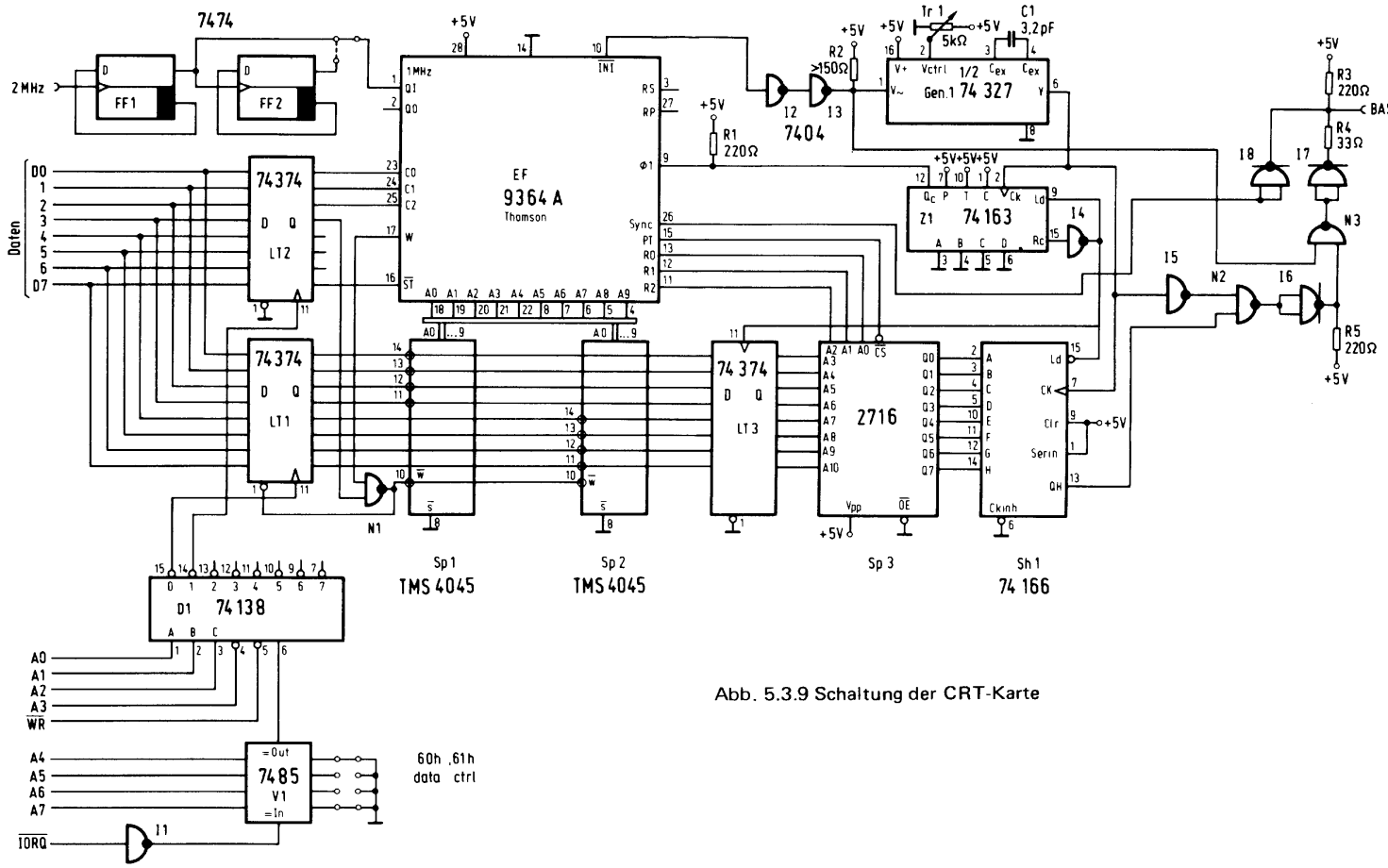


Abb. 5.3.9 Schaltung der CRT-Karte

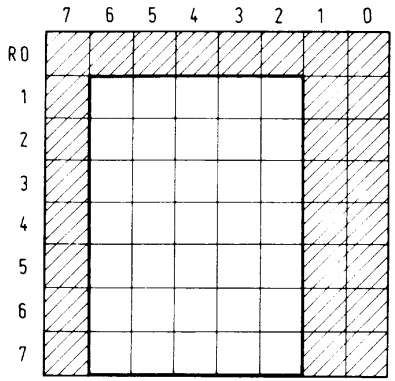


Abb. 5.3.10 Zeichenraster 8*8

Eingang besitzt, denn es gibt auch solche mit getrennten Eingängen und die sind für unsere Zwecke nicht brauchbar. *Abb. 7-4* zeigt einen Selbstbau Monitor mit einer 5" Röhre. In *Abb. 7-5* ist ein Monitor mit einer großen 19" Bildröhre abgebildet. Ein großer Bildschirm ist in jedem Fall einem kleinen vorzuziehen.

In *Abb. 5.3.8* ist ein Blockschaltbild unseres CRT-Interface angegeben. Es gibt inzwischen eine Vielzahl von CRT-Controller-Bausteinen. Einer davon ist besonders elegant, da er einen einfachen Aufbau ermöglicht. Wir verwenden daher den Baustein EF 9364 A von Thomson-EFCIS. Er übernimmt die gesamte Video-Steuerung. Außerdem kann er Funktionen wie Bildschirmlöschen, positionieren o.ä. erfüllen. Ferner wird noch ein Bildwiederholspeicher benötigt, in dem alle abzubildenden Zeichen stehen und ein Zeichengenerator in dem der Zeichensatz fest gespeichert ist. Abb. 5.3.9 zeigt die ausgeführte Schaltung. Sie erlaubt ein Bildschirmformat von 64 Zeichen pro Zeile mit insgesamt 16 Zeilen. Als Bildwiederholspeicher dienen die ICs4045 Sp1 und Sp2. Er kann 1024 Zeichen speichern und entspricht damit genau dem Bildformat 64*16. Die Daten des Bildwiederholspeichers werden über einen Zwischenspeicher an den Zeichengenerator geführt. Als Zeichengenerator dient das EPROM 2716. Als Zeichenmatrix dient in Punktraster 8*8. Nun wird aber auch ein Zwischenraum wzischen zwei Zeichen benötigt. In *Abb. 5.3.10* ist die Aufteilung dargestellt. Der schraffierte Teil ist der Zwischenraum in horizontaler und vertikaler Richtung. *Abb. 5.3.11 a...j* betreffen den Zeichengenerator und damit den Inhalt des EPROMs.

Die Daten gelangen vom Ausgang des EPROMs an ein Schieberegister Sh1. Es hat die Aufgabe die prallel ankommenden Bits eines Zeichens nacheinander auszulesen. Die herausgeschobenen Bits sind schon direkt das Videosignal.

Der Bildspeicher wird zyklisch ausgelesen. Dies übernimmt der Videocontroller.EF 9364. Der Baustein benötigt zwei unterschiedliche Taktfrequenzen. Zum Einen einen Takt mit der Frequenz 1 MHz an dem Eingang QI. Daraus werden die ganzen Synchronsignale abgeleitet. Dann wird noch ein Takt am Einang PHI 1 benötigt, der den Zeichentakt darstellt. Er dient der Bildaufbereitung in horizontaler Richtung. Mit dem Generator Gen1 wird der Punkttakt erzeugt. Mit TR1 läßt er sich abgleichen. Dieser Takt gelangt an den Teiler Z1. Er erzeugt einmal einen Zeichentakt und steuert außerdem das Schieberegister.

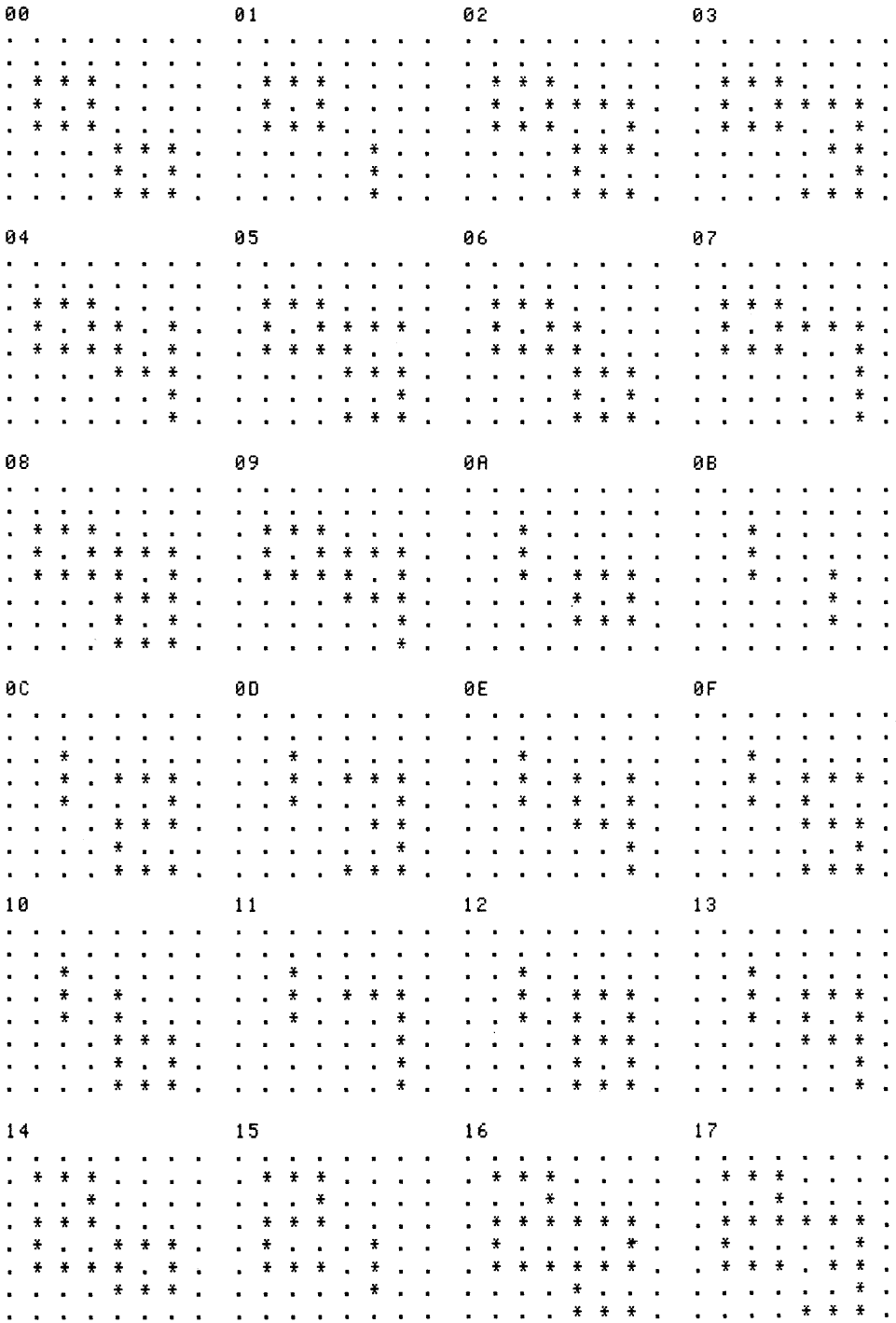
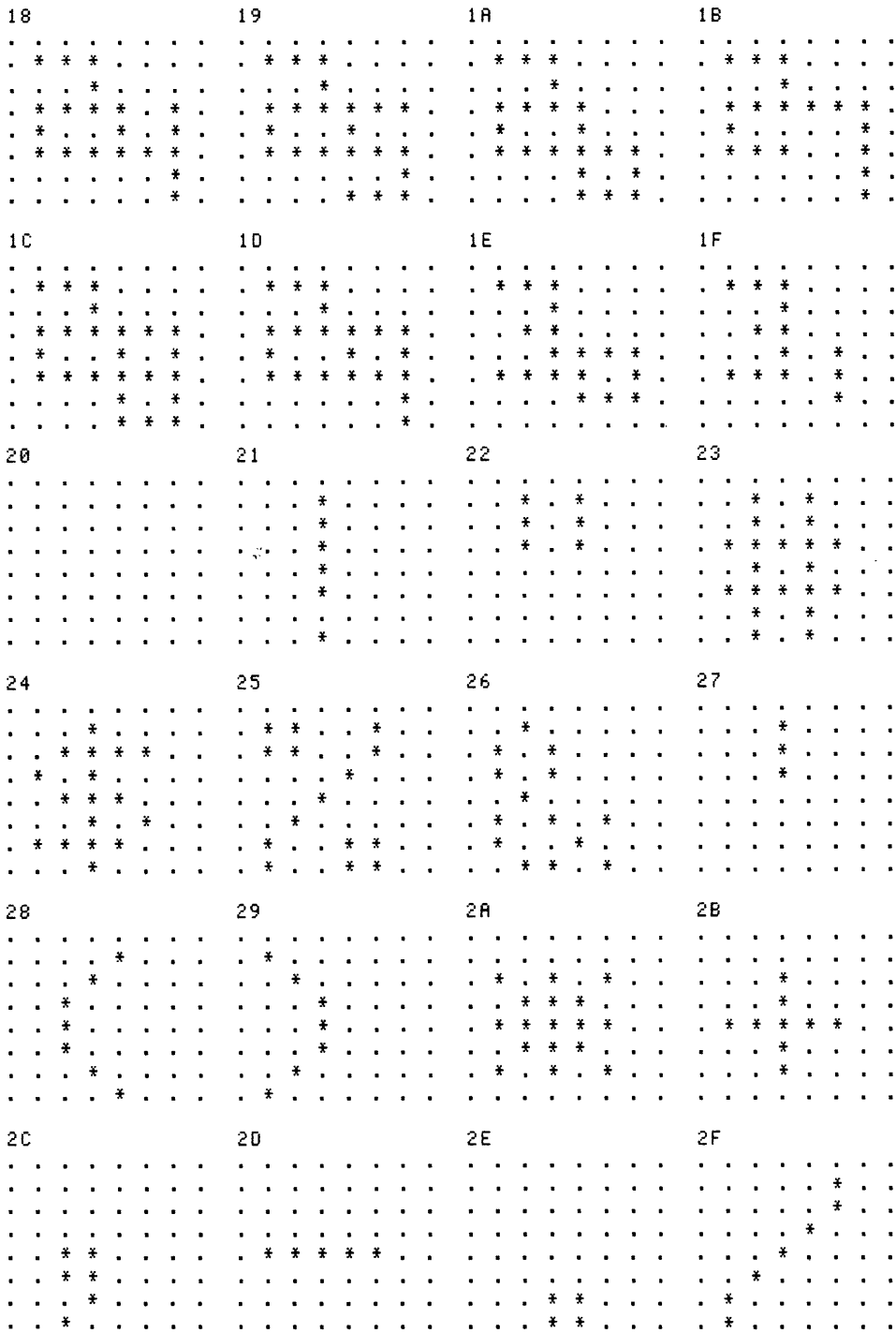
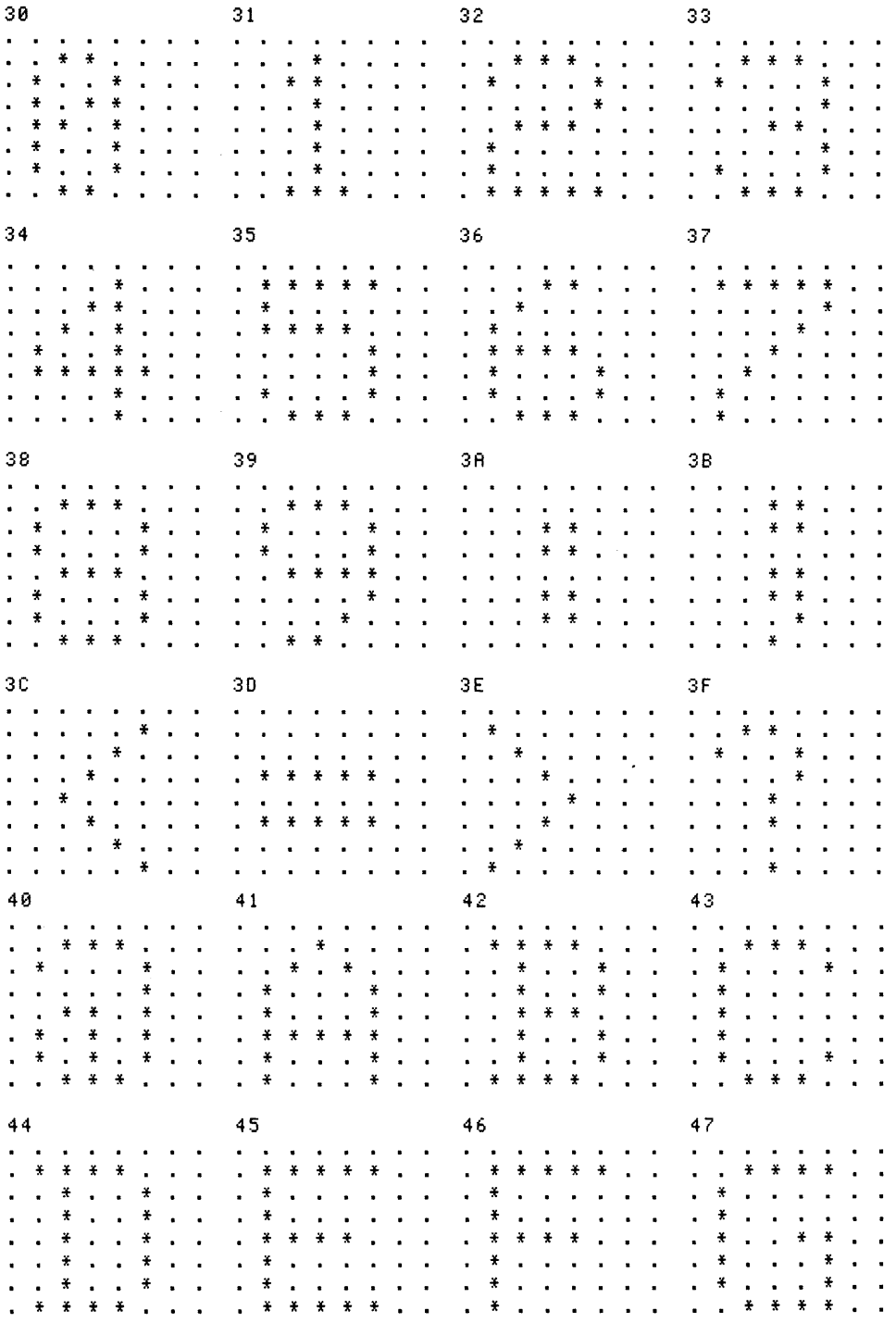


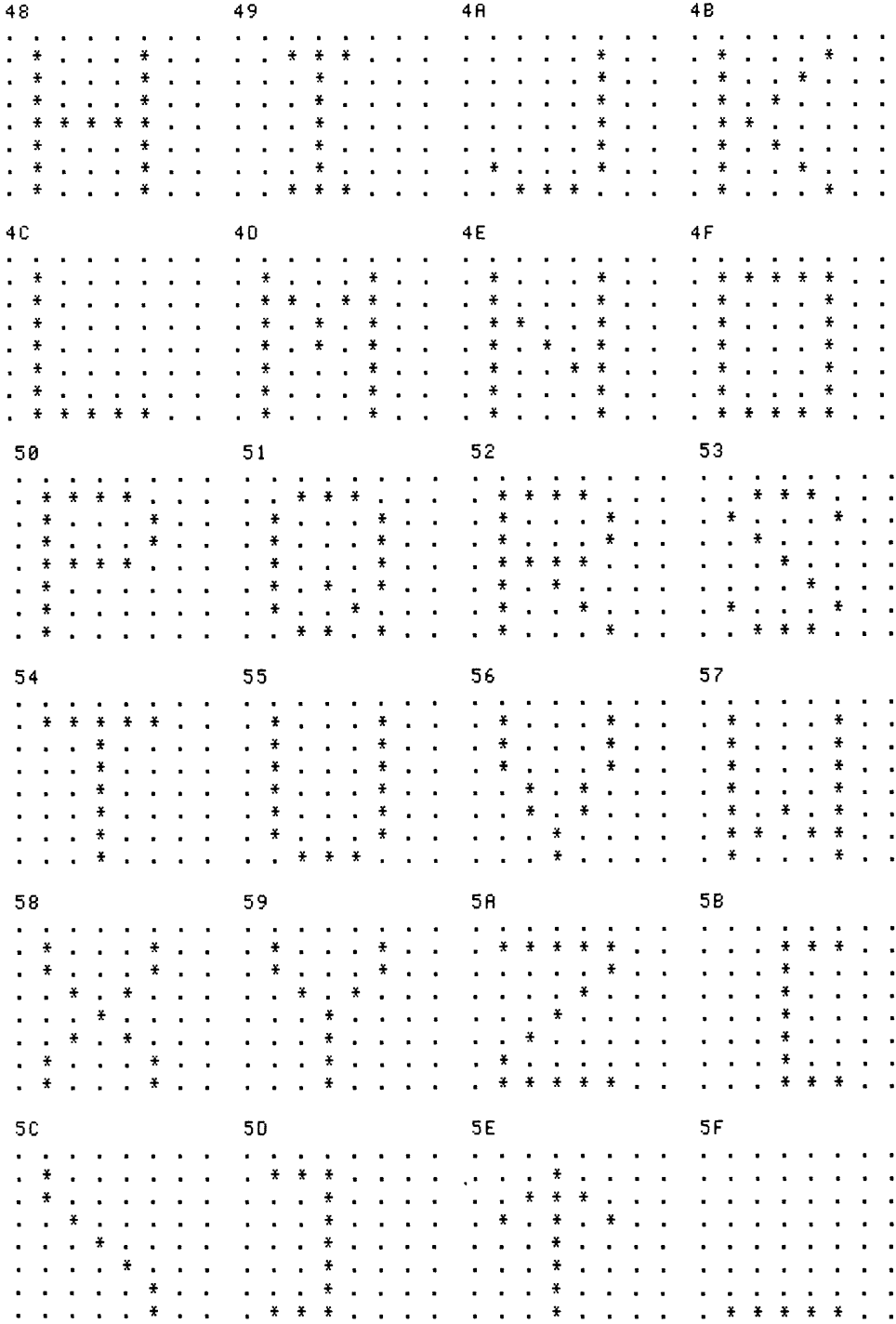
Abb. 5.3.11



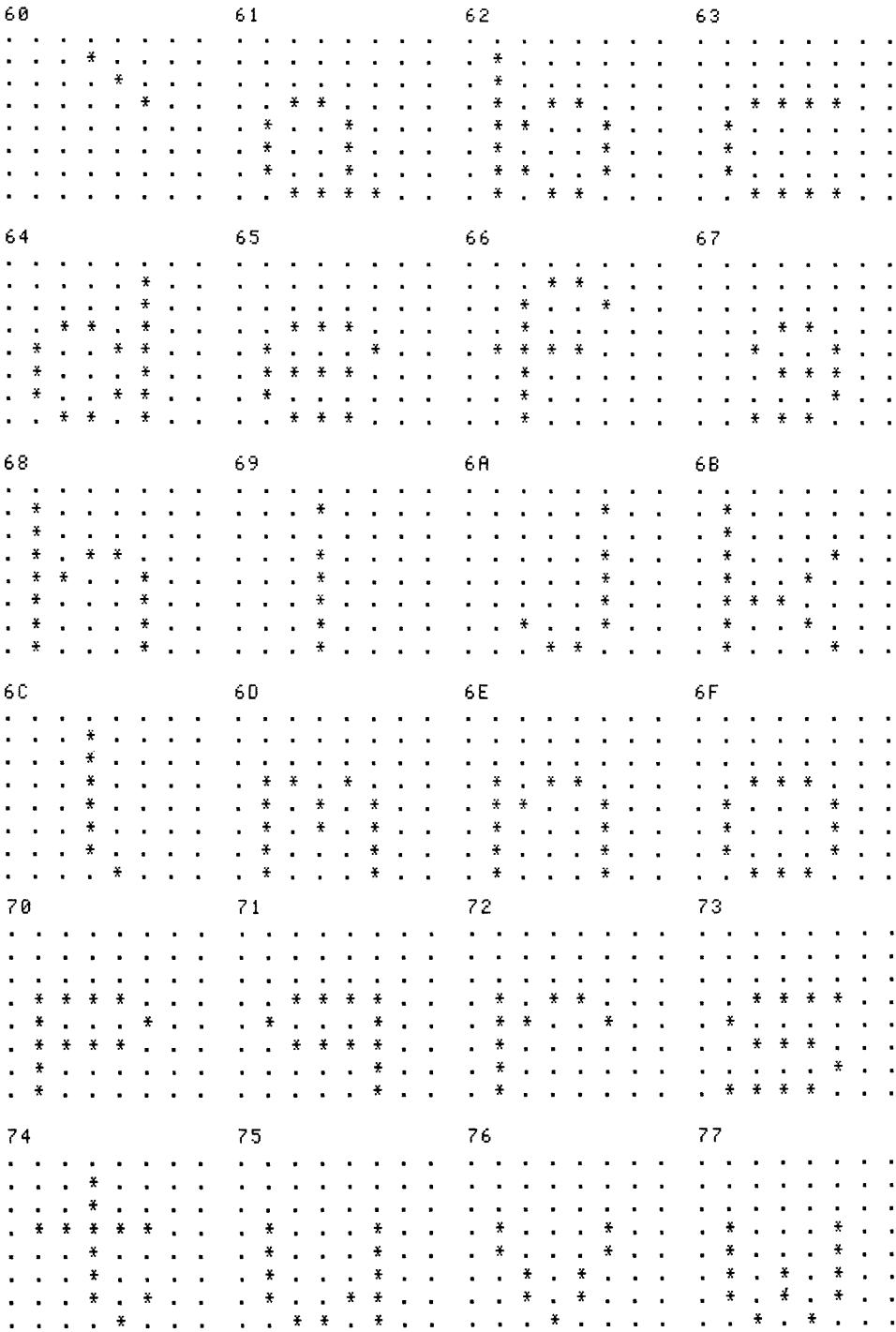
zu Abb. 5.3.11



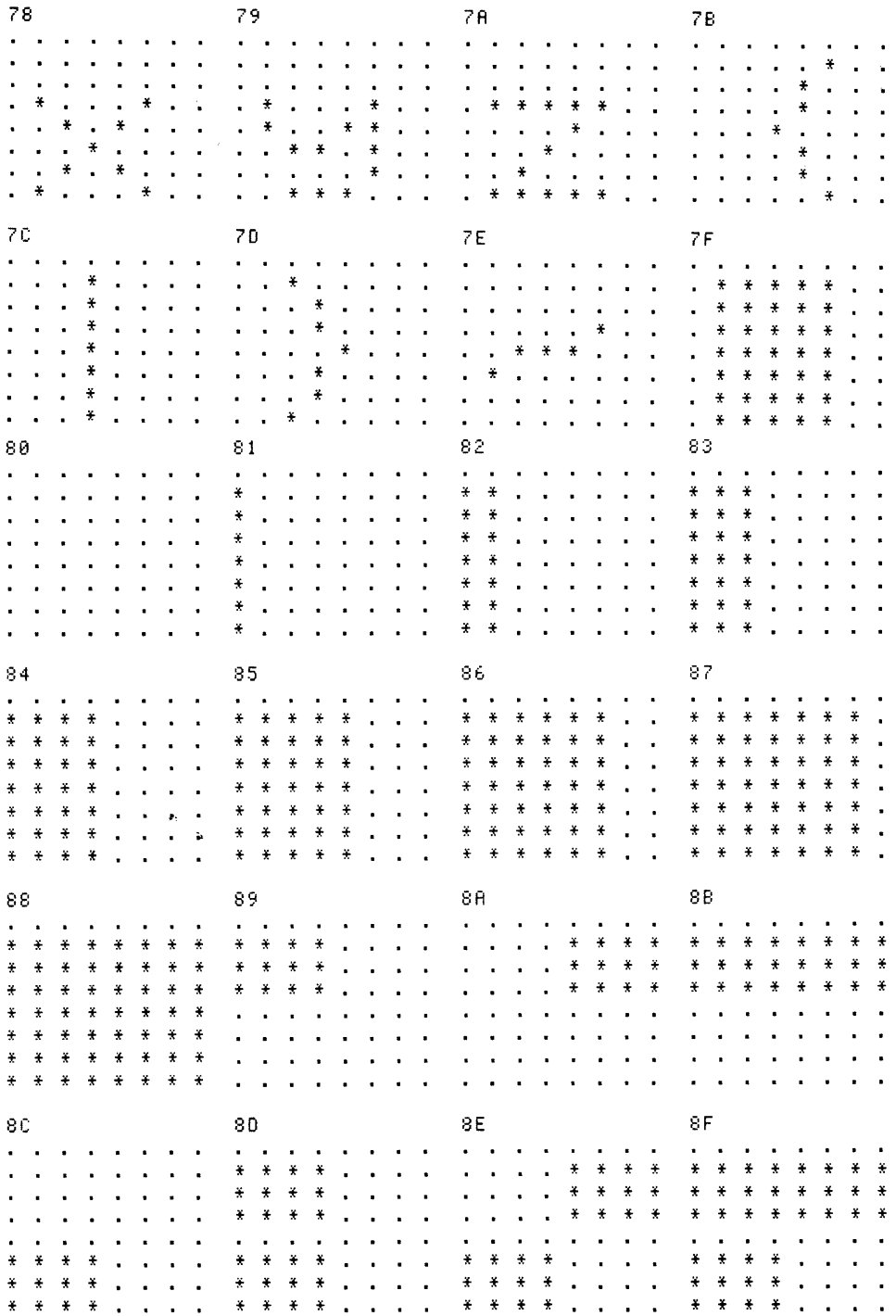
zu Abb. 5.3.11



zu Abb. 5.3.11



zu Abb. 5.3.11



zu Abb. 5.3.11

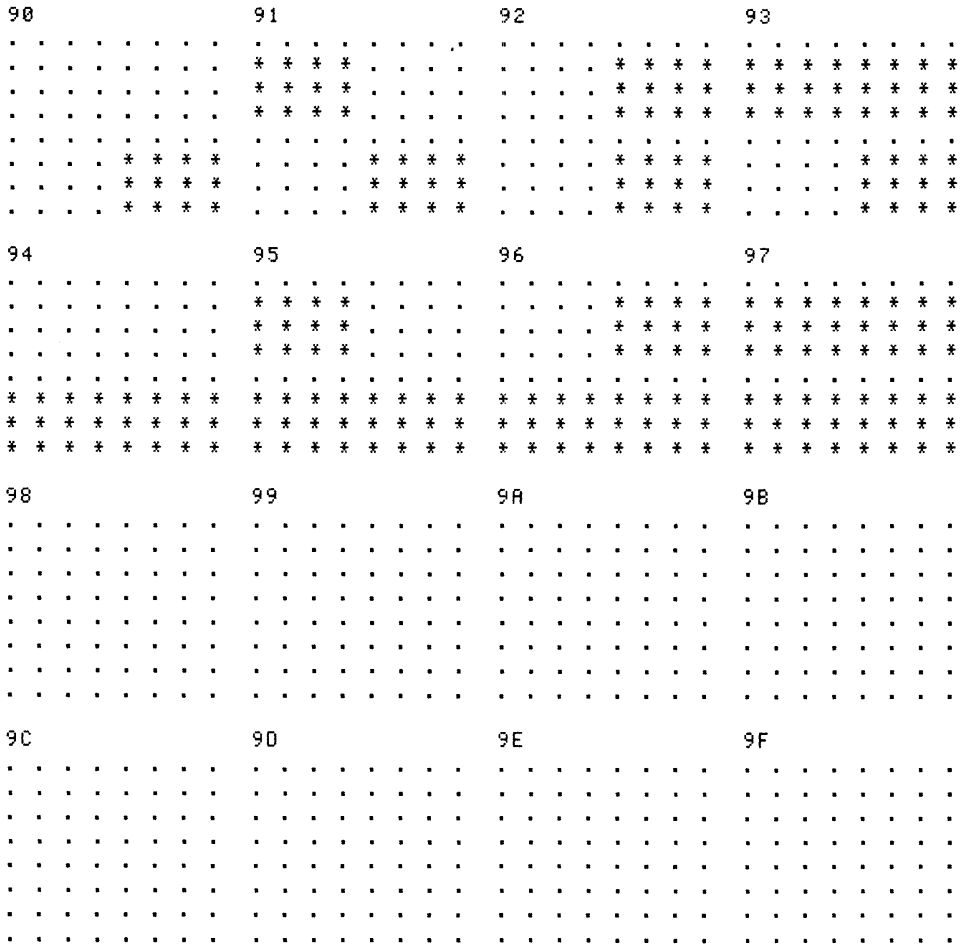


Abb. 5.3.11 Beispiel eines Zeichengenerators

Die Daten für den Bildwiederholspeicher können vom Z80 mit Hilfe des Zwischenspeichers LT1 vorbereitet werden. Durch ein Kommando an den CRT-Controller wird der Einschreibvorgang ausgelöst. Der CRT-Controller erhält seine Befehle über das Latch LT2. Im Listing des CRT-Programms (Softwareabschnitt) sind die einzelnen Befehle kurz erklärt. Wir werden uns noch etwas mit den einzelnen Signalen beschäftigen. A0 bis A9 ist die Zeichenadresse. Bei einem Einschreibevorgang stellt sie die Adresse dar, bei der der Zugriff erfolgen soll. RO0 bis RO2 ist die Zeilenadresse für den Zeichengenerator. Dazu wird hier wie folgt gezählt. 0-1-2-3-4-5-6-7-0-0-0-0 und dann wieder von vorne. Die Adresse 0 wird immer bei einem Zwischenraum angesprochen. Daher darf an dieser Stelle im Zeichengenerator kein Bit den Wert Eins haben. Der Ausgang -INI des CRT-Controllers dient dazu den Horizontaltakt zu synchronisieren. -INI ist immer dann auf einem High-

Pegel, wenn der Oszillator laufen soll. Liegt er auf 0 so wird der Oszillator gestoppt. Der Ausgang SYNC beinhaltet die Synchronsignale H-Sync und V-Sync. Am Ausgang PT steht ein Cursor-Signal an. Dies ist eine meist blinkende Schreibmarke au dem Bildschirm, die angibt an welcher Stelle das nächste Zeichen eingeschrieben wird. Der Ausgang PT führt bei uns an den Eingang -CS des EPROMs. Liegt der Wert 1 so wird das EPROM gesperrt und damit die Stelle dunkelgetastet. Die Ausgänge RS und RP benötigen wir nicht, sie dienen einer Bildwiederholspeichererweiterung. Der Ausgang W des CRT-Controllers führt an das Gatter N1. Dort wird es mit dem Bit 3 des Latches LT2 verknüpft. Liegt es auf dem Wert 0 so ist der Ausgang von N1 permanent auf 1 und ein Schreibvorgang ist gesperrt. Liegt es auf dem Wert 1, so hängt der Werk des Ausgangs N1 vom Zustand des Ausgangs W ab. Liegt er auch auf 1 so wird der Inhalt von LT1 in die gerade durch A0 bis A9 adressierte Zelle geschrieben. Die Adressteuerung übernimmt der CRT-Controller, der deshalb die gerade aktuelle Cursoradresse an den Speicher anlegt. Der Eingang -ST des CRT-Controller übergibt mit der steigenden Flanke von -ST, den an dem Latch LT2 anstehenden Befehl an den Controller.

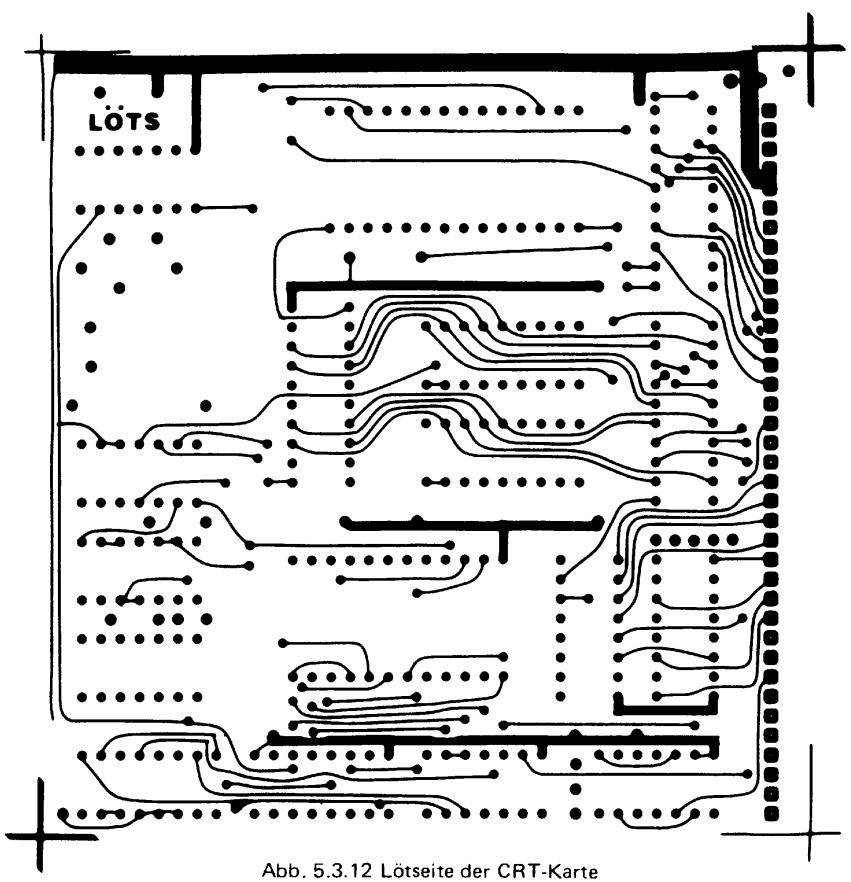


Abb. 5.3.12 Lötseite der CRT-Karte

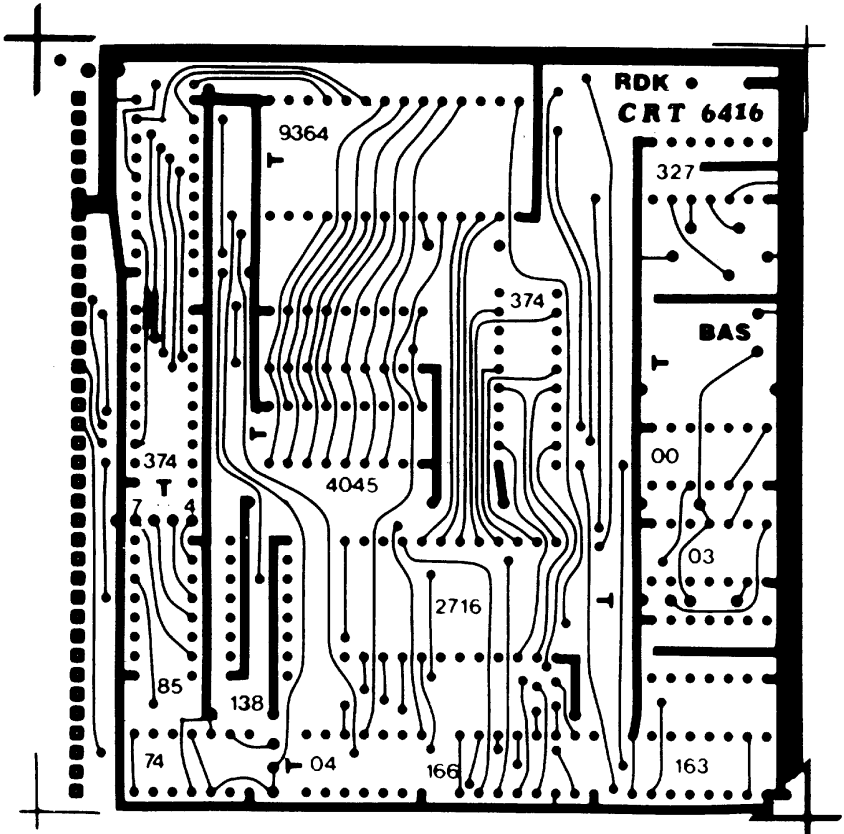


Abb. 5.3.13
Bestückungsseite
der CRT-Karte

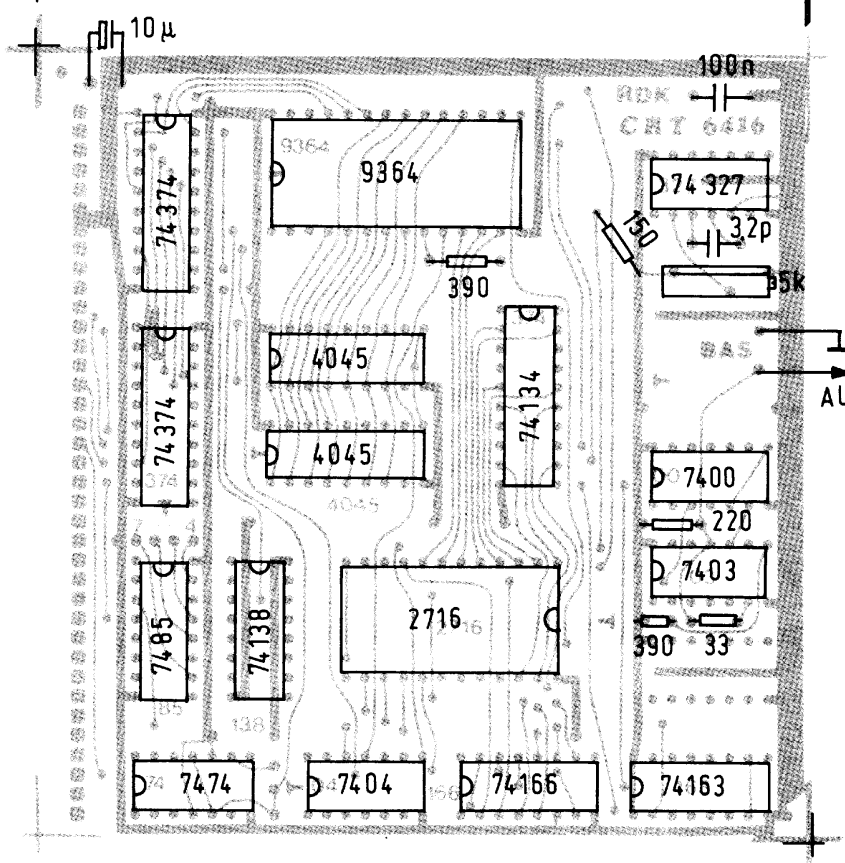


Abb. 5.3.14
Bestückungsplan
der CRT-Karte

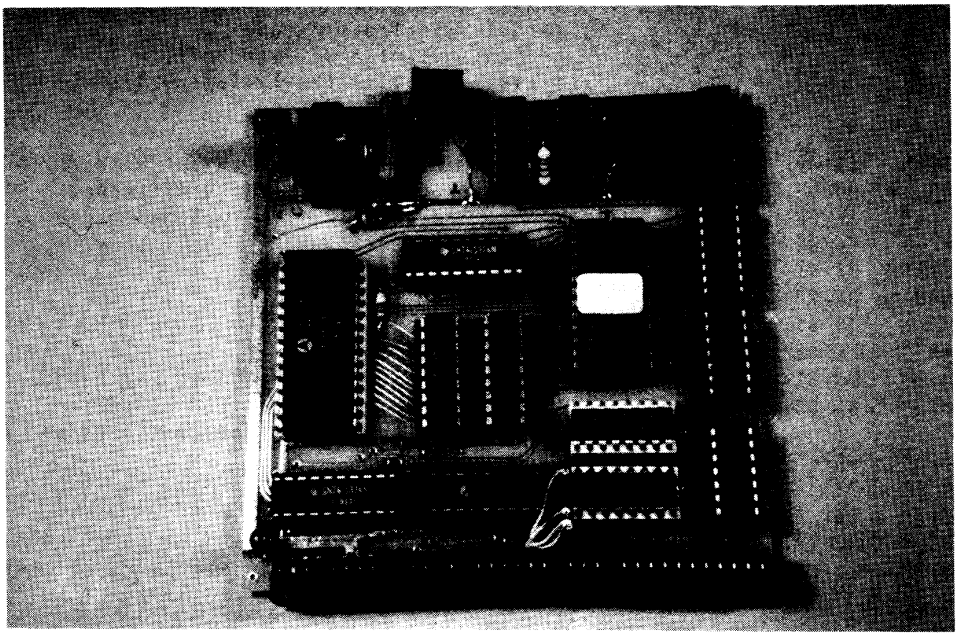


Abb. 5.6 CRT-Karte

Nun zum Aufbau. *Abb. 5.3.12* zeigt die Lötseite der CRT-Karte, *Abb. 5.3.13* zeigt die Bestückungsseite und *Abb. 5.3.14* zeigt den Bestückungsplan.

Beim Aufbau wird mit der Bestückung aller passiven Bauteile begonnen. Dabei sind Anstelle der ICs unbedingt Sockel zu verwenden, um den nachfolgenden Test zu vereinfachen. Als Adresse wird 60h (61h) eingestellt. Die Brücken sind im Schaltplan eingezeichnet.

1. Test. Nach dem Einschluß an die Versorgungsspannung wird kontrolliert, ob alle ICs die richtige Spannung erhalten. Insbesondere ist die Versorgungsspannung des Controller-Chip zu überprüfen. An Pin 28 müssen +5V liegen und an Pin 14 muß Masse sein.

2. Einsetzen des ICs 7474 (oder LS). Betrieb zusammen mit der CPU-Karte (SBC- oder große CPU + ROMRAM) sowie entweder der Keyboard-Karte KEY oder das seriellen Interface. Man muß an Pin 1 des CRT-Controllers eine Frequenz von 1 MHz anliegen. Es erfolgt das einsetzen der Bausteine 74LS374, 74LS138, 74LS85 und 7404 (hier kein LS-Typ verwenden), wobei der 7404 neben dem 7474 liegt und den Inverter I1 beinhaltet.

3. Eingabe des Programms nach *Abb. 5.3.15*. Wurde die serielle Karte verwendet, so kann das Programm über ein externes Datensichtgerät auch in den internen RAM-Speicher eingegeben werden, z. B. auf Adresse 1200h beginnend. Dabei müssen aber die Sprungadressen um den Wert 1200h erhöht werden. Bei der Eingabe ist darauf zu achten, daß im Listing MSB und LSB die Adressen in lesbarer Form ausgegeben werden, bei der Eingabe muß aber zuerst LSB und dann MSB eingegeben werden. Ohne serielle Karte muß

```
                        ;*********************************
                        ;* CRT-9364 Testprogramm         *
                        ;* Continuierliches Einschreiben *
                        ;*********************************

0000'                   start:
0000'   3E 41                   ld a,'A'        ;A auf Bildschirm
0002'   D3 60                   out (60h),a     ;auf datenport
0004'   3E 0F                   ld a,1111b      ;zeichenausgabe
0006'   F6 80                   or 10000000b
0008'   D3 61                   out (61h),a
000A'   E6 7F                   and 01111111b
000C'   D3 61                   out (61h),a
000E'   F6 80                   or 10000000b
0010'   D3 61                   out (61h),a     ;ende zeichenaus
0012'   01 02B3                 ld bc,8300/12   ;8.3 ms warten
0015'                   wait:                   ;wartet 12ys pro bc
0015'   0B                     dec bc
0016'   79                     ld a,c
0017'   B0                     or b
0018'   C2 0015'               jp nz,wait
001B'   C3 0000'               jp start        ;naechstes zeichen
                        ;
```

Abb. 5.3.15 Testprogramm CRT-Karte

das Programm in einem EPROM abgelegt werden und nach RESET gestartet wird. Das Programm schreibt fortwährend den Buchstaben ‚A' auf den Bildschirm. Damit können alle Funktionen des CRTs getestet werden.

4. Die Dekodierung wird getestet. Nach Start des Programms muß das Oszillogramm nach *Abb. 5.3.16* vorliegen. Dort sind übrigens zwei Oszillogramme überlagert. Ein Kanal wird mit Pin 15 des 74LS138 verbunden und dient auch als Trigger. Nun kann mit dem

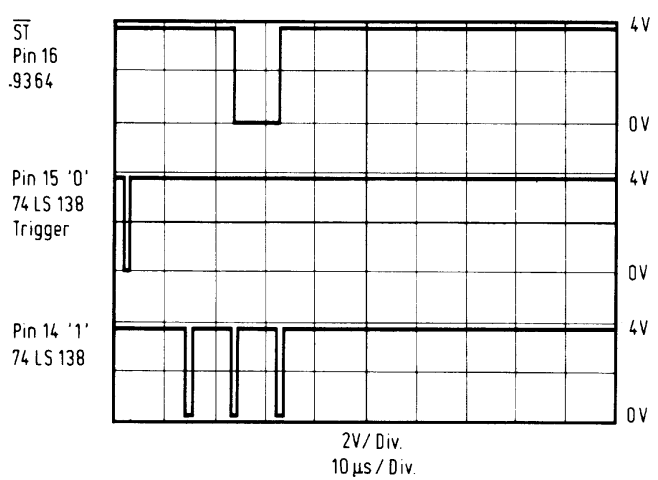


Abb. 5.3.16 Oszillogramm CRT-Karte

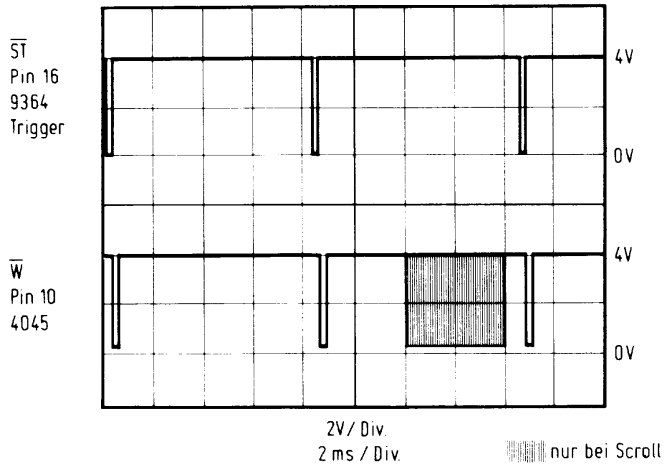


Abb. 5.3.17 Oszillogramm CRT-Karte

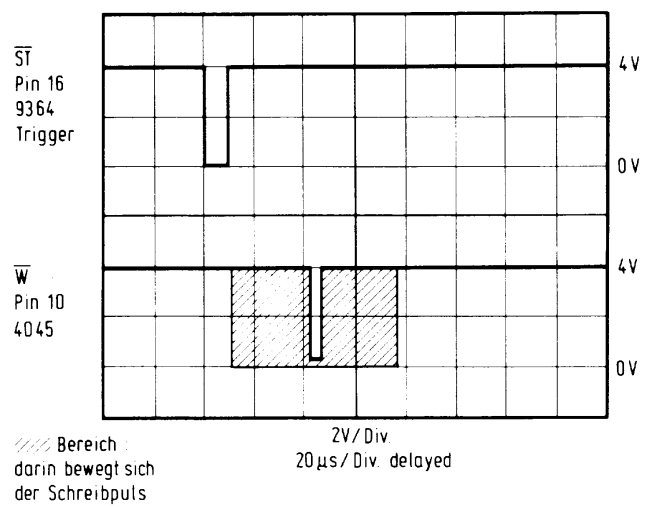


Abb. 5.3.18 Oszillogramm CRT-Karte

anderen Kanal entweder das Signal -ST oder der Ausgang ‚1' (Pin 14) des Dekoders beobachtet werden. Diejenigen, die nur einen Kanal besitzen, verbinden den Ausgang des Dekoders 74LS138 Pin 15 mit dem Trigger und beobachten die anderen Signale nacheinander mit einem Kanal.

5. Einsetzen der restlichen Bausteine. Nun gibt es eine Vielzahl zu messen. Zum Einen kann jetzt einmal der Monitor an den CRT-BAS-Ausgang angeschlossen werden. Es müßte bereits ein stehendes Bild entstehen. Mit dem Trimmer TR1 muß die horizontale Zeichenfrequenz so abgeglichen werden, daß die einzelnen Buchstaben jeweils gerade in eine Zeile passen. Sollte das Signal nicht erscheinen, so fahren wir mit dem Test fort.

6. *Abb. 5.3.17* zeigt ein Oszillogramm für den Einschreibevorgang. An dem Pin 10 (-W) des Speichers 4045 erscheint nach jedem Einschreibebefehl (Pin 16 -ST an 9364)

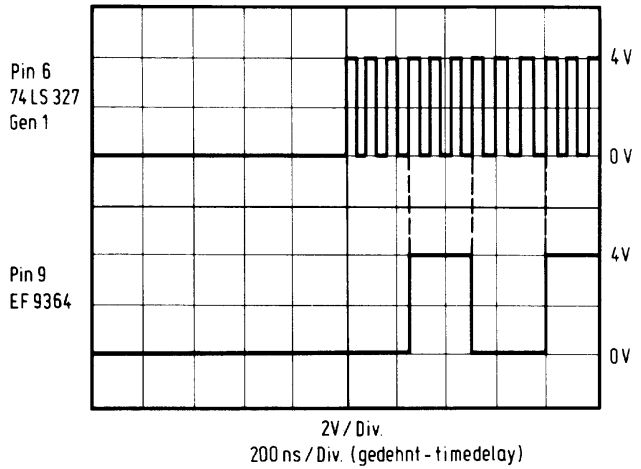


Abb. 5.3.19 Oszillogramm CRT-Karte

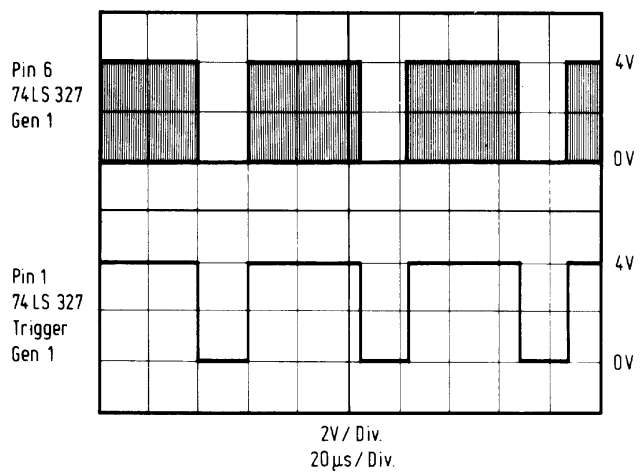


Abb. 5.3.20 Oszillogramm CRT-Karte

ein Schreibsignal. Wird eine Zeile gescrollt (verschieben des Bildinhalts nach oben zusätzlich mit einem Löschvorgang der unteren Zeile), so kommt kurz eine Pulsgruppe ins Bild.

7. *Abb. 5.3.18* zeigt eine Ausschnittvergrößerung. Dieses Bild ist auf dem Scop nur schwer zu bekommen und hängt von den Möglichkeiten des jeweiligen Scops ab. Gelingt es dem Leser, es zu reproduzieren, so läßt sich erkennen, daß der Einschreibevorgang des CRT-Bausteins nicht immer zum gleichen Zeitpunkt nach dem Schreibbefehl erscheint, sondern in einem kleinen Bereich wandert.

8. Test des Oszillators für den Zeichen und Punkttakt. *Abb. 5.3.19* zeigt das Oszillogramm. Dieses Bild muß auch ohne Testprogramm entstehen. Der Oszillator schwingt nur in der Zeit in der der sichtbare Teil des Bildes durchlaufen wird. An Pin 9 des 9364 muß der Oszillatortakt heruntergeteilt sein. Dies ist der Zeichentakt.

9. *Abb. 5.3.20* zeigt nochmals den Start-Stop-Bereich des Oszillators etwas deutlicher. Liegt das Signla an Pin 1 des Generators 74LS327 auf einem 0-Pegel, so schwingt der Oszillator nicht. Pin 1 ist die Versorgungsspannung des Oszillators. Daher ist es auch ratsam, einen Pull-up-Widerstand an dem Eingang (ca. 150 Ohm bis 470 Ohm) zu verwenden, da die Ausgangsfrequenz auch vom Pegel dieses Pins abhängt. Die Oszillatorfrequenz liegt ungefähr bei 11 MHz.

10. *Abb. 5.7* zeigt das BAS-Signal wie es nun am Eingang des Monitors bei einem gefüllten Bildschirm (z. B. Buchstabe „A", oder nach dem Einschalten ohne Monitorprogramm) vorliegt. Deutlich sind die Impulsgruppen der einzelnen Bildzeilengruppen zu sehen. Eine Gruppe besteht dabei aus mehreren Einzelzeilen, die zusammen die Höhe eines Zeichens ausmachen.

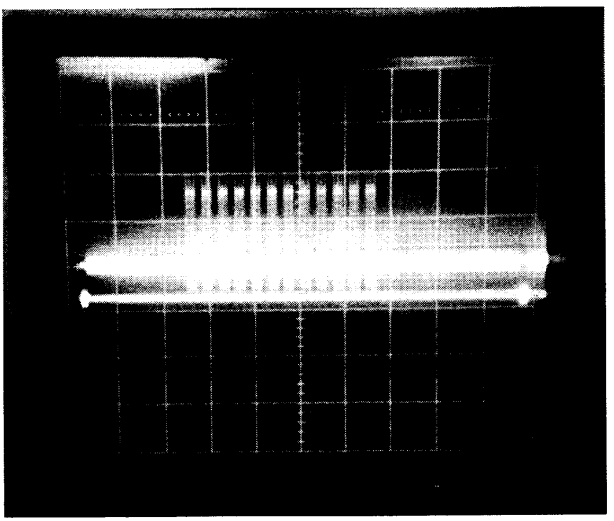


Abb. 5.7 Oszillogramm CRT-Karte

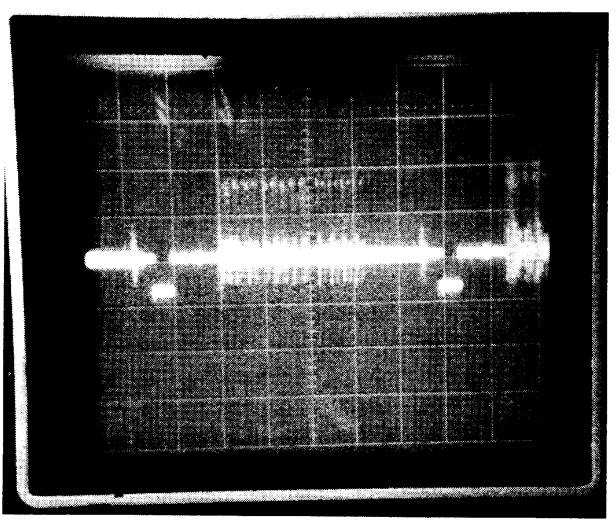


Abb. 5.8 Oszillogramm CRT-Karte

11. *Abb. 5.8* zeigt das Oszillogramm einer einzelnen Zeile. Dabei müssen dort Zeichen vorhanden sein. Dazu wurde hier mit einem verzögerten Triggersignal gearbeitet und getriggert wurde mit Hilfe des gesamten BAS-Signals aber auf dem vertikalen Abschnitt. Wird nun auf das Zeilensignal ohne Verzögerung getriggert, so ist die Zeile im Prinzip auch so erkennbar, aber der Informationsinhalt nicht so klar zu sehen. Erscheint das Signal wie angegeben, so muß auf dem Monitor bereits ein Bild sichtbar sein.

12. Ist die Tastatur nach dem vorherigen Abschnitt aufgebaut und stimmt die Schalterstellung, so müssen sich nun bei Vorhandensein des Monitorprogramms Befehle eingeben lassen und auf dem Bildschirm erscheinen. Nach dem RESET muß der Bildschirm zunächst gelöscht werden und dann die Monitormeldung in der oberen Hälfte erscheinen. Wird eine Taste der Tastatur betätigt, so muß der Befehl auf dem Bildschirm ausgegeben werden. Test z. B. Eingabe der Sequenz:

D100 200cr

cr steht dabei für die Betätigung der Taste carriage return (Wagenrücklauf).

Auf dem Bildschirm muß der Inhalt des Speicherbereichs 100 bis 200 in hexadezimaler Schreibweise ausgegeben werden, siehe auch den Softwareabschnitt zum Monitorprogramm.

5.4 Aufbau eines EPROM-Programmierers

Um nun endlich dem Problem der mühsamen Eingabe von Programmieren mit dem Dil-Schaltern zu entgehen, wird als nächstes eine EPROM-Programmier-Karte aufgebaut. Im Prinzip arbeitet diese Schaltung genauso wie die in Abb. 4.2.5, nur daß hier der Prozessor die Steuerung übernimmt. Mit der Karte können sowohl 2716, 2732 und 2764 der Intel-Familie (und kompatibler) programmiert werden. Das Steuerprogramm befindet sich bereits in unserem Monitor. Da uns nun die CPU zum Test mit einem Datensichtgerät oder der CRT-Karte + Tastatur zur Verfügung steht, wird es wesentlich einfacher sein, neue Karten wie diese aufzubauen. Testprogramme die evtl. benötigt werden, können im RAM abgelegt werden.

Abb. 5.4.1 zeigt die Schaltung des EPROM-Programmierers. Die Ausgabe der Daten erfolgt über das Latch LT1. Da es auch möglich sein muß, vom EPROM Daten zurückzulegen, gibt es den Buffer B1. Ferner muß der Ausgang des Zwischenspeichers LT1 beim Einlesen in den TRI-State-Zustand versetzt werden können. Dies geschieht über den Freigabe-Eingang -OE. Der Freigabe erfolgt per Software über Latch LT3 mit dem auch andere Steuersignale und ein Teil der höherwertigen Adressen ausgegeben werden. Mit dem Treiber B2 kann der Zustand des Monoflops abgefragt werden. Das Monoflop wird zur Erzeugung des Programmierpulses verwendet. Aus Sicherheitsgründen wird dies nicht durch Software ausgeführt, da sonst bei Fehlbedienungen eine Zerstörung des zu programmierenden EPROMs erfolgen könnte. Das Monoflop wird mit dem Trimmer Tr1 auf eine Zeitkonsante von 1ms eingestellt. Für die Programmierung eines Bytes sind aber 50ms nötig.

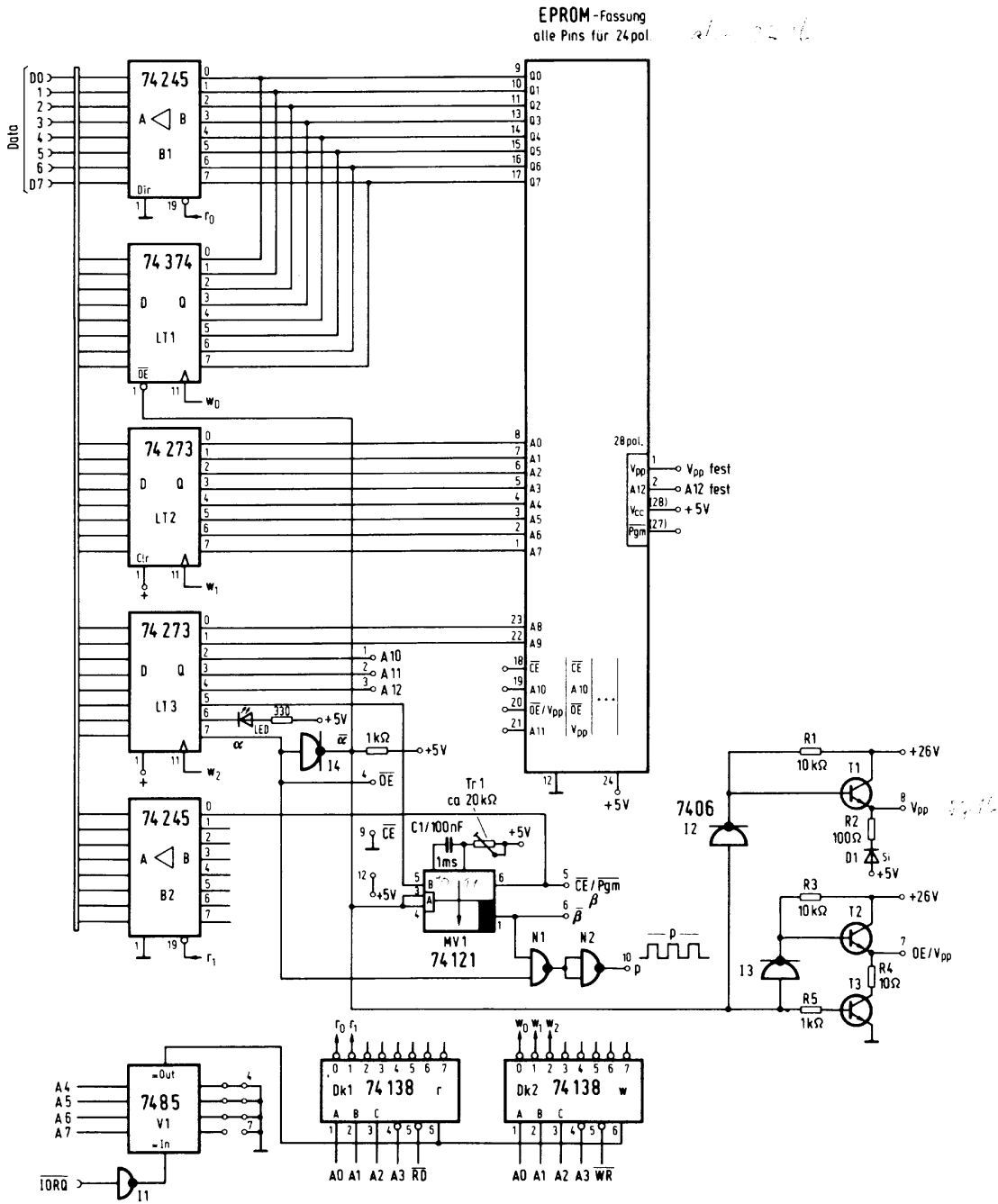


Abb. 5.4.1 Schaltung der EPROM-Programmier-Karte

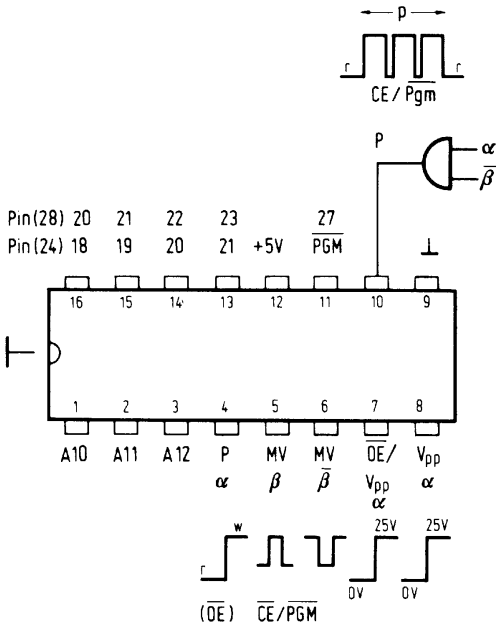


Abb. 5.4.2 Belegung des Dil-Steckers

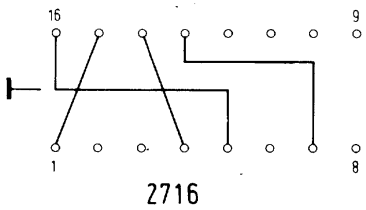


Abb. 5.4.3 Dil-Stecker für 2716

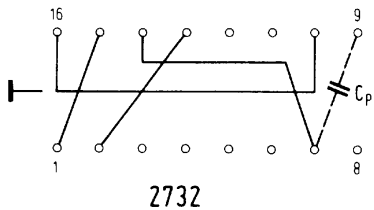


Abb. 5.4.4 Dil-Stecker für 2732

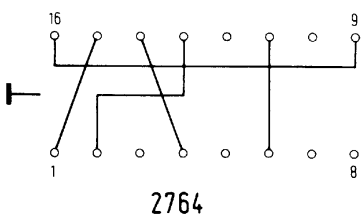


Abb. 5.4.5 Dil-Stecker für 2764

	ena (α)	led	trg (β)
Read	Ø	1	Ø
Programm	1	Ø	Ø
	1	Ø	1
	1	Ø	Ø

Abb. 5.4.6 Read//Programmier Sequenzen

	7	6	5	4	3	2	1	Ø	
8ØH	D7	D6	D5	D4	D3	D2	D1	DØ	Input
8ØH	D7	D6	D5	D4	D3	D2	D1	DØ	Output
81H	x	x	x	x	x	x	x	busy 1=warten	Input
81H	A7	A6	A5	A4	A3	A2	A1	AØ	Output
82H	ena (OE)	led 0=an	trg	A12	A11	A1Ø	A9	A8	—

Abb. 5.4.7 Belegung der Ports

Diese Zeit wird durch eine Programmierschleife im Monitorprogramm erzeugt, dabei wird das Monoflop 50 Mal getriggert. Als Programmierspannung werden 26V zugeführt. Diese Programmierungsspannung wird mit T1, T2 und T3 per Software an das EPROM geschaltet. Die Ausgänge des Monoflops, sowie der von den Programmierspannungen sind dazu auf einen Dil-Sockel geführt. Darauf kann ein Dil-Stecker gesteckt werden, der die jeweilige Konfiguration für das zu,programmierende EPROM enthält. *Abb. 5.4.2* zeigt die Belegung des Dil-Sockels. *Abb. 5.4.3* zeigt die Belegung des Dil-Steckers bei der Programmierung eines 2716. *Abb. 5.4.4* zeigt die Belegung für den 2732 und in *Abb. 5.4.5* ist die Belegung für den 2764 gegeben.

Abb. 5.4.6 zeigt die Bitbelegung zur Steuerung der Karte. Bei einem Lesevorgang muß das Enable-Bit auf 0 liegen, -led wird auf 1 gelegt, damit leuchtet die LED nicht und trg hat ein 0-Signal. Beim Programmieren wird die untere Bitfolge nacheinander angelegt und damit ein Puls von 1ms Dauer ausgelöst. In *Abb. 5.4.7* ist nochmals zusammenfassend die Bedeutung der einzelnen Ports wiedergegeben.

Abb. 5.4.8 zeigt die Lötseite der Platine, *Abb. 5.4.9* zeigt die Bestückungsseite und in *Abb. 5.4.10* ist der Bestückungsplan abgebildet.

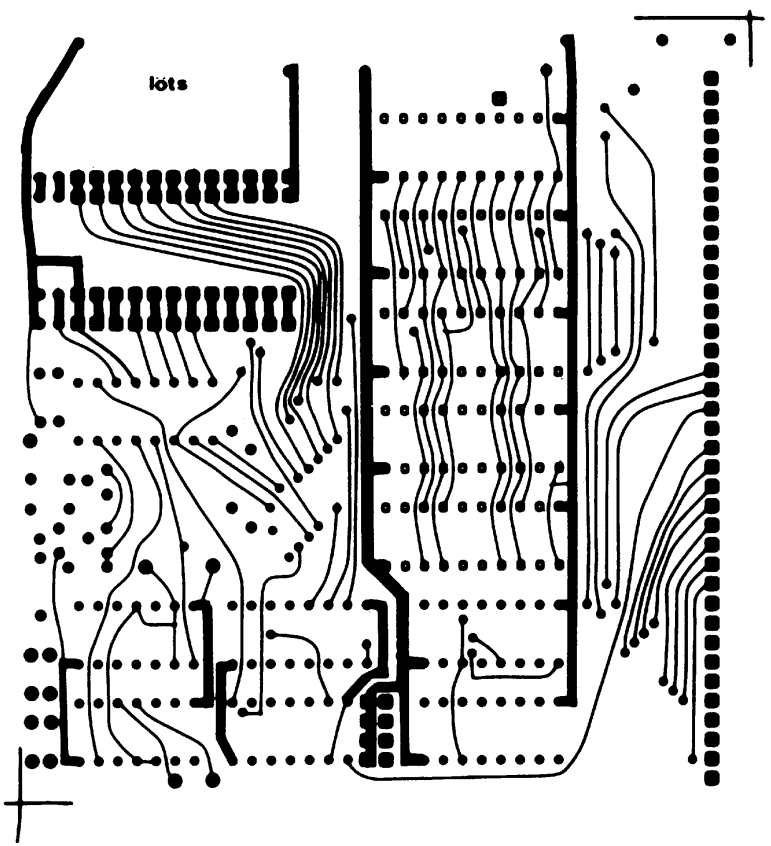


Abb. 5.4.8 Lötseite der ERPOM-Karte

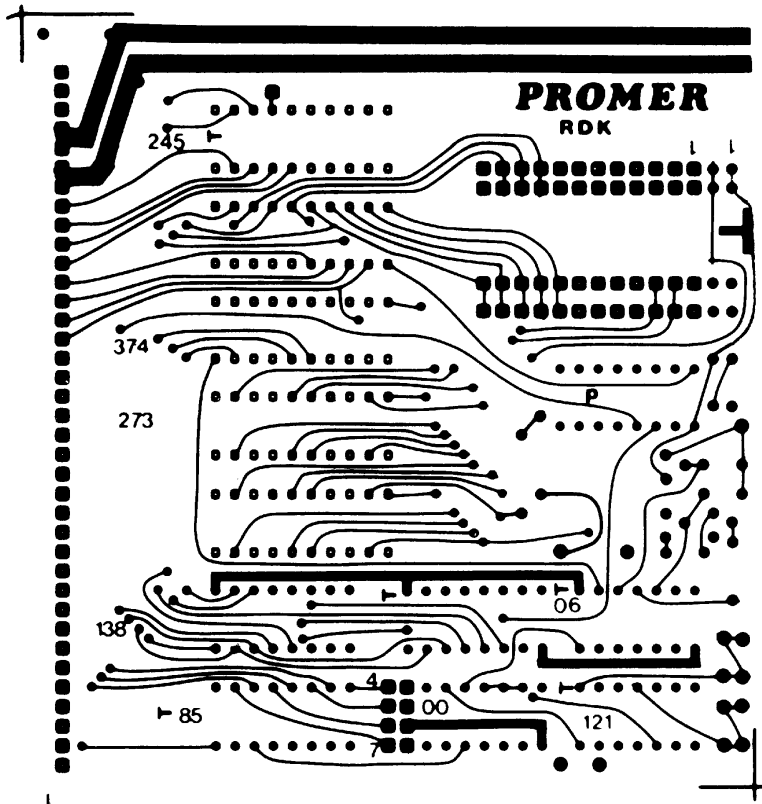


Abb. 5.4.9
Bestückungsseite
der EPROM-Karte

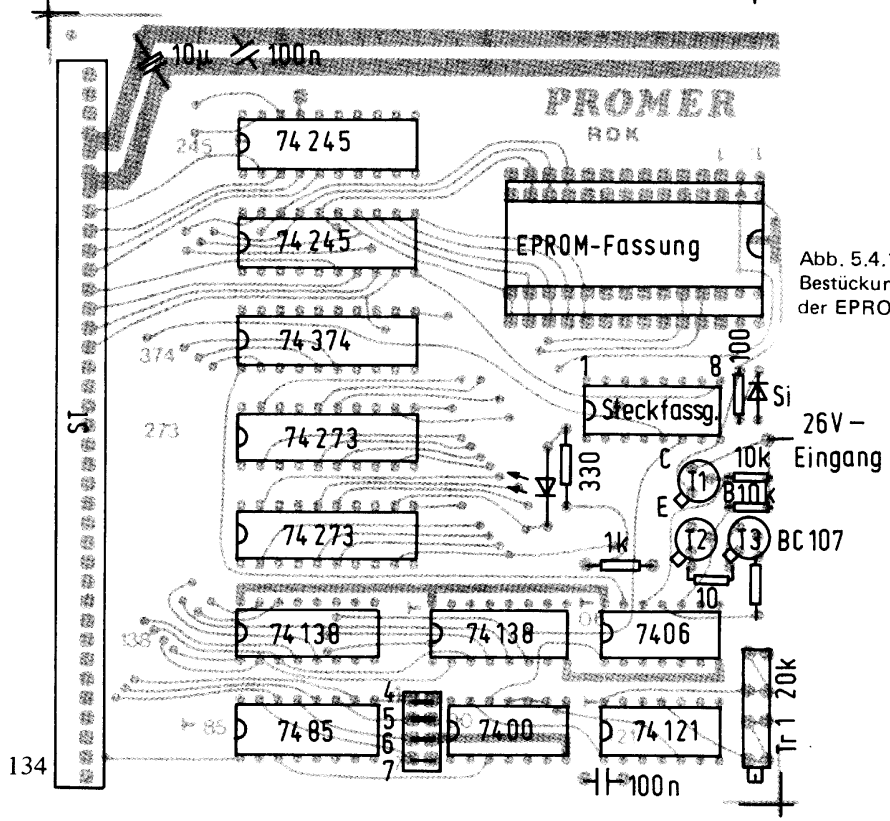


Abb. 5.4.10
Bestückungsplan
der EPROM-Karte

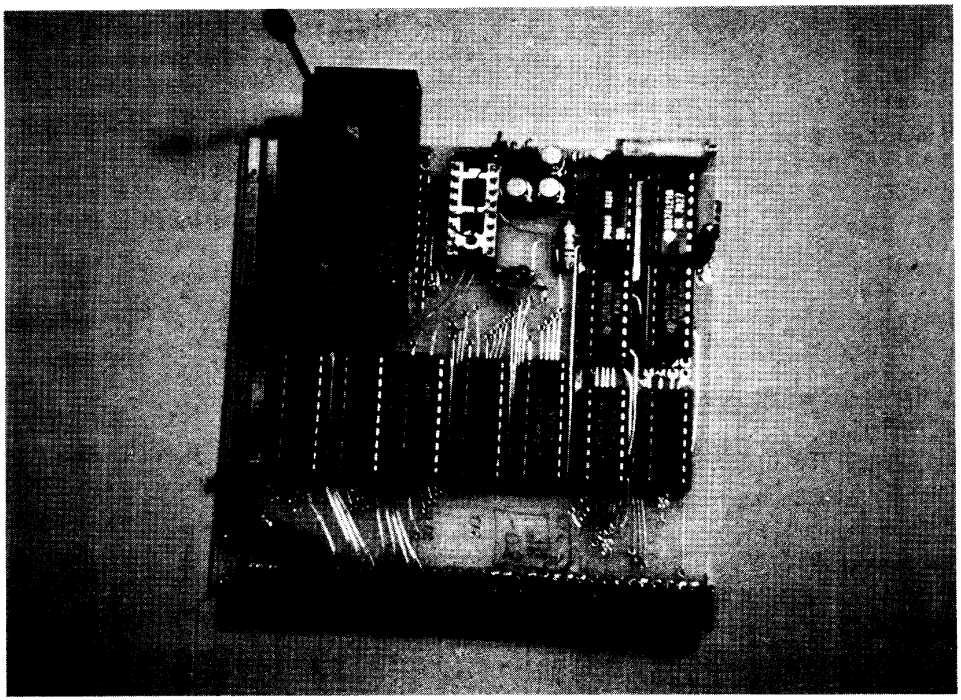


Abb. 5.9 EPROM-Karte

Nun kann mit dem Zusammenbau begonnen werden. Es werden wieder überall Sockel verwendet. Bei dem EPROM-Sockel kann eine spezielle Programmierfassung verwendet werden, wie sie in *Abb. 5-9* abgebildet ist. Dabei ist es zweckmäßig, gleich eine 28polige Fassung zu verwenden, auch wenn nur 24polige EPROMs programmiert werden sollen. Dann können später auch die 2764-EPROMs programmiert werden. Die Programmierfassung kann auch über Leitungen (maximal 30 cm) mit der Platine verbunden werden, um sie z. B. bei der Verwendung eines Gehäuses an die Frontplatte zu führen. Das gleiche gilt für den Dil-Sockel, der die Verdrahtung für den jeweiligen EPROM-Typ tragen soll. Alle passiven Bauteile werden bestückt.

1. Test Versorgungsspannungsanschluß sowie Computeranschluß. Dabei wird die 26V-Programmierspannung bis zuletzt nicht angeschlossen. Alle Messungen können dann gefahrlos (für die ICs) ausgeführt werden. Es muß nun gemessen werden, ob die Versorgungsspannung auch ankommt.

2. Einsetzen aller ICs, außer des Dil-Steckers und des EPROMs. Die Karte wird von Adresse 80 h an adressiert. Dazu sind die Brücken gemäß Schaltbild und Bestückungsplan einzulöten. *Abb. 5.4.11* zeigt ein kleines Testprogramm. Es kann mit Hilfe des Monitors z. B. auf Adresse 1200 h eingegeben werden. Damit wird die Dekodierung getestet. *Abb. 5.4.12* zeigt das Oszillogramm der einzelnen Dekoderausgänge, die an die verschiedenen Latches und Buffer führen. Die Signale wurden nacheinander gemessen, dabei bleibt das Signal von LT1 am Trigger und dient daher als Bezugspunkt.

```
                        ;***********************************
                        ;* Prom Programmierer Test Dekoder *
                        ;***********************************

0000'   AF              start:  xor a           ;ac:=0
0001'   D3 80                   out (80h),a     ;daten
0003'   D3 81                   out (81h),a     ;adresse lsb
0005'   D3 82                   out (82h),a     ;adr msb+bef
0007'   DB 80                   in a,(80h)      ;daten
0009'   DB 81                   in a,(81h)      ;ready
000B'   18 F3                   jr start        ;wiederholen
```

Abb. 5.4.11 Testprogramm EPROM-Programmiergerät

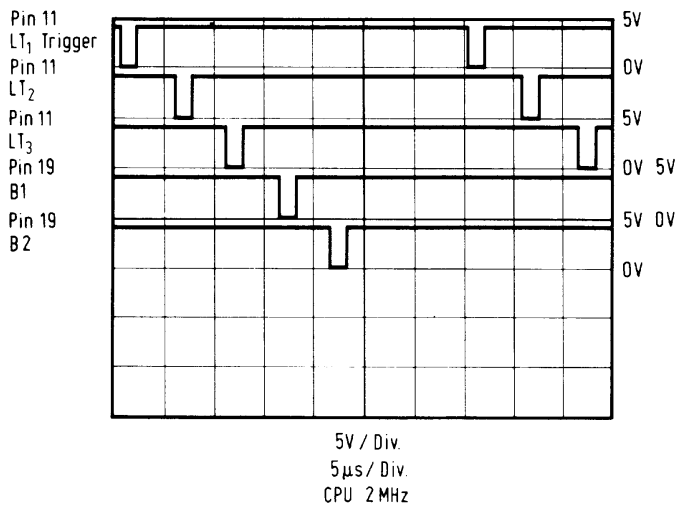


Abb. 5.4.12 Oszillogramm EPROM-KARTE

```
                        ;***************************************
                        ;* Prom Programmierer Adressdatatest *
                        ;***************************************

0000'   3E 80           start:  ld a,10000000b  ;enable promer
0002'   D3 82                   out (82h),a
0004'   D3 80           loop:   out (80h),a     ;nur beide lsbs
0006'   D3 81                   out (81h),a     ;adr
0008'   3C                      inc a           ;next wert
0009'   18 F9                   jr loop
```

Abb. 5.4.13 Testprogramm der EPROM-Karte

3. Nun kann ein kleiner Vortest mit dem Monitor direkt ausgeführt werden. Dazu werden die Befehle

Q082 80cr
Q080 55cr

eingegeben. Nach der Eingabe des ersten Befehls muß die LED aufleuchten.

Mit QI80cr muß der Wert 55h ausgegeben werden. Dann Eingabe von

Q080 AAcr

Nach QI80cr muß AAh ausgegeben werden. Damit sind die Datenleitungen getestet. Bei der Eingabe von

Q082 40cr

muß die LED wieder ausgehen.

4. Nun kann ein kleines Testprogramm die Ausgabe nochmals kontrollieren. *Abb. 5.4.13* zeigt das Programm. Es wird auch wieder auf Adresse 1200h eingegeben und gestartet. *Abb. 5.4.14* zeigt ein Oszillogramm zu diesem Testprogramm. Nach jedem Einschreibepuls muß der Datenausgang D0 wechseln. Etwas verzögert wechselt die Adresse A0, wie in *Abb. 5.4.15* gezeigt ist. Die Adressen und Daten (A0..A7, D0..D7) haben mit

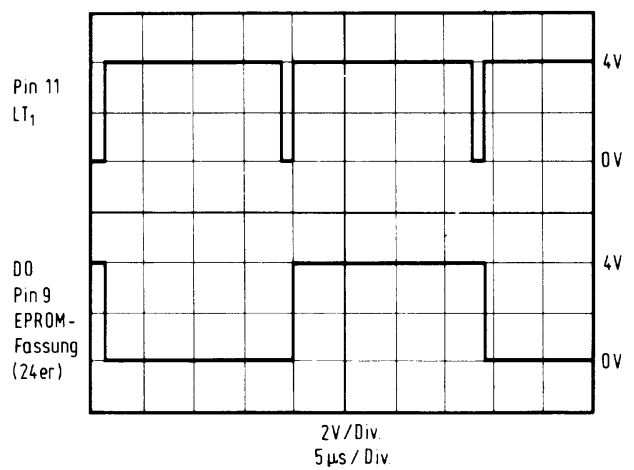


Abb. 5.4.14
Oszillogramm EPROM-Karte

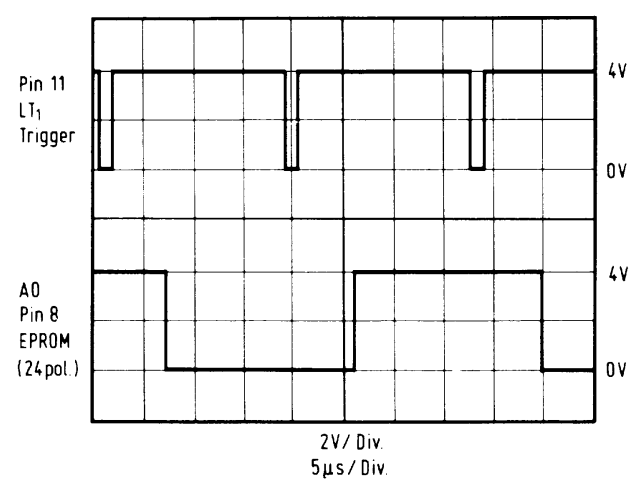


Abb. 5.4.15
Oszillogramm EPROM-Karte

aufsteigender Nummer jeweils die halbe Frequenz. Mit diesem Testprogramm wird LT3 nicht überprüft.

5. Nun muß das Monoflop eingestellt werden. Dazu wird dem Monitor der Befehl zur Programmierung gegeben:

PW1000 1FFFcr

(cr steht für die Eingabe der Taste CR bzw. RETURN)
z .B. für den 4K Bereich 1000h bis 1FFFh. Dann Eingabe von Y nach der Frage PRG YES=Y. Mit dem Scop wird an Pin 1 des Monoflops gemessen. *Abb. 5.4.16* zeigt das Oszillogramm. Mit dem Trimmer Tr1 muß nun so abgeglichen werden, daß die Zeitdauer zwischen zwei Pulsen 1ms beträgt. Dabei ist an dem Ausgang -Q des Monoflops ein Low-Puls mit dieser Breite entscheidend. Nach einiger Zeit meldet sich der Monitor mit PRG

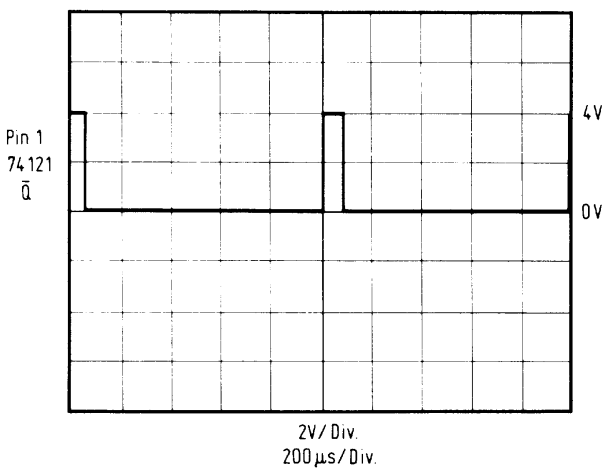


Abb. 5.4.16
Oszillogramm EPROM-Karte

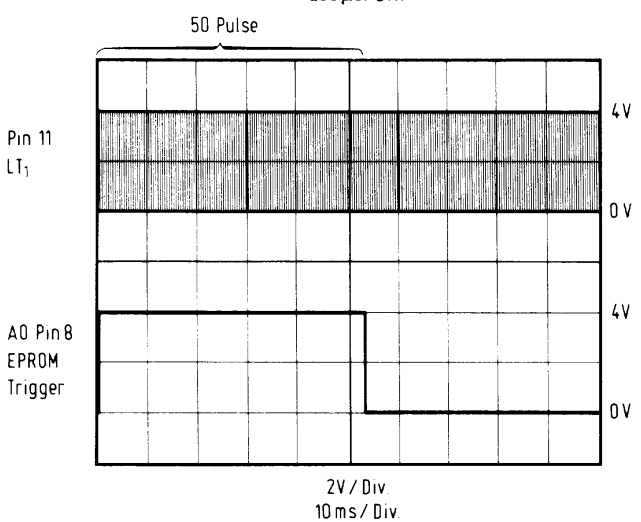


Abb. 5.4.17
Oszillogramm EPROM-Karte

ERR, da ja noch kein EPROM programmiert wurde. Wird aber z. B. nur der Wert FFh programmiert, so erfolgt die Meldung BURNED.

6. Da für den Programmiervorgang 50 ms benötigt werden, erfolgt ein Adreßwechsel auf A0 erst nach 50 Programmierpulsen *Abb. 5.4.17* zeigt das Oszillogramm.

7. Nun wird der zum gewünschten EPROM gehörige Dil-Stecker verlötet und eingesteckt. Ferner wird die Programmierspannung angeschlossen. Sie muß nun bei dem Programmierbefehl an das entsprechende Pin des EPROMs automatisch angeschaltet werden. Die Pulsformen sind so wie schon in Kapitel 6 besprochen.

8. Bei einem 2732 (Intel). Eingabe des Befehls

PW1000 1FFFcr

Danach fragt der Monitor PRG YES=Y und die Eingabe des Buchstabens Y veranlaßt die Programmierung. Zum Programmieren des gesamten EPROMs muß natürlich auch der RAM-Platz des Systems vorhanden sein. Bei den EPROMs kann also nur mit der großen CPU+RAM/ROM-Karte gearbeitet werden, wenn das ganze EPROM belegt werden soll, doch muß der Speicher auch vorhanden sein, um diese Programme zu entwickeln.

Nach dem Beenden der Programmierung darf nicht die Meldung PRG ERR erscheinen, sondern die Meldung BURNED. Dann ist das EPROM richtig programmiert. Da der Speicher beim SBC nicht zur Aufnahme der ganzen 4K reicht, werden wir nun einen Teil der programmierten Daten ins RAM lesen.

9. Das Einlesen geschieht mit dem Befehl PR. Beispiel:

PR1200 12FFcr

In dem Bereich 1200h wurden aufsteigend die ersten 256 Daten des EPROMs abgelegt. Bei einem leeren EPROM muß der Wert FFh eingelesen werden.

Damit ist der Aufbau und Test beendet. Sollten sich bei der Programmierung Schwierigkeiten ergeben haben, so kann das auch an dem EPROM liegen, z. B. führt ein nicht gelöschtes EPROM auch zu PRG ERR oder die Programmierspannung stimmt nicht oder das verwendete EPROM hat andere technische Daten (z. B. Intel 2716 und Texas 2716 sind verschieden). Die Pulsformen sind am besten mit den abgedruckten im Datenblatt des jeweiligen Herstellers zu vergleichen. Dabei ist wichtig, daß wir hier mit 50 mal 1ms Pulsen anstatt mit einem 50 ms Puls programmieren, was aber nicht stört.

5.5 Universal IO-Karte

Um die Möglichkeit zu bieten, eigene Peripherieschaltungen oder Bausteine anzuschließen, gibt es die IO-Karte. Darauf befinden sich zwei Ausgabeports mit je 8 Bit sowie zwei Eingabe-Ports, die auch bidirektional verwendet werden können. Auf einem freien Lochrasterfeld kann eine eigene Schaltung aufgebaut werden. Dabei braucht man sich nicht mehr um die Dekodierung zu kümmern, da das gesamte Businterface schon fertig vorbereitet ist.

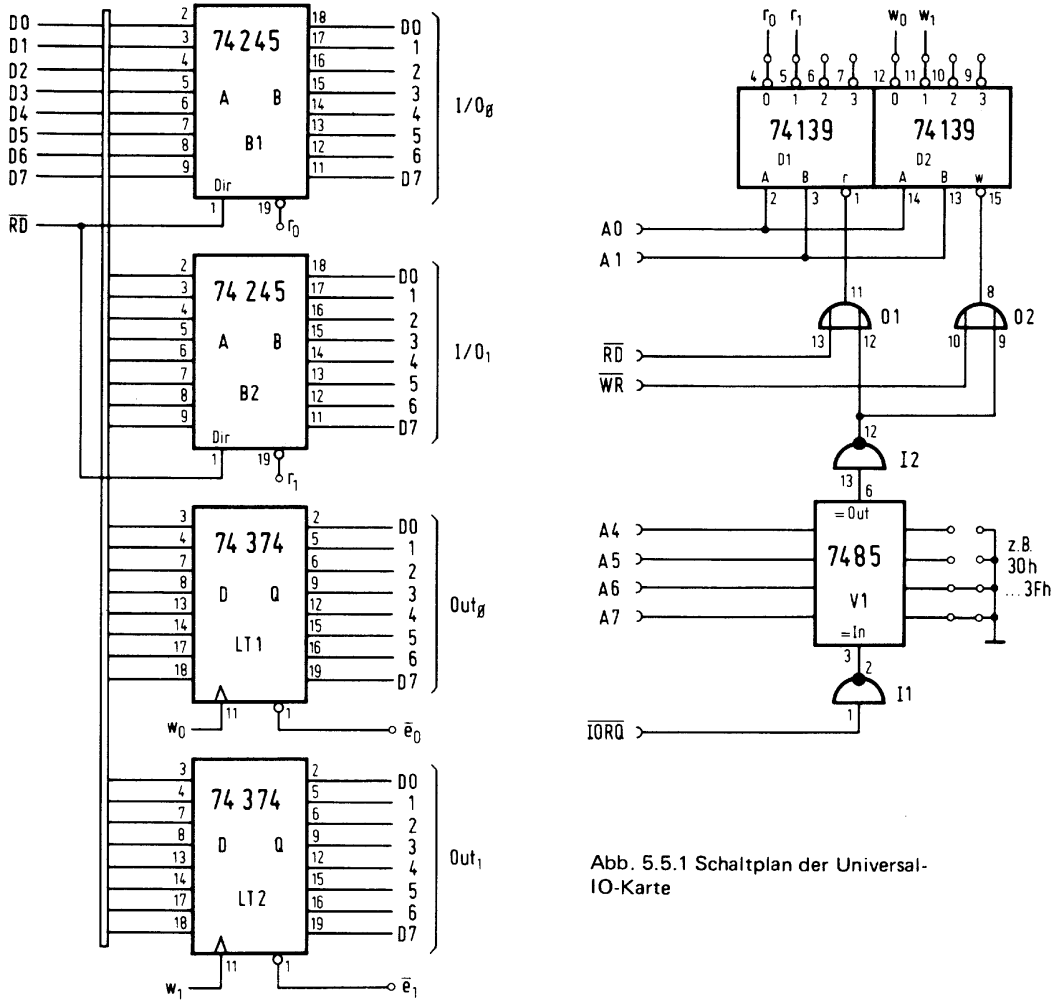


Abb. 5.5.1 Schaltplan der Universal-IO-Karte

Abb. 5.5.1 zeigt die Schaltung. Als Ausgabeports dienen die Schaltkreise 74LS374, also zwei Latches mit TRI-State-Ausgang. Als universale Eingabeports sind 74LS245 verwendet. Sie können auch bidirektional betrieben werden. Die Adressdekodierung übernimmt der Baustein 74LS139, in dem zwei einzelne Dekoder enthalten sind. Am Vergleicher V1 läßt sich die Adresse der IO-Karte einstellen. Dabei kann die Karte 16 IO-Adressen belegen, beginnend mit der eingestellten. Die unteren beiden Adressen sind für die vorhandenen IO-Ports verwendet. Die Ein- und Ausgänge der Ports führen auf die Verdrahtungsfläche. *Abb. 5.5.2* zeigt die Zuordnung der einzelnen Bits zu den Lochrasterplätzen.

Abb. 5.5.3 zeigt die Lötseite der Platine, *Abb. 5.5.4* zeigt die Bestückungsseite und *Abb. 5.5.5* zeigt den Bestückungsplan für den Fall, daß alle IO-Ports verwendet werden sollen. In *Abb. 5.10* ist die Platine mit einer Beispielanwendungsverdrahtung abgebildet.

Der Aufbau erfolgt wie auch bei den anderen Karten zunächst mit dem Einlöten der Sockel. Dabei empfiehlt es sich erst einmal, nicht die eigene Schaltung aufzubauen. Dann kann am Anfang der Standard-Test erfolgen.

1. Test. *Abb. 5.5.6* zeigt ein kurzes Testprogramm. Es spricht alle Ports an. Damit kann die Dekodierung und Adressierung der Karte getestet werden. Das Programm wird z. B. ab Adresse 1200h eingegeben.

2. Es wird an den Ausgängen der Dekoder gemessen. Der Ausgang des Dekoders D2 Pin 12 wird als Trigger verwendet. Dann wird mit dem zweiten Kanal nacheinander Pin 11 von D2, dann Pin 4 von D1 und an Pin 5 von D1 gemessen (alle am selben IC 74LS139).
Es muß ein zeitlich versetztes Pulsraster ähnlich zu Abb. 5.4.12 entstehen.

3. Nun kann der Bustreiber B1 oder auch B2 außer als reiner Eingabeport, auch als bidirektionaler Treiber arbeiten. Dazu ist in der Schaltung bereits der Richtungseingang mit dem -RD-Signal verbunden. Wird die bidirektionale Betriebsweise gewünscht, so muß

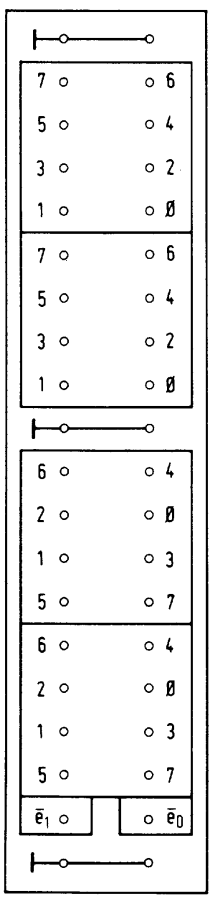


Abb. 5.5.2 IO-Anschlüsse

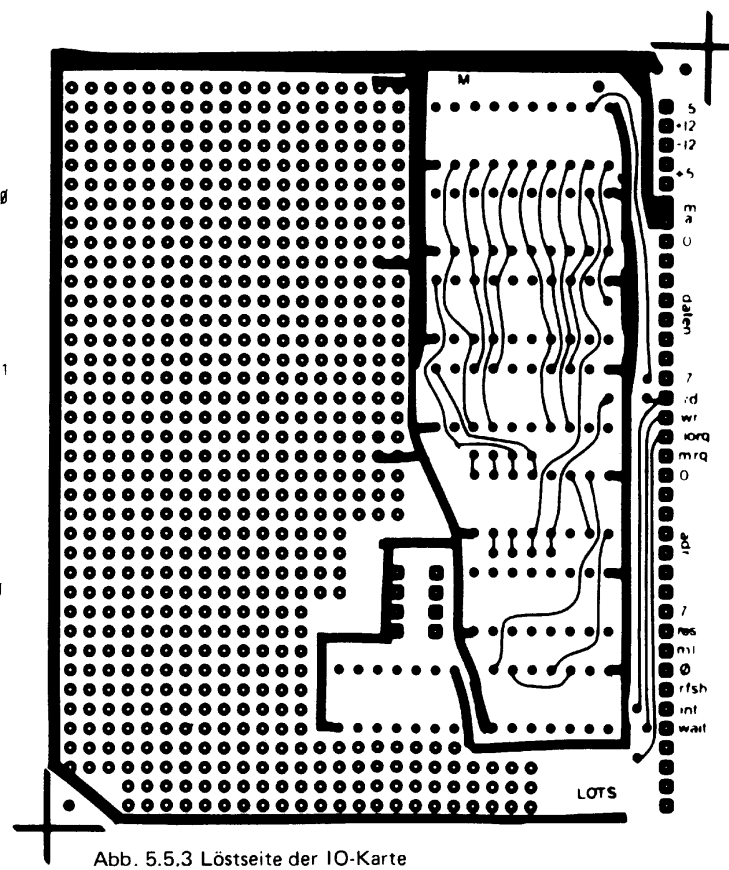


Abb. 5.5.3 Löstseite der IO-Karte

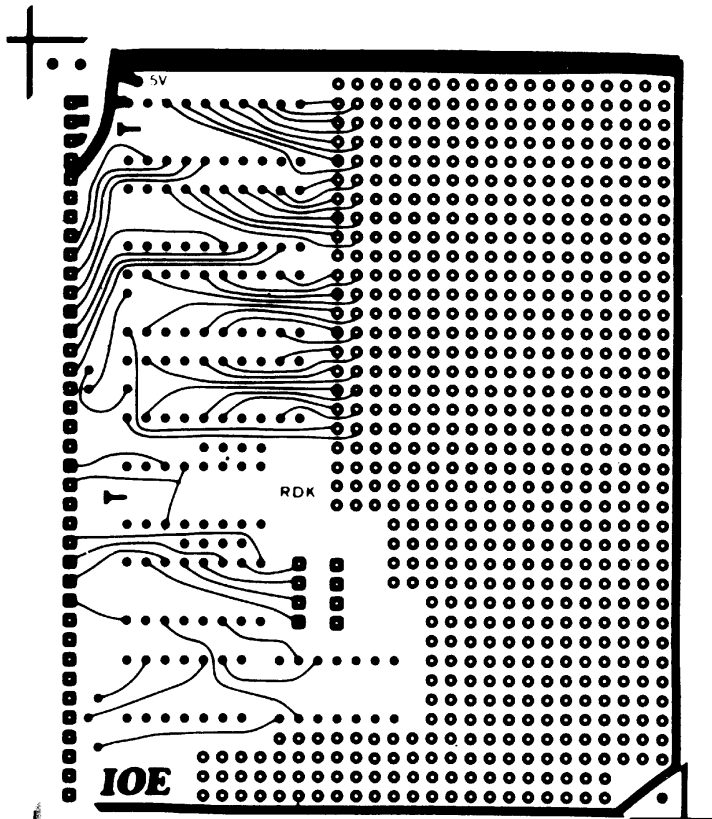


Abb. 5.5.4 Bestückungsseite der IO-Karte

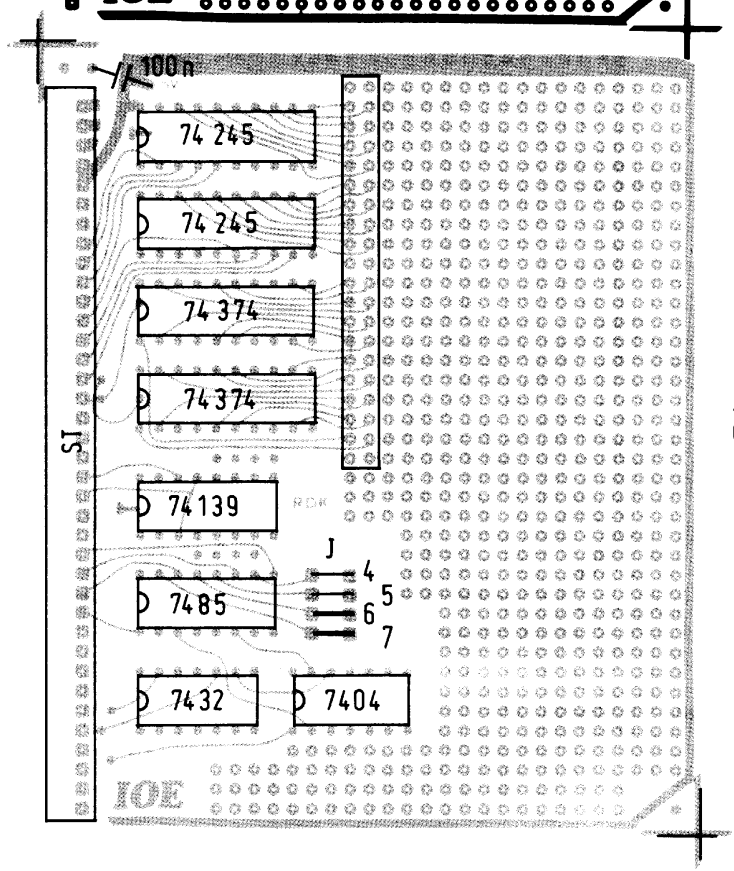


Abb. 5.5.5 Bestückungsplan der IO-Karte

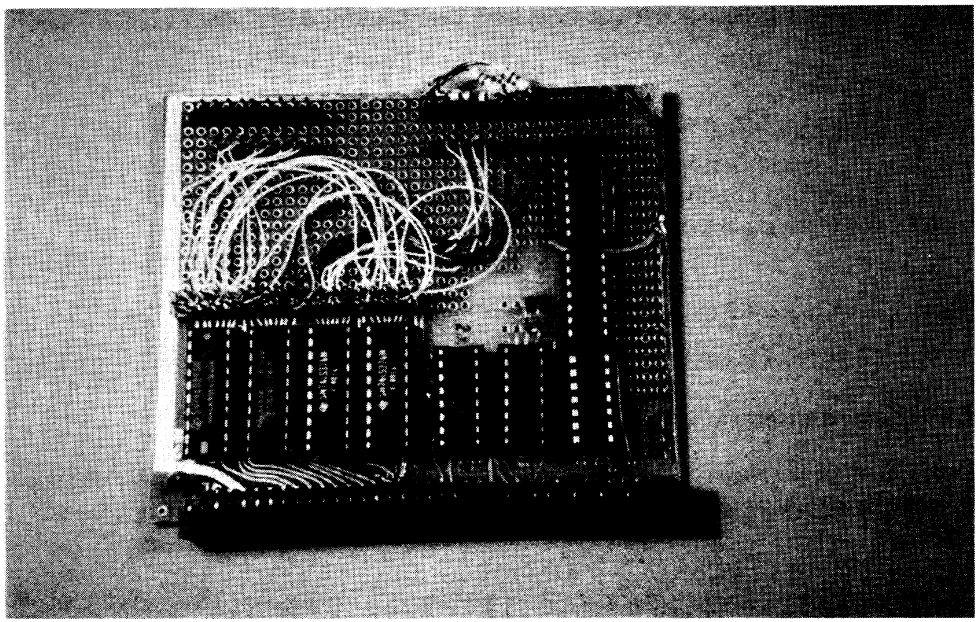


Abb. 5.10 IO-Karte

allerdings die Dekodierung etwas anders durchgeführt werden. Entweder wird das Auswahlsignal an Pin 19 durch eine eigene Schaltung erzeugt oder der Eingang von 01 Pin 13 wird anstelle an -RD an OV geschaltet. Dann spricht der Dekoder sowohl bei einer Eingabe, als auch bei einer Ausgabe an. Der Dekoder D2 darf dann bei den unteren beiden Adressen nicht verwendet werden, da er auch bei einer Ausgabe anspricht, es werden also entweder die beiden Latches LT1 und LT2 weggelassen oder an einen anderen Ausgang des Dekoders D2 (Pin 10, 9) gelegt. Beim Betrieb der beiden Latches LT1 und LT2 ist darauf zu achten, daß sie OV an den Pins 1 brauchen, damit die Ausgänge vom Tri-State in den Normalzustand gelangen. Dies ist insbesondere beim nächsten Test wichtig.

```
                        .z80

                        ;************************************
                        ;* IO Karte Vortest Dekodierung     *
                        ;************************************

0000'                   start:                  ;adresse z.B. 30h..
0000'   D3 30                   out (30h),a     ;LT1
0002'   D3 31                   out (31h),a     ;LT2
0004'   DB 30                   in a,(30h)      ;B1
0006'   DB 31                   in a,(31h)      ;B2
0008'   18 F6                   jr start
```

Abb. 5.5.6 Testprogramm IO-Karte

4. Test der Ausgabe. Sie läßt sich direkt mit dem Monitor testen. Dazu kann mit dem Befehl

Q030 55cr

z. B. das Bitmuster 55 auf den Latch LT1 gegeben werden. Mit dem Scop können dann die Ausgänge geprüft werden.

5. Die Eingabefunktion läßt sich genauso testen. Wird eine der Eingänge von 74LS245 z. B. B1 auf 0 gelegt, so kann dies durch den Befehl

QI30cr

geprüft werden. Bei B2 muß QI31 eingegeben werden.

5.6 Kassetteninterface

Bisher mußten alle eingegebenen Programme nach dem Stromausschalten wieder neu eingegeben werden. Hier wird eine Karte vorgestellt, die das Abspeichern von Programmen

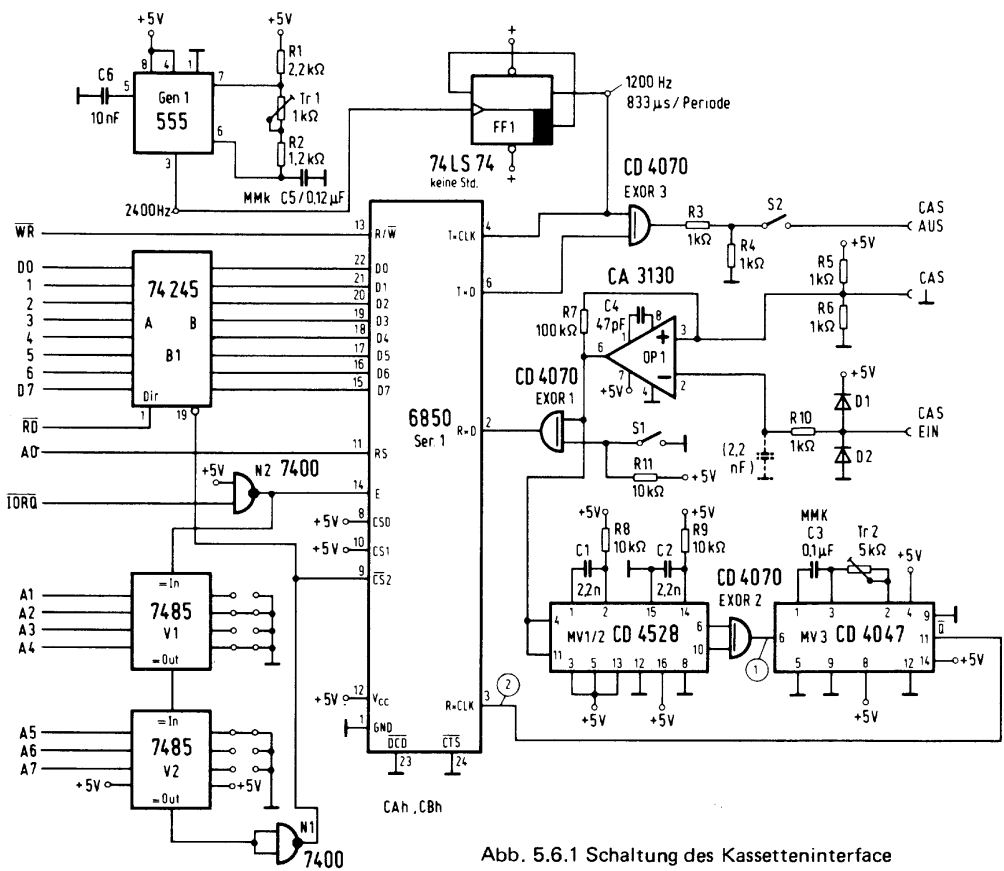


Abb. 5.6.1 Schaltung des Kassetteninterface

auf einem gewöhnlichen Tonbandkassettenrecorder erlaubt. Dabei wird mit einer Geschwindigkeit von 1200 Baud gearbeitet.

Abb. 5.6.1 zeigt die Schaltung. Als serieller Umsetzer wird hier der Baustein 6850 verwendet. Er erlaubt den Betrieb mit einer *1 Taktrate und ist daher für das Interface geeignet, wie wir noch sehen werden. Taktrate *1 bedeutet, daß beim Senden oder beim Empfang der Takt identisch mit der Baudrate ist. Beim Empfang muß der Takt synchron zu den einlaufenden Daten sein. Beim 6850 wird mit der steigenden Flanke des Taktes das Datensignal abgetastet. Die Baudrate wird hier mit einem IC Gen 1 (TI 555) erzeugt. Dabei ist sie nicht quarzstabilisiert, um den teueren Quarz einzusparen. Der Ausgang des Oszillators wird an einen Teiler FF1 geführt, um einen symmetrischen Takt zu erhalten. Dies ist unbedingt erforderlich, um die Modulation sauber durchführen zu können. Die Karte wird mit Hilfe der Vergleicher V1 und V2 adressiert und mit Hilfe von Brücken auf die Adresse CAh, CBh eingestellt. Der Buffer B1 dient nur der Pufferung des Datenbus. Der Baustein 6850 arbeitet ähnlich wie der Baustein 6551, nur daß er keinen internen Baudrate-Generator besitzt.

Nun zum Modulationsverfahren. Es wird ein Verfahren mit dem Namen PE oder Phase-encoding verwendet. Das heißt, die Phase einer Ausgangsfrequenz wird in Abhängigkeit der zu übertragenden Daten moduliert. Andere Verfahren sind z. B. das Frequenzmodulationsverfahren, bei dem sich die Frequenz in Abhängigkeit der Daten ändert. Das PE-Verfahren erlaubt aber höhere Übertragungsraten und ist einfacher zu verarbeiten. Dennoch findet man auf dem Markt häufig schlecht ausgeführte Modulationsverfahren mit FSK (Frequenzmod.), die den Kassetteninterface-Schaltungen allgemein einen schlechten Ruf gebracht haben. Meist arbeiten sie mit Baudraten bis 300 Baud und sind zudem fehleranfällig.

Unser Prinzip ist sehr einfach. *Abb. 5.6.2* zeigt den Vorgang bei der Modulation und Demodulation des Signals. Oben ist der Takt, bei uns sind es 1200 Hz, abgebildet. Er wird direkt zum seriellen Baustein 6850 geführt und ist gleichzeitig die Baudrate. Darunter ist das Ausgangssignal einer Datenübertragung, das direkt am Datenausgang des ICs 6850 anliegt, abgebildet. Es entsteht beim Aussenden der Zeichen 0FFh, 59h. Diese beiden Signale, Takt und Daten, werden nun Exclusiv-Oder verknüpft. Dies geschieht mit dem Gatter EX3. Am Ausgang entsteht dann das fertig modulierte Signal. In ihm kommen nur zwei Frequenzen vor, einmal 1200 Hz und dann noch 600 Hz. Damit ist die beanspruchte Bandbreite sehr klein.

Wie kann aber aus diesem Gemisch wieder das ursprüngliche Signal gewonnen werden? Das ist einfacher als man denkt. Zunächst wird auf jede Flanke des hereinkommenden Signals ein Monoflop getriggert. Es entsteht dann bei jeder Flanke ein kurzer Puls (Signal 1). Dieses Signal wird nun an ein weiteres Monoflop geführt. Es besitzt die Zeitkonstante 3/4 Tges. Wobei Tges=833.3 ys ist, bei 1200 BAUD. Am Ausgang des Monoflops entsteht ein Taktsignal. Dieses Signal ist nach dem ersten Zeichen gültig. Davor allerdings nicht, da es nach dem Einsetzen des Signals erst einmal einrasten muß. Bei der steigenden Flanke können die Daten (also direkt das modulierte Signal) abgetastet werden. Daraus ergibt sich wieder das ursprüngliche Signal.

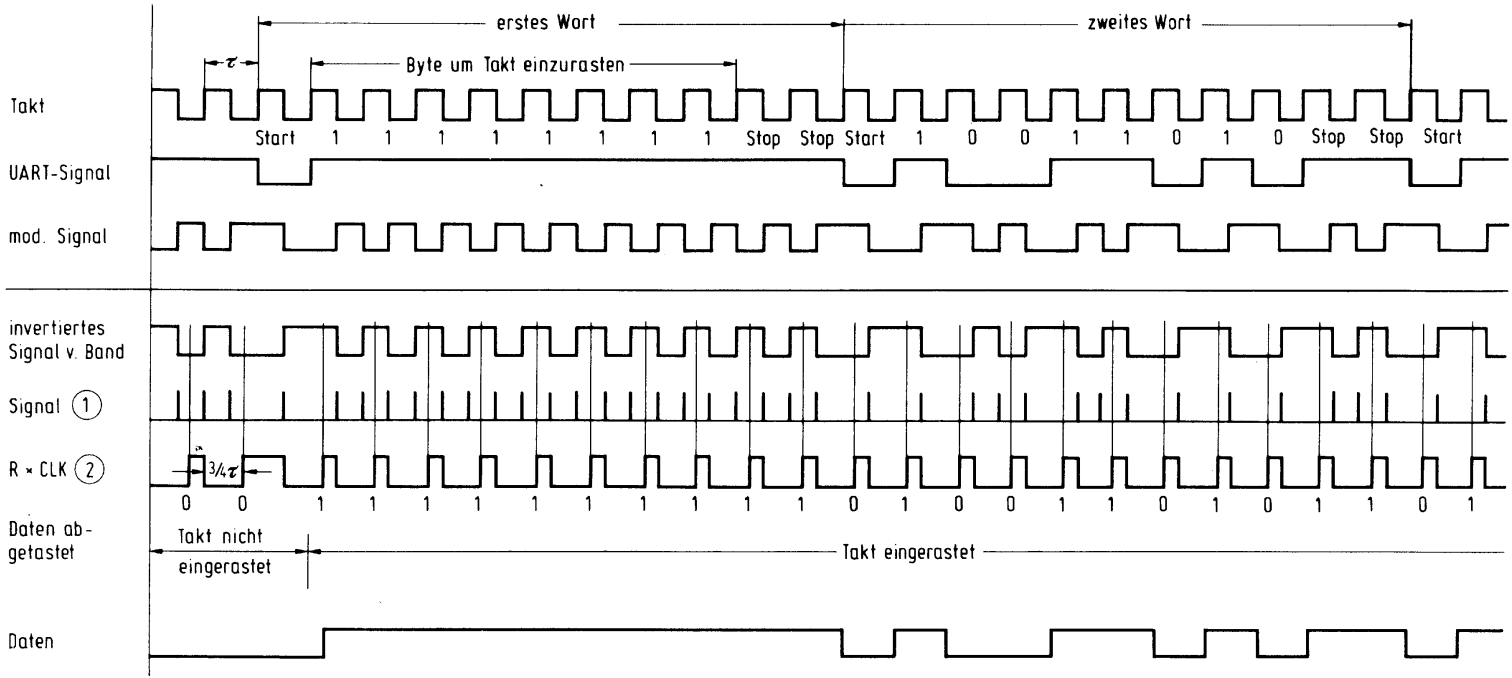


Abb. 5.6.2 Impulsdiagramm des Aufzeichnungsverfahrens

Um auf beide Flanken reagieren zu können, sind hier die beiden Monoflops MV1 und MV2 verwendet. Dabei reagiert das eine Monoflop auf die positive Flanke und das andere auf die negative Flanke des Datensignals. Die beiden Ausgangssignale werden über ein Exclusiv-Oder-Gatter EX2 gemischt und an das Monoflop MV3 geführt. An diesem Monoflop liegt der einzige Punkt des Demodulators, bei dem Abgleich nötig ist. Die Zeitkonstante des Monoflops muß mit TR2 auf 3/4 Tges abgeglichen werden.

Das modulierte Signal gelangt vom Ausgang des Kassettenrekorders direkt an den Eingang des OP1. Er ist als Komperator geschaltet und verstärkt das Signal auf TTL-Pegel. Die Masse des Kassettenrecorders ist nicht identisch mit dem OV-Pegel des Computers, sondern liegt auf 2.5 V. Damit wird erreicht, daß bei der Aufnahme ein gleichspannungsfreies Signal an den Recorder gelangt und umgekehrt. Der Ausgang des OPs gelangt einmal an das Exklusiv-Oder Gatter EX1 und an die beiden Monoflops MV1 und MV2. Das Exclusiv-Oder-Gatter EX1 dient dazu, eine eventuell durch den Recorder bedingte Polarisationsumkehr des Signals wieder rückgängig zu machen. Mit dem Schalter S1 kann die Polarität eingestellt werden. Der Schalter S2 dient dazu, bei der Wiedergabe den Datenausgang vom Rekorder zu trennen und damit Störungen zu vermeiden.

Abb. 5.6.3 zeigt die Bedeutung des Befehls-Registers des seriellen Bausteins. In *Abb. 5.6.4* ist die Aufteilung der einzelnen Bits des Status-Registers dargestellt.

Nun zum Aufbau. *Abb. 5.6.5* zeigt die Lötseite der Platine, *Abb. 5.6.6* zeigt die Bestückungsseite und in *Abb. 5.6.7* ist der Bestückungsplan abgebildet.

Beim Aufbau wird mit der Bestückung der passiven Bauteile begonnen. Dann werden die IC-Sockel eingelötet. Wichtig ist bei den beiden Kapazitäten C3 und C5 die Verwen-

```
Control Register (hier Port 0CAh schreibend)

7       6       5       4       3       2       1       0

|       |       |       |       |       |       0       0  /1
|       |       |       |       |       |       0       1  /16
|       |       |       |       |       |       1       0  /64
|       |       |       |       |       |       1       1  Master Reset
|       |       |       |       |       |
|       |       |       0       0       0  7 Bits + Even Par. + 2 Stop Bits
|       |       |       0       0       1  7 Bits + Odd Par.  + 2 Stop Bits
|       |       |       0       1       0  7 Bits + Even Par. + 1 Stop Bit
|       |       |       0       1       1  7 Bits + Odd Par.  + 1 Stop Bit
|       |       |       1       0       0  8 Bits + 2 Stop Bits
|       |       |       1       0       1  8 Bits + 1 Stop Bit
|       |       |       1       1       0  8 Bits + Even Par. + 1 Stop Bit
|       |       |       1       1       1  8 Bits + Odd Par.  + 1 Stop Bit
|       |       |
|       0       0  -RTS auf Low Transm. Interrupt disbled
|       0       1  -RTS auf Low Transm. Interrupt enabled
|       1       0  -RTS auf High Transm. Interrupt disabled
|       1       1  -RTS auf Low Ausgabe eines BREAK-Zeichens
|                       (dauerhaft 0 am Datenausgang) und
|                       Transmitter Interrupt disabled
|
1 = Receiver Interrupt enabled
```

Abb. 5.6.3 Control-Register des 6850

```
Status Register (hier Port 0CAh lesend)

7       6       5       4       3       2       1       0

|       |       |       |       |       |       |       1=Receive Daten
|       |       |       |       |       |       |         sind da
|       |       |       |       |       |       1=Transmitter ist leer
|       |       |       |       |       -DCD
|       |       |       |       -CTS
|       |       |       FE (Framing Fehler)
|       |       OVRN (Receiver Ueberlauf)
|       PE (Paritaets Fehler)
IRQ (Interrupt Request)
```

Abb. 5.6.4 Status-Register des 6850

dung von hochwertigen Bauteilen (z. B. MMK Folienkondensatoren 0.1 yF und 0.12 yF), da davon die Stabilität der Schaltung abhängt.

1. Messen der Versorgungsspannungen an den ICs.
2. Einsetzen der Bausteine 74LS245, 2*74LS85, 74LS00. Einlöten der Brücken für die Adresse 0CAh, 0CBh, wie im Schaltplan und Bestückungsplan gezeigt.

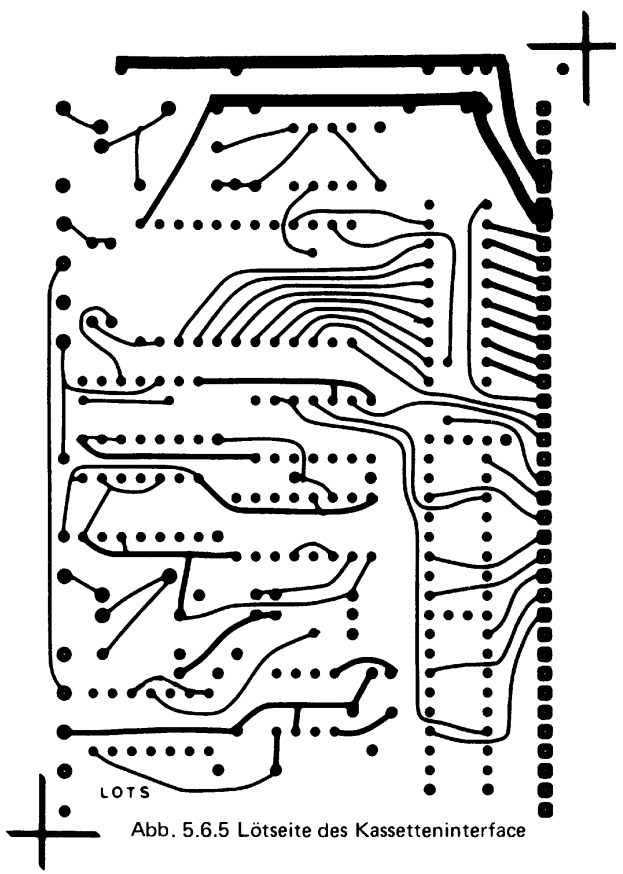


Abb. 5.6.5 Lötseite des Kassetteninterface

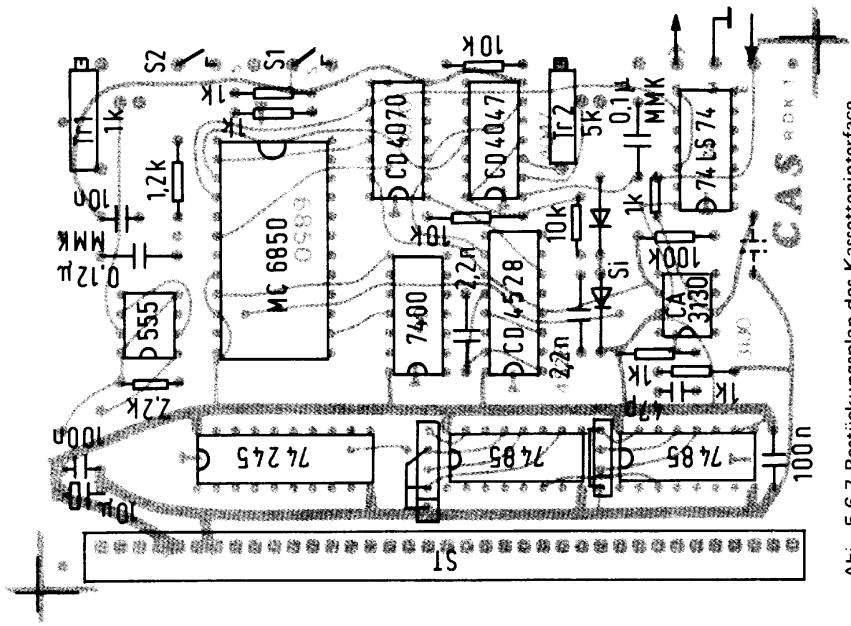


Abb. 5.6.7 Bestückungsplan des Kassetteninterface

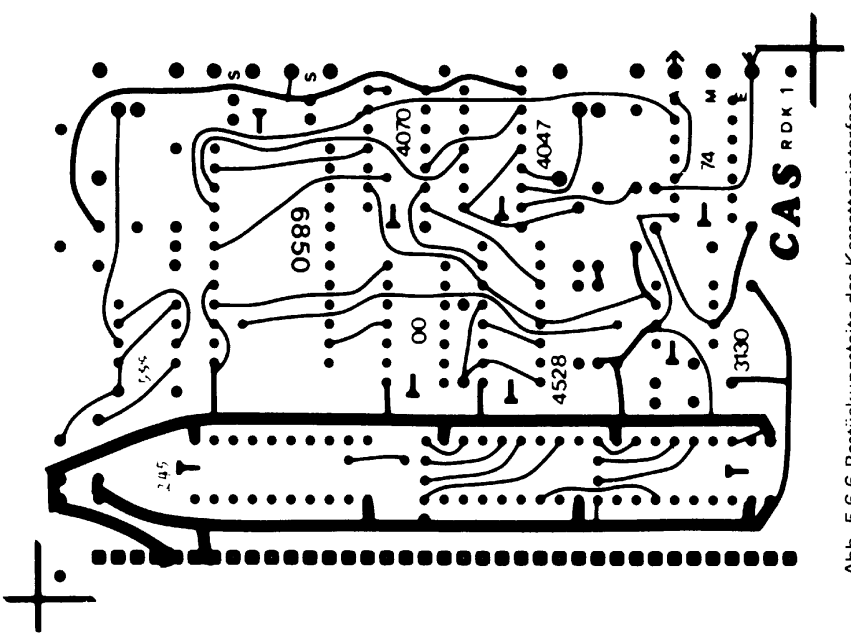


Abb. 5.6.6 Bestückungsseite des Kassetteninterface

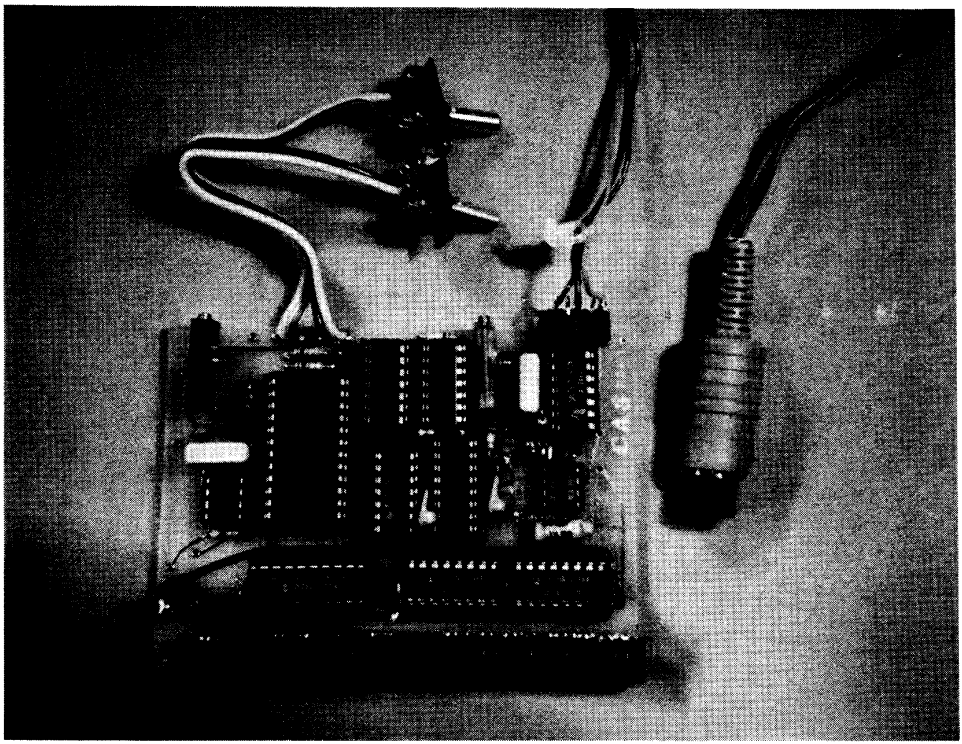


Abb. 5.11 Kassetteninterface

```
                        ;************************************
                        ;* Test Kassetteninterface          *
                        ;* Dekoder Test                     *
                        ;************************************

0000'   DB CA           start:  in a,(0cah)      ;vier pulse
0002'   DB CB                  in a,(0cbh)      ;werden bei
0004'   D3 CA                  out (0cah),a     ;-cs erzeugt
0006'   D3 CB                  out (0cbh),a     ;
0008'   18 F6                  jr start
```

Abb. 5.6.8 Testprogramm Kassetteninterface

3. Eingabe des Testprogramms aus *Abb. 5.6.8* z. B. auf Adresse 1200h. Damit wird die Dekodierung getestet. Es muß ein Oszillogramm nach *Abb. 5.6.9* entstehen.

4. Einsetzen der restlichen Bausteine. Als FF1 muß unbedingt ein 74LS74 eingesetzt werden und es darf kein Standard-Baustein verwendet werden, da sich sonst Störungen auf der Taktleitung wegen der Belastung des Generators Gen1 ergeben.

5. Messen an Pin 4 des Bausteins 6850. Dies ist die Baudrate. Sie muß auf eine Frequenz von 1200 Hz mit Hilfe des Trimmers Tr1 eingestellt werden. Der genaue Abgleich läßt

sich am besten mit einem Frequenzmeßgerät durchführen, aber auch das Scop genügt im Prinzip. Die Periodendauer des Taktes ist 833.3 ys. Auf der Leitung dürfen keine Störungen sichtbar sein.

6. Nun kann das Testprogramm aus *Abb. 5.6.10* eingegeben werden. Es hat die Aufgabe, ein Bitmuster auszugeben. In *Abb. 5.6.11* sind ein paar Ausgangswerte oszillographiert.

7. Mit Hilfe einer kleinen Zusatzschaltung, die *Abb. 5.6.12* zeigt, kann der Aufnahmeteil abgeglichen werden. Die Zusatzschaltung simuliert den Recorder. Dadurch steht ein kontinuierliches Signal zur Verfügung. Der Schalter S2 wird zuvor eingeschaltet.

8. Zum Abgleich muß mit Trimmer Tr2 die Zeit 3/4 Tges eingestellt werden. Es ergibt sich dann ein Oszillogramm nach *Abb. 5.6.13*. Mit jeder Flanke des Datensignals wird das Monoflop getriggert. An Pin 3 liegt das regenerierte Taktsignal. Dabei ist das Signal 3 Einheiten auf 0-Pegel und eine Zeiteinheit auf einem 1-Pegel. 4 Zeiteinheiten sind 833.3 ys.

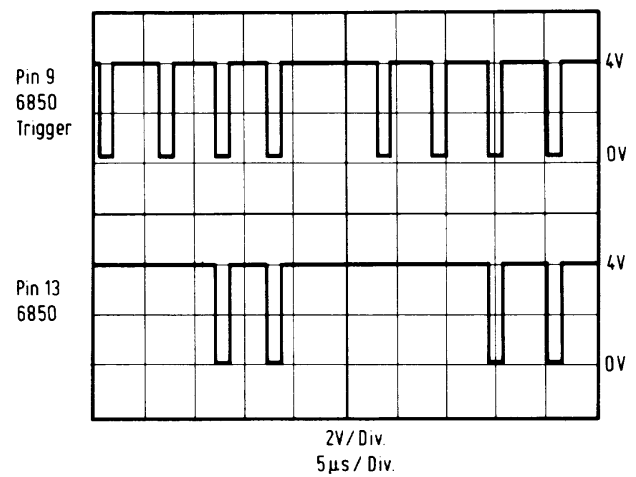


Abb. 5.6.9 Oszillogramm Kassetteninterface

```
                        ;***********************************
                        ;* Test und Abgleich Sender        *
                        ;* bei Kassetteninterface          *
                        ;***********************************

0000'   3E 53           startm: ld a,53h        ;baudrate * 1
0002'   D3 CA                   out (0cah),a    ;command port
0004'   3E 50                   ld a,50h        ;2stop 8daten
0006'   D3 CA                   out (0cah),a
0008'   0E 56                   ld c,01010110b  ;test datenmuster 'V'
000A'   DB CA           loop:   in a,(0cah)     ;status testen
000C'   E6 02                   and 00000010b   ;transmitter
000E'   28 FA                   jr z,loop       ;warten bis leer
0010'   79                      ld a,c          ;zeichen
0011'   D3 CB                   out (0cbh),a    ;datenport
0013'   18 F5                   jr loop         ;und nochmals
```

Abb. 5.6.10 Testprogramm Kassetteninterface

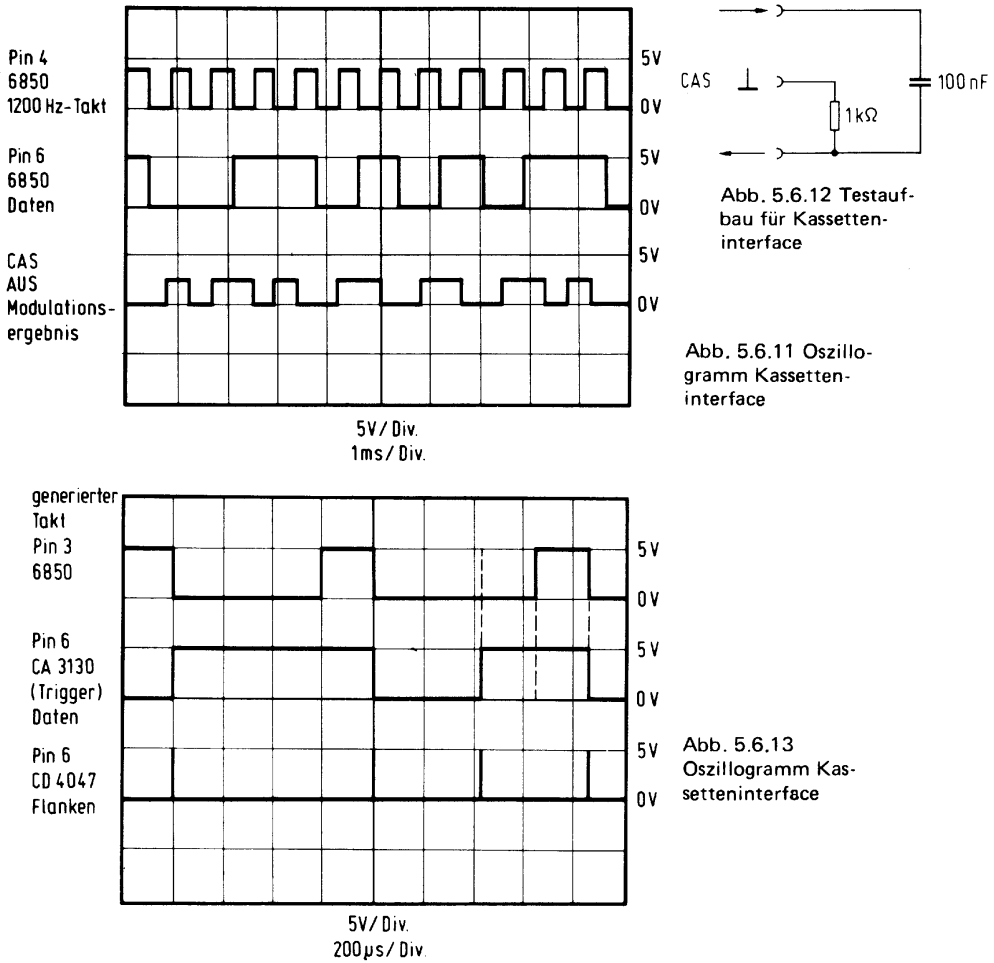


Abb. 5.6.12 Testaufbau für Kassetteninterface

Abb. 5.6.11 Oszillogramm Kassetteninterface

Abb. 5.6.13 Oszillogramm Kassetteninterface

9. Nun kann der Kassettenrecorder angeschlossen werden. Eingabe des Testprogramms nach *Abb. 5.6.14*. Damit kann der Dateninhalt einer Aufzeichnung auf der Console ausgegeben werden. Es wird zunächst ein Datenbereich, z. B. unser Testprogramm auf Kassette aufgezeichnet. Dies geschieht mit dem Befehl:

```
W1200  12FFcr
```

Dazu muß, falls die Karte KEY vorhanden ist, der Dil-Schalter richtig eingestellt sein.

7	6	5	4	3	2	1	0
x	x	x	1	x	x	x	x

Bit 4 liegt auf 1. Damit wird die Ausgabe des Befehls W auf die Kassette gelenkt.

10. Nun wird das Testprogramm gestartet und der Recorder zurückgespult. Es muß nun eine lesbare Ausgabe im Intel-Hexformat (siehe Monitorbeschreibung) auf dem Bild-

```
                                ;************************************
                                ;* Test und Abgleich Empfaenger     *
                                ;* bei Kassetteninterface           *
                                ;************************************

                                        org 1200h       ;absolutes prg
                                                        ;start in ram
1200'   31 122D'                starte: ld sp,stack     ;init stackpointer
1203'   3E 53                           ld a,53h        ;baudrate * 1
1205'   03 CA                           out (0cah),a    ;command port
1207'   3E 50                           ld a,50h        ;2stop 8daten
1209'   03 CA                           out (0cah),a
120B'   DB CA                   loop1:  in a,(0cah)     ;status testen
120D'   E6 01                           and 00000001b   ;receiver ready
120F'   28 FA                           jr z,loop1      ;warten bis zeichen da
1211'   DB CB                           in a,(0cbh)     ;holen
1213'   4F                             ld c,a          ;nun Monitorunterprg
1214'   CD 0009                         call 0009       ;ROUTINE CO
1217'   18 F2                          jr loop1
1219'                                   defs 20         ;reserve fuer stack
122D'                           stack:  defs 1          ;dort starten lassen
```

Abb. 5.6.14 Testprogramm Kassetteninterface

schirm erfolgen. Ist dies nicht der Fall, so wird der Schalter S1 umgestellt, denn ggf. liefert der Kassettenrecorder ein inverses Datensignal. Dann aber muß die Schaltung arbeiten.

11. Nun erfolgt der letzte Test. Der Rechner wird zur Probe einmal kurz ausgeschaltet. Nach dem Reset wird der Befehl:

Rcr

eingegeben und der Kassettenrecorder eingeschaltet. Nach einer kurzen Zeit, kurz bevor das Signal verstummt und der Einlesevorgang beendet ist, muß sich der Monitor wieder melden. Mit dem S-Befehl oder D-Befehl kann auf Adresse 1200h nachgesehen werden, ob die Daten angekommen sind. Bei einem Lesefehler wird dies vom Monitor mit CHECKSUM gekennzeichnet, da jede Aufzeichnung mit einer Prüfsumme versehen wird.

5.7 Sound-Generator

Hier wird eine kleine Karte beschrieben, die sowohl Töne als auch Geräusche produzieren kann. Dazu ist ein IC verwendet, das drei programmierbare Tongeneratoren sowie einen Rauschgenerator beinhaltet. Jeder Tongenerator kann in der Tonhöhe sowie Lautstärke programmiert werden. Ferner ist die Tonlage des Rauschgenerators programmierbar. Alle Ausgangssignale der Generatoren können gemischt werden und anstelle der Lautstärkeprogrammierung kann auch ein programmierbarer Hüllkurvengenerator verwendet werden.

Abb. 5.7.1 zeigt die Schaltung der Karte. Die Dekodierung erfolgt wie üblich mit einem Vergleicher und der Bustreiber B1 dient zur Trennung des Datenbusses. Der Baustein be-

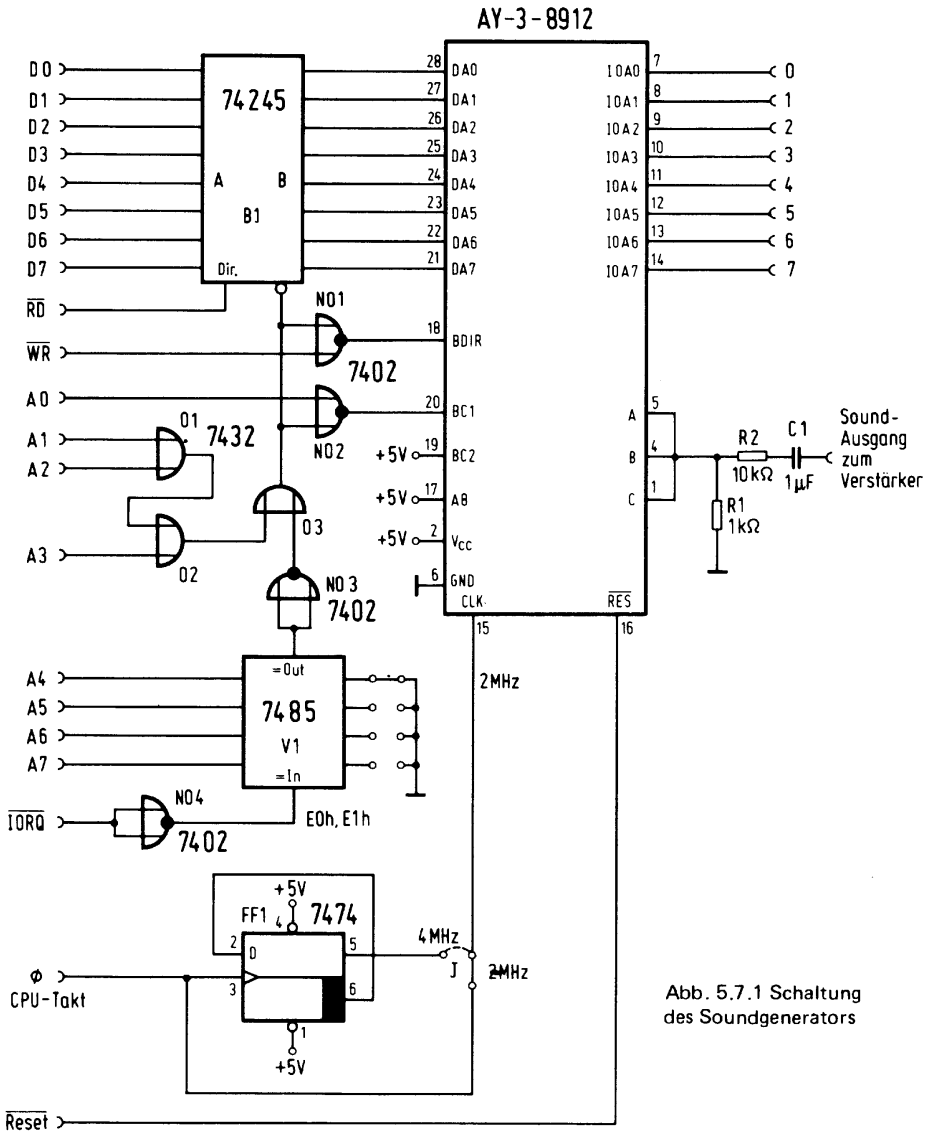


Abb. 5.7.1 Schaltung des Soundgenerators

nötigt ein paar ungewöhnliche Signale mit der Bezeichnung BDIR, BC1, die aber im Prinzip ähnlich wirken wie R/-W und -CS. Die Signale werden mit den Gattern N01 und N02 erzeugt. In BC1 ist außerdem noch die Adreßinformation enthalten. Das IC wird mit zwei Adressen angesprochen. Die untere Adresse, bei uns also E0h, führt an ein internes Adreßregister. Die darin enthaltene Adresse bewirkt die Auswahl eines von 16 internen Registern. Die Daten werden über E1h an das IC ausgegeben. Daten können auch aus den internen Registern gelesen werden, dies geschieht aber über die Adresse 0E0h.

Reg.		7	6	5	4	3	2	1	0
0	Kanal A Ton LSB	7	6	5	4	3	2	1	0
1	Kanal A MSB	x	x	x	x	B	A	9	8
2	Kanal B Ton MSB	7	6	5	4	3	2	1	0
3	Kanal B MSB	x	x	x	x	B	A	9	8
4	Kanal C Ton LSB	7	6	5	4	3	2	1	0
5	Kanal C MSB	x	x	x	x	B	A	9	8
6	Rauschperiode	x	x	x	x	3	2	1	0
7	Freigabe 0=an	In/Out (IOB)	In/Out (IOA)	Rausch C	Rausch B	Rausch A	Ton C	Ton B	Ton A
8	Kanal A Amplitude	x	x	x	M	L3	L2	L1	L0
9	Kanal B Amplitude	x	x	x	M	L3	L2	L1	L0
A	Kanal C Amplitude	x	x	x	M	L3	L2	L1	L0
B	Hüllkurvenperiode LSB	7	6	5	4	3	2	1	0
C	Hüllkurvenperiode MSB	F	E	D	C	B	A	9	8
D	Hüllkurvenform	x	x	x	x	Cont	ATT	ALT	Hold
E	I/O (Port A)	7	6	5	4	3	2	1	0
F	I/O (Port B) nur bei 8910	7	6	5	4	3	2	1	0

M=1, dann Hüllkurve

Abb. 5.7.2 Register des Soundgenerators

Die Ausgänge A, B und C des Sound-Generators werden direkt zusammengeschaltet und am Sound-Ausgang hinter C1 liegt die NF-Spannung an. Sie kann z. B. an den Tonbandeingang eines Radios geführt werden oder an einen getrennt aufgebauten NF-Verstärker. Der Sound-Generator wird mit einer Frequenz von 2 MHz betrieben, dabei ergibt sich der beste Frequenz-Bereich der Tongeneratoren, z. B. für die Erzeugung von Melodien. Mehr als 2 MHz verträgt der Baustein nicht. Daher ist für ein 4 MHz-Z80-System auf der Karte noch ein Teiler mit FF1 aufgebaut. Über die Brücke J kann dann die Frequenz ausgewählt werden. Nun zu den internen Registern.

Abb. 5.7.2 zeigt die Aufteilung. Mit den Registern R0 bis R5 wird die Tonhöhe der einzelnen Generatoren eingestellt. Jeweils zwei Register bestimmen einen Ton, da mit 12 Bit gearbeitet wird. Dabei wird der Eingangstakt zunächst einmal immer durch 16 dividiert. Dann wird durch herunterzählen eines 12 Bit-Zählers, der mit dem angegebenen Wert geladen wird, die Ausgangsfrequenz erzeugt.

Eine Rauschquelle ist mit Register R6 programmierbar. Die Grundfrequenz des Rauschgenerators wird durch herunterteilen der Taktfrequenz um 16 erreicht. Ein 5 Bit-Wert lädt dann wieder einen Zähler der heruntergezählt wird und damit die Frequenz bestimmt.

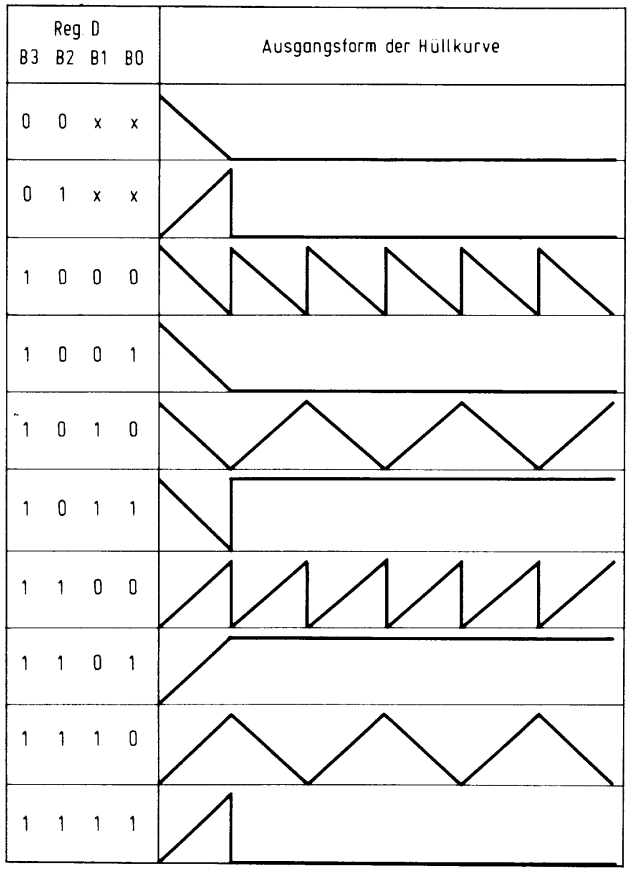


Abb. 5.7.3 verschiedene Hüllkurven

Mit Register 7 können die einzelnen Quellen freigegeben werden. Eine 0 im entsprechenden Bit gibt sie frei. Im Soundgenerator ist aber auch noch ein Parallelport vorhanden. Die Richtung kann ebenfalls programmiert werden. 0 programmiert den Port als Eingang. Dabei gibt es im AY-3-8912, der Baustein, den wir verwenden, nur einen Port, wohingegen der sonst kompatible Baustein AY-3-8910 zwei Ports enthält.

Die Amplitude der drei Kanäle (Ton oder Rauschen) kann mit den Registern R8 bis RA bestimmt werden. Dabei gibt ein Wert von 0 bis 15 die Lautstärke an. Die Lautstärke wird logarithmisch eingestellt. Zum Ausschalten eines Kanals wird der Wert auf 0 gesetzt. Ist das Bit 4 gesetzt, so wird die Lautstärkeeinstellung von einem Hüllkurvengenerator gesteuert. Die Hüllkurvenperiode läßt sich mit den Registern RB und RC einstellen, dabei wird mit 16 Bits gearbeitet, um auch Perioden mit sehr großer Zeitdauer erhalten zu können.

Mit Register RD kann die Hüllkurvenform eingestellt werden. *Abb. 5.7.3* zeigt die verschiedenen Hüllkurvenformen, die programmierbar sind. Die Hüllkurve wird beim Einschreiben in Register RD gestartet.

Register RE ist die direkte Verbindung zum IO-Port. RF ist beim Ay 9812 nicht verwendet.

Abb. 5.7.4 zeigt die Lötseite der Platine des Sound-Generators. In *Abb. 5.7.5* ist die Bestückungsseite gezeigt und *Abb. 5.7.6* zeigt den Bestückungsplan.

Da die Schaltung nicht sehr umfangreich ist, ist der Aufbau recht einfach. Es werden passive Bauteile und Sockel eingelötet. Die Brücke wird so eingelötet, daß sich Adresse 0E0h (0E1h) ergibt.

1. Messen der Versorgungsspannungen, bevor die ICs eingesetzt werden.
2. Einsetzen aller ICs. Eingabe des Programms nach *Abb. 5.7.7*. Nach dem Start wird erst einmal die Dekodierung geprüft. Am Ausgang des Vergleichers V1 Pin 6 müssen Pulsgruppen erscheinen. An Pin 15 des Soundgenerators muß ein 2 MHz Takt anliegen und Pin 16 des 8912 muß auf einem High-Pegel liegen.
3. Nun wird am Ausgang des Soundgenerators hinter dem Kondensator C1 gemessen.

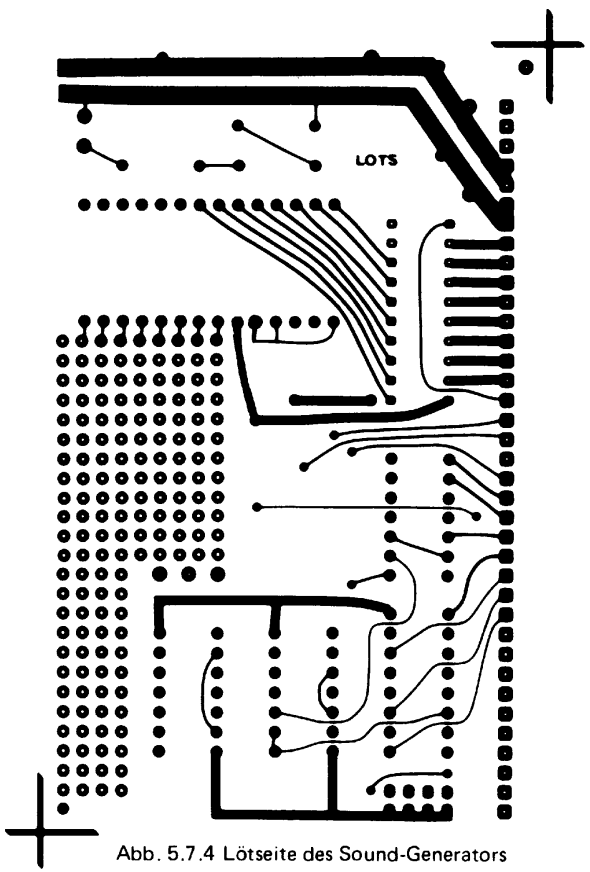


Abb. 5.7.4 Lötseite des Sound-Generators

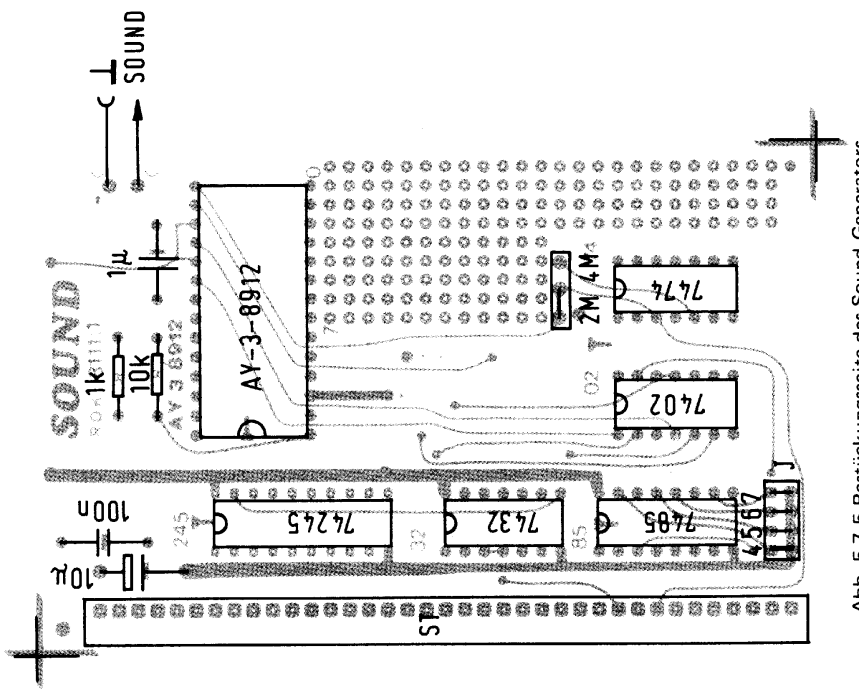


Abb. 5.7.5 Bestückungsseite des Sound-Generators

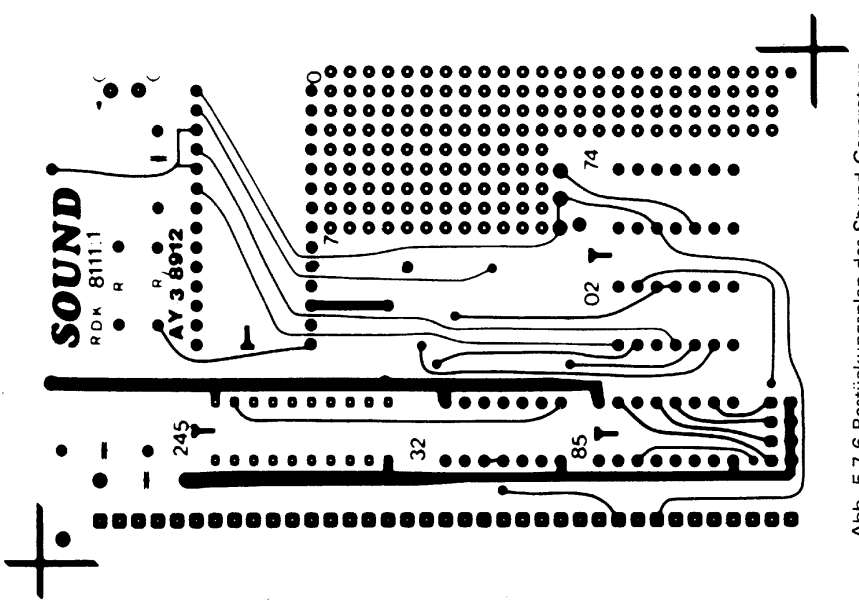


Abb. 5.7.6 Bestückungsplan des Sound-Generators

Dort muß eine Frequenz mit einer Periode von etwa 700 ys anstehen. Dann ist die Schaltung in Ordnung.

Beispiele für den Soundgenerator finden sich im Softwarekapitel.

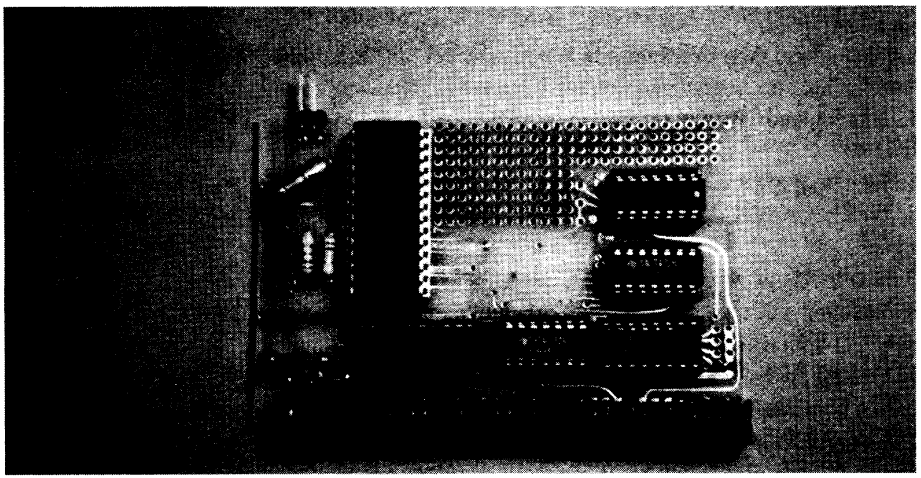


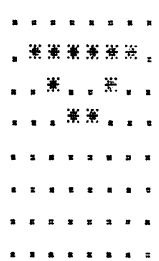
Abb. 5.12 Sound-Generator

```
                        ;*********************************
                        ;* Test des Soundgenerators      *
                        ;*********************************

                                org 1200h

1200'   3E 00           start:  ld a,0          ;adresse 0
1202'   D3 E0                   out (0e0h),a    ;adresslatch
1204'   3E 55                   ld a,55h
1206'   D3 E1                   out (0e1h),a
1208'   3E 01                   ld a,1
120A'   D3 E0                   out (0e0h),a    ;ton kanal 0
120C'   3E 00                   ld a,0          ;msb=0
120E'   D3 E1                   out (0e1h),a
1210'   3E 07                   ld a,7
1212'   D3 E0                   out (0e0h),a
1214'   3E FE                   ld a,11111110b  ;kanal A ton frei
1216'   D3 E1                   out (0e1h),a
1218'   3E 08                   ld a,8
121A'   D3 E0                   out (0e0h),a    ;lautstaerke
121C'   3E 0F                   ld a,15         ;auf maximum
121E'   D3 E1                   out (0e1h),a
1220'   18 0E                   jr start        ;und wiederholen
                                                ;fuer messungen
                                                ;fuer ton nur
                                                ;einmal noetig
                                                ;die routine zu
                                                ;durchlaufen
```

Abb. 5.7.7 Testprogramm für den Sound-Generator

5.8 Fragen zum Kapitel 5

1. Nennen Sie ein paar Beispiele für Peripherie-Geräte.
2. Mit welcher Schnittstelle (parallel oder seriell) lassen sich Daten schneller übertragen und warum?
3. Wieviel Zeichen pro Sekunde lassen sich bei 300 Baud mit 7 Datenbits + Parität und einem Stopbit übertragen?
4. Das folgende Bitmuster ist, um ein Paritätsbit zu einer neuen Parität zu erweitern: 10010110
5. Welche Aufgabe haben die Treiber ICs T1 bis T4 in Abb. 5.1.4?
6. Welcher ASCII-Code gehört zu dem Zeichen ‚o', die Angabe soll in HEX (sedezimal) erfolgen?
7. Warum ist in Abb. 5.1.12 RTS mit CTS verbunden?
8. Welcher ASCII-Code wird durch die Betätigung der Tasten CTRL-C bei einer Standard-ASCII-Tastatur ausgegeben?
9. Welche Aufgabe hat der Zeichengenerator bei der CRT-Karte?
10. Wo liegt der Bildwiederholspeicher bei der CRT-Karte (ICs angeben)?
11. Es soll folgendes Zeichen im Zeichengenerator abgelegt werden:



Wie sieht die Bitkombination im EPROM aus?

12. Das Zeichen soll bei dem ASCII-Code 7Fh (127d) erscheinen, bei welcher Anfangsadresse muß das Zeichen im EPROM abgelegt werden?
13. Wozu dient der Transistor T3 in Abb. 5.4.1?
14. Warum liegt der Eingang von I1 an -IORQ und warum genügt es nicht, den Eingang einfach auf Masse zu legen?
15. Wozu dient der Schalter S1 in Abb. 5.6.1?
16. Was bewirkt der Hüllkurvengenerator beim Soundgenerator, siehe Abb. 5.7.3?

6 Software

Dieses Kapitel stellt eine Begleitung zu den Kapiteln 4 bis 5 dar und kann auch völlig getrennt davon gelesen werden. Nach einer Einführung in die Z80-Programmierung folgen Beispiele mit großen und kleinen Programmen. Die Beispiele können zusammen mit den in Kapitel 4 und 5 beschriebenen Karten ausprobiert werden. Die Abschnitte 6.5, 6.6 und 6.7 sind unabhängig voneinander.

6.1 Z80-Aufbau und Befehle

Bevor es an den Z80 geht, müssen ein paar Grundvoraussetzungen und Begriffe geklärt werden. Bei Mikrorechnern werden verschiedene Zahlensysteme verwendet. Einmal das dezimale System, dann aber auch das binäre, oktale oder hexadezimale (korrekterweise als sedezimal bezeichnet). Dazu ein paar Beispiele. Die Zahl 123 soll in allen drei Systemen dargestellt werden. Erst einmal die Umrechnung in binär:

Die nächst kleinere Zweierpotenz (2 hoch n) ist 64, sie läßt sich von 123 abziehen und es bleibt 59 als Rest. Davon kann man 32 abziehen und es bleibt 27. Davon läßt sich 16 abziehen und es bleibt 11, 8 geht ebenfalls, Rest 3, 4 geht nicht, aber 2 und schließlich 1. Damit haben wir die Zahl zerlegt:

64 + 32 + 16 + 8 + 0 + 2 + 1

Im Binärsystem lautet die Zahl also

1111011

Wir wollen die Zahl nun oktal darstellen. Dazu geht man am besten von der Binärzahl aus und teilt sie von rechts nach links in Dreiergruppen auf:

001 111 011

Nun wird die jeweilige Dezimahlzahl unter die Dreiergruppen geschrieben und es ergibt sich

173 als Oktalzahl.

Zur Umrechnung in HEX geht man ähnlich vor, die Binärzahl wird von rechts nach links in Vierergruppen aufgeteilt.

0111 1011

Und nun muß die entsprechende Dezimalzahl unter die Gruppe geschrieben werden:

7 11

11 ist aber eine dezimale Darstellung. Beim HEX-System benötigen wir weitere Zahlen, man hat sich auf die Buchstaben A bis F geeinigt und nimmt sie hinzu, so daß gilt:

1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

dann ergibt sich die Zahl

7B

als Hexzahl der dezimalen Zahl 123.

Um künftig die einzelnen Zahlendarstellungen auseinanderhalten zu können, wird hinter die Zahl eine Kennung geschrieben, dabei wird für das dezimale System keine Kennung verwendet, in Zweifelsfällen jedoch ein kleines d. Beim oktalen System, das wir jedoch nicht verwenden, wäre das Zeichen „o" die Kennung, beim binären System wird ein kleines „b" verwendet und beim hexadezimalen (sedezimalen) System wird ein kleines „h" gebraucht. Damit ergibt sich für das obere Beispiel

123 = 123d = 1111011b = 173o = 7Bh

Weiteres Beispiel

16333 = 16333d = 11111111001101b = 37715o = 3FCDh

Die Umrechnung ins dezimale System ist auch sehr einfach. Dazu folgendes Beispiel:

4BAh ist ins dezimale System umzurechnen.

Es gilt

4*16*16 + B*16 + A

ist die dezimale Darstellung.
Was aber mit B*16? Dazu wird die Zahl dezimal umgerechnet und B ist 11.
Die Zuordnung lautet:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Somit ist

4BAh = 4*16*16 + 11*16 + 10 = 1210d

die Lösung der Aufgabe.

Wir kommen zum Innenleben des Z80s. Er besitzt eine Vielzahl von Speichereinheiten, die Register genannt werden. In solch einem Register können zum Beispiel die Zwischenwerte von Rechnungen abgelegt werden. Manche der Register können auch Adressen aufnehmen, um den externen Speicher anzusprechen. Eines dieser Register ist der uns schon bekannte Programmzähler. *Abb. 6.1.1* zeigt das Innenleben des Z80s. Durch die vielen Register darf man sich nicht verwirren lassen. Zwei Register sind für uns zunächst bedeutend. Einmal der schon bekannte Programmzähler, der mit PC bezeichnet ist und dann ein Register mit der Bezeichnung Akku A. Der Akku ist bei Rechnern meist ein besonderes Register. In ihm können Rechnungen ausgeführt werden, aber auch logische Verknüp-

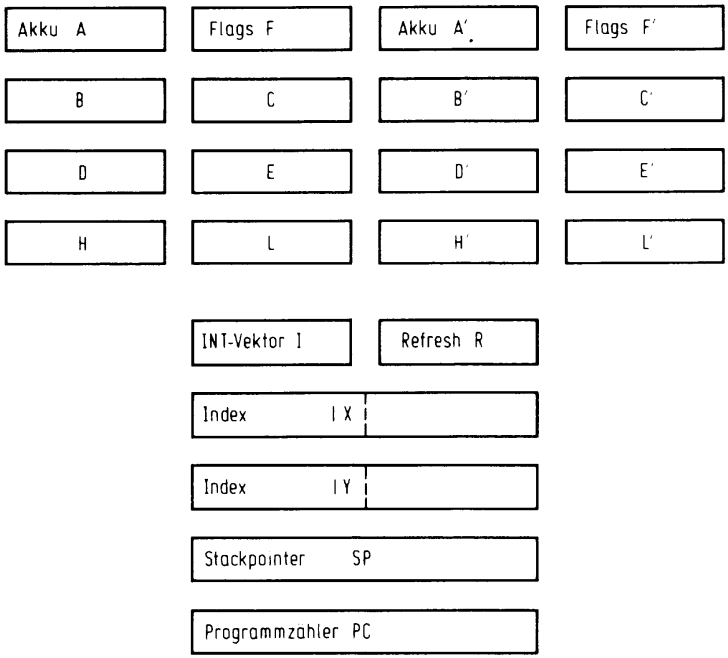


Abb. 6.1.1 Register des Z80

fungen durchgeführt werden. Im Prinzip würden wir mit diesen beiden Registern, dem Akku und dem Programmzähler, auskommen. Tatsächlich gibt es andere Mikrocomputer, die praktisch nur diese beiden Register besitzen. Beim Z80 wird aber mindestens noch ein Register zum sinnvollen Arbeiten benötigt, wir wählen das Register B hinzu. Es wird benötigt, um mit dem Akku zusammenzuarbeiten (wir hätten auch eines der anderen Register C, D, E, H oder L verwenden können, aber keines der restlichen).

Nun fangen wir an, ein paar Befehle zusammenzustellen. Eine wichtige Gruppe sind die Ladebefehle. Sie haben die Aufgabe, einen Wert aus dem Speicher z. B. in ein Register zu laden. Man hat sich darauf geeinigt, von einem Ladevorgang zu sprechen, wenn vom Speicher in den Mikrorechner transportiert wird und von wegspeichern, wenn vom Mikrorechner in den Speicher transportiert wird, oder allgemeiner, Laden beim Transport von einer Umgebung in einen spezielleren Teil und Speichern von einem speziellen Teil in eine größere Umgebung. Die Definition ist aber nicht so streng zu sehen.

Der Z80 kann 64K Bytes Speicher, also 65536 Speicherzellen mit je 8 Bit Datenbreite adressieren. Der Ladebefehl kann von einer dieser Zellen den Inhalt in das Register A transportieren. Wie wird der Ladebefehl nun angegeben. Der Befehl wird dem Z80 als Operations-Code zugeführt, ähnlich wie in dem Beispiel der Verkehrsampelsteuerung aus Kapitel 5. Damit besitzt er eine binäre Darstellung:

00111010 11111111 hhhhhhhh

wobei die Abkürzung l für den niederwertigen Adreßteil stehen und h für den höherwertigen Adreßteil. Doch nun werden Sie sicher bemerken, das sind ja nicht 8 Bits, sondern 24. Wie geht das bei einem 8 Bit-Rechner? Ganz recht. Der Operationscode ist tatsächlich 24 Bit lang, doch wird er, da er sonst nicht auf den Datenbus des Z80 paßt, in drei 8-Bit-Gruppen zerlegt. Das sieht dann so aus:

00111010
llllllll
hhhhhhhh

Schon besser! Doch, was hat es mit der höher- und niederwertigen Adresse auf sich? Um die 65536 Zellen zu adressieren, wird eine 16-Bit-Adresse benötigt. Diese Adresse muß in zwei Teile zerlegt werden. Der Z80 will dabei zuerst die untere Hälfte der Adreßbits haben und dann die obere Hälfte. Beispiel: Es soll von der Zelle 4567h in den Akku geladen werden. Die Adresse 4567h ist binär 0100010101100111b. Also ergibt sich

00111010
01100111
01000101

Nun ist diese binäre Schreibweise nicht gerade lesbar und daher wird im allgemeinen die hexadezimale Schreibweise bei der Angabe von Befehlscodes verwendet. In Hex lautet das ganze:

3A
67
45

Nun ist auch das noch nicht sehr lesbar, man stelle sich ein langes Programm vor, das nur aus Zahlenreihen besteht, wer soll sich merken und wissen, was dabei passiert? Daher hat sich jemand die mnemotechnische Darstellung von Operationscodes überlegt.
Der Befehl wird dann geschrieben:

LD A, (4567h)

Und nun wird die Funktion deutlich. LD steht für LOAD oder LADE, A steht für den Akku A und (4567h) steht für den Inhalt von 4567h, wobei ( ) immer als „Inhalt von" gelesen wird. Zur Aufschreibung der Programme wird die mnemotechnische Version verwendet und vor der Eingabe in den Computer wird die Darstellung in die HEX-Darstellung übersetzt. Das Übersetzen kann dabei von Hand geschehen, so wie wir es üben werden, oder es kann auch mit Hilfe des Computers selbst durchgeführt werden, der dazu ein Programm, den Assembler bekommt, der die Arbeit macht. Nun können wir uns einen Grundbefehlssatz zusammenstellen, der etwas lesbarer ist, als nur die reine Binärform.

Grundbefehlssatz:

Dazu folgende Konventionen:

adresse	ist eine Abkürzung dafür, daß hier eine beliebige 16-Bit-Zahl stehen darf
data	eine 8-Bit-Zahl (0..255) darf angegeben werden

Neben der allgemeinen Form ist noch ein Beispiel mit Codierung angegeben.

Alle Codeangaben sind auch ohne die Angabe des Zeichens h in HEX.

LD A,(adresse) LD A,(1234h) 3A 34 12

der Akkumulator A wird mit dem Inhalt der Zelle (adresse)

LD (adresse),A LD (1234h),A 32,34 12

der Inhalt des Akkumulators A wird auf die Adresse (adresse) abgespeichert. Dies ist die genaue Umkehrung des vorherigen Befehls.

LD A, B LD A, B 78

Der Inhalt des Registers B wird in den Akkumulator A geladen. Der Operationscode ist nur 1 Byte lang.

LD B,A LD B,A 47

Der Inhalt des Akkumulators A wird in das Register B gespeichert. Hier ist der Operations-Code nur ein Byte lang.

LD A,data LD A,5 3E 05

In den Akkumulator wird der Wert data geladen.

LD B,data LD B,5 06 05

In das Register B wird der Wert data geladen.

Nun folgen ein paar arithmetische Befehle:

ADD A,data ADD A,1 C6 01

Der Inhalt des Akkumulators A wird um data erhöht. Der Operationscode ist ein Zwei-Byte-Befehl.

Im Beispiel wird Register A um eins erhöht.

ADD A,B ADD A,B 80

Zum Inhalt des Akkumulators A wird der Inhalt des Registers B addiert.

SUB data SUB 1 D6 01

Vom Inhalt des Akkumulators wird der Wert data subtrahiert.

SUB B SUB B 90

Vom Inhalt des Akkumulators A wird der Inhalt des Registers B subtrahiert.

Wir haben schon aus Kapitel 5 gelernt, daß mit den obigen Befehlen nur ein lineares Programm aufgebaut werden kann. Wir wollen aber auch den Programmzähler verändern können und dazu gibt es die sogenannten Sprungbefehle.

JP adresse JP 1200h C3 00 12

Der nächste Befehl nach diesem wird von Adresse 1200h geholt. Damit wurde ein Programmsprung ausgeführt.

Es genügt nicht, nur springen zu können, dies muß auch von einer Entscheidung abhängig gemacht werden können. Dazu gibt es bedingte Sprungbefehle.

JP Z,adresse JP Z,1200h CA 00 12

Die Bedingung liegt in der Abfrage eines Bits, das als Null-Flag bekannt ist. Es wird bei arithmetischen Befehlen verändert und gibt an, wann der Inhalt eines Registers 0 wurde. Wurde der Akku A aufgrund einer Subtraktion 0, so wird das Null-Flag (Zero-Flag) gesetzt. Folgt daraufhin der obige Sprungbefehl, so wird er ausgeführt, wurde der Akku zuvor nicht 0 gesetzt, so ist das Flag (oder Merker) nicht gesetzt und der Sprung wird nicht ausgeführt.

JP NZ,adresse JP NZ,1205 C2 05 12

Hier wird der Sprung ausgeführt, wenn das Null-Flag nicht gesetzt war, das Verhalten ist also genau umgekehrt wie beim vorherigen Befehl.

Mit diesen Befehlen können wir nun ein paar Programm erstellen.

Es sollen zwei Zahlen addiert werden. Die beiden Zahlen stehen im Speicher. Die erste Zahl steht auf Adresse 1305h und die zweite Zahl soll auf 1306h stehen. Zum Addieren müssen die beiden Zahlen zunächst in die Register transportiert werden, da eine Addition nur innerhalb der Register möglich ist.
Es gilt also:

```
LD A,(1305h)    ; laden erster Operand
LD B,A          ; transport nach B
LD A,(1306h)    ; laden zweiter Operand
```

Einen Befehl LD B,(adresse) gibt es beim Z80 leider nicht. Dann wird die Addition durchgeführt:

```
ADD A,B
```

und und nun muß das Ergebnis noch abgespeichert werden. Dies kann mit dem Befehl

```
LD (1307h),A
```

durchgeführt werden. Das Resultat wird dann in Speicherzelle 1307h zu finden sein.

Das ganze Programm sieht dan so aus:

```
LD A,(1305h)
LD B,A
LD A,(1306h)
ADD A,B
LD (1307h),A
```

Nun ist dieses Programm aber dem Rechner noch nicht verständlich. Es muß dazu in Maschinensprache umgesetzt werden. Das Programm soll dabei auf Adresse 1200h beginnen. Die Umcodierung kann mit Hilfe der Liste von vorher durchgeführt werden und es ergibt sich:

```
1200    3A 05 13    LD A,(1305h)
1203    47          LD B,A
1204    3A 06 13    LD A,(1306h)
1207    80          ADD A,B
1208    32 07 13    LD (1307h),A
120B    ........    ............
```

Im Speicher wird die Sequenz dann in aufeinanderfolgende Zellen gelegt:

```
1200h:  3A  05  13  47  3A  06  13  80  32  07  13
```

Kommt der Prozessor auf die Stelle 1200h, so wird unser Programm ausgeführt.

Nun die Anwendung einer unbedingten Anweisung. Wir wollen die Verkehrsampelsteuerung nachbilden. Dazu benötigen wir eine Verbindung mit der Außenwelt. Der Z80 besitzt dafür eigene Befehle, die IO-Befehle:

```
IN A,(data)          IN A,(41h)          DB 41
```

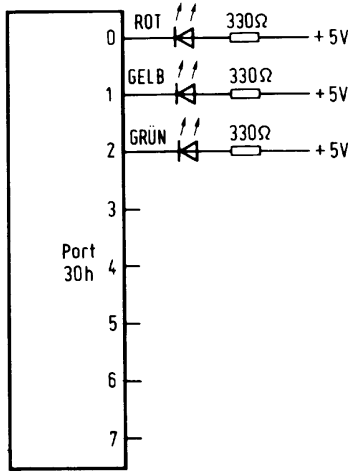


Abb. 6.1.2 Ampelsteuerung

Der Port data wird in den Akkumulator A geladen. Hier wird nur eine 8 Bit Adresse angegeben, da der Z80 nicht mehr als 256 IO-Adressen ansprechen kann.

OUT (data),A OUT (54h),A D3 54

Der Inhalt des Akkumulators A wird auf die Adresse 54h ausgegeben.

Mit diesen IO-Befehlen läßt sich die Steuerung realisieren. Dazu wird an einen Port, z. B. der mit Adresse 30h, die Verkehrsampel angeschaltet. Ein Port besitzt beim Z80 im allgemeinen 8 Datenbits, wir benötigen aber nur drei bei einer Ampel. *Abb. 6.1.2* zeigt den Anschluß. Dem Bit 0 ist das rote Signal zugeordnet, Bit 1 erhält das Signal Gelb und Bit 2 wird an Grün angeschlossen. Soll z. B. gelb leuchten und die anderen nicht, so muß am Ausgang des Ports Bit 1 auf 0 sein, die anderen auf 1.0 daher, weil die LEDs, die hier als Ausgabe verwendet wurden mit der Kathode am Ausgang liegen und daher nur dann leuchten, wenn sich ein 0V-Pegel am Ausgang befindet. Als erstes legen wir uns also die Bitmuster für die einzelnen Fälle zurecht.

ROT:	xxxxx110b
GELB:	xxxxx101b
GRÜN:	xxxxx011b
ROTGELB:	xxxxx100b

Nun muß der Ablauf des Programms festgelegt werden. Es soll folgen: ROT – ROTGELB – GRÜN – GELB – ROT – ROTGELB . . . . . Um diese Sequenz zu erhalten, müssen die einzelnen Bitmuster nacheinander an den Port gelegt werden.

Dazu wird das Bitmuster zunächst in den Akku A geladen und dann mit dem Ausgabebefehl an das Port ausgegeben. Die ganze Sequenz soll sich dann wiederholen und dazu verwenden wir den Sprungbefehl. Das Zeichen x steht für beliebig, da die restlichen Bits nicht verwendet wurden. Wir können es z. B. mit 0 belegen.

```
LD A,00000110b      ; Lade ROT
OUT (30h),A         ; Ausgabe ROT
LD A,00000100b      ; Lade ROTGELB
OUT (30h),A         ; Ausgabe ROTGELB
LD A,00000011b      ; Lade GRÜN
OUT (30h),A         ; Ausgabe GRÜN
LD A,00000101b      ; Lade GELB
OUT (30h),A         ; Ausgabe GELB
JP zurück           ; Rücksprung
```

Bei dem Sprungbefehl steht einfach JP zurück. Wohin zurück, ist hier die Frage. Dazu muß eine Marke angegeben werden, man nennt dies auch LABEL. Die Marke trägt die Bezeichnung „zurück".
Das Programm sieht dann wie folgt aus:

```
zurück:   LD A,00000110b    ; Lade ROT
          . . . wie vorher . . .
          JP zurück
```

Nun ist klar, wohin gesprungen werden soll. Der Doppelpunkt hinter der Marke „zurück" dient dabei als Kennung und macht deutlich, daß hier von irgendwoher hingesprungen werden soll. Nun muß das ganze noch kodiert werden. Wir wollen das Programm nun einmal auf Adresse 0 anfangen lassen.

```
0000  3E 06       zurück:  LD A,00000110b    ; Lade ROT
0002  D3 30                OUT (930h),A      ; Ausgabe ROT
0004  3E 04                LD A,00000100b    ; Lade ROTGELB
0006  D3 30                OUT (30h),A       ; Ausgabe ROTGELB
0008  3E 03                LD A,00000011b    ; Lade GRÜN
000A  D3 30                OUT (30h),A       ; Ausgabe GRÜN
000C  3E 05                LD A,00000101b    ; Lade GELB
000E  D3 30                OUT (30h),A       ; Ausgabe GELB
0010  C3 00 00             JP zurück         ; Rücksprung
```

Als Sprungadresse wird hier der Wert 0000 eingesetzt. Würde das Programm auf Adresse 1234h starten, so wäre die Adresse der Marke „zurück" 1234h und die Sequenz wäre dann:

```
1234  3E 06       zurück:  LD A,00000110b    ; Lade ROT
                   . . . . . . . . .
1244  C3 34 12             JP zurück         ; Rücksprung
```

Der Adreßteil wird dabei als 34 12 angegeben, da beim Z80 bei allen 16 Bit Adressen zuerst die niederwertige Hälfte im Operationscode angegeben wird und dann die höherwertige.

Unser Programm hat noch einen kleinen Schönheitsfehler. Es arbeitet einwandfrei, wenn der Z80 mit einem Einzeltakt betrieben wird. Doch normalerweise arbeitet die CPU mit einem 2 MHz-Takt und die Ampelsequenz wird dadurch so schnell durchlaufen,

daß man sie nicht mehr sieht. Nach jeder Ausgabe müßte also eine kurze Pause erfolgen, bevor der nächste Schritt erfolgen kann. Dies läßt sich mit einer sogenannten Warteschleife erreichen. Dies ist ein Programmteil, in dem die CPU sehr lange bleibt, bevor sie diesen wieder verläßt. Eine Warteschleife läßt sich zum Beispiel durch einen Zähler verwirklichen, der herunter gezählt wird. Der Zähler wird mit einer sehr großen Zahl vorbesetzt, daß selbst der Z80 eine Weile (Sekundenbereich bei unserer Ampel) braucht, um ihn herunterzuzählen.

Wie läßt sich ein Zähler realisieren? Indem der Wert 1 von einer Zelle subtrahiert wird. Und dies geschieht solange, bis der Wert = 0 ist.

Als Rückwärtszähler dient die Zelle 1300h. Sie muß am Anfang mit dem Startwert geladen werden:

```
        LD A,255d        ; laden mit maximalem Wert
        LD (1300h),A     ; und abspeichern in die Zelle
warte:  LD A,(1300h)     ; Zählschleife
        SUB 1            ; -1 bilden
        LD (1300h),A     ; wieder zurück
        JP NZ,warte      ; bis Akku = 0 wiederholen
- - -   Ende der Warteschleife - - -
```

Wie lange bleibt aber der Prozessor in dieser Schleife? Dazu müssen wir die Ausführungszeit pro Befehl kennen.

LD A,(adresse) benötigt 13 Taktzyklen also 13 * 500 ns = 6.5 ys bei einem 2 MHZ Takt
SUB data benötigt 7 Taktzyklen also 7 * 500 ns = 2.3 ys
LD (adresse),A benötigt 13 Taktzyklen also 13 * 500 ns = 6.5 ys
JP NZ,adresse benötigt 10 Takzyklen also 10 * 500 ns = 5 ys
Damit ergibt sich für einen Schleifendurchlauf:

Tges = 6.5 ys + 2.3 ys + 6.5 ys + 5 ys = 20.3 ys

Die Schleife wird 255 mal (nicht 256) durchlaufen, also ist

Twarte = 255 * 20.3 ys = 5176.5 ys = 5.1765 ms

5 ms sind aber noch zu wenig. Wir wollen z. B. 1 Sekunde haben. Dazu muß die Schleife 200 mal ausgeführt werden. Wir bauen also eine Schleife um die Schleife. Das Programm zeigt *Abb. 6.1.3*. Diesmal wurde erstmalig ein automatischer Übersetzer verwendet, um das Programm zu codieren, es läßt sich aber hier noch genausogut per Hand durchführen.

Der Assembler gibt beim Objekt-Code, so nennt man die codierten Befehle alle 16-Bit-Adressen in der lesbaren Form, also zuerst MSB und dann LBS aus. Bei der Eingabe in den Computer muß die Reihenfolge umgekehrt werden. Beispielsweise würde im Speicher ab Adresse 1200h folgende Sequenz stehen:

```
3E C8 32 00 13 3E FF  32 01 13 3A  01 13 D6  01 32 01 13
C2 0A 12 3A 00 13 D6  01 32 00 13  C2 05 12
```

Die Warteschleife müßte nun nach jeder Ausgabe auf den IO-Port (Befehl OUT (30h),A) eingeschoben werden. Sie wird daher vier Mal benötigt. Das Programm wird dadurch schon recht umfangreich. Bei Computern gibt es eine Möglichkeit, Routinen, die oft benötigt werden, nur einmal abzuspeichern und von verschiedenen Programmteilen her

```
                                ; Warteschleife fuer eine Sekunde start
                                ; auf adresse 1200h

1200    3E C8                           ld a,200        ;aeussere Schleife 200 Mal
1202    32 1300                         ld (1300h),a    ;zaehler 1
1205    3E FF                   warte1: ld a,255        ;innere Schleife 255 Mal
1207    32 1301                         ld (1301h),a    ;zaehler 2
120A    3A 1301                 warte2: ld a,(1301h)    ;zaehler 2 innen
120D    D6 01                           sub 1
120F    32 1301                         ld (1301h),a    ;-1
1212    C2 120A                         jp nz,warte2    ;bis = 0
1215    3A 1300                         ld a,(1300h)    ;aussen zaehler 1
1218    D6 01                           sub 1
121A    32 1300                         ld (1300h),a
121D    C2 1205                         jp nz,warte1    ;innen zaehler neu belegen

                                ; -------- schleife fertig --------
```

Abb. 6.1.3 Warteschleife

aufrufen zu können. Das ist die Unterprogrammtechnik. In *Abb. 6.1.4* ist ein Beispiel mit einem Unterprogramm gezeigt. Das Unterprogramm UPR wird zweimal vom Hauptprogramm aus aufgerufen. Nach jedem Aufruf muß das Hauptprogramm an der Stelle weiter fortgeführt werden, an der der Aufruf erfolgt. Dazu muß sich der Prozessor die Adresse merken, bei der der Aufruf erfolgte. Dies geschieht beim Z80 mit Hilfe des Stackpointers, der im Register SP liegt. Dort ist eine Adresse (16 Bit) untergebracht, die auf

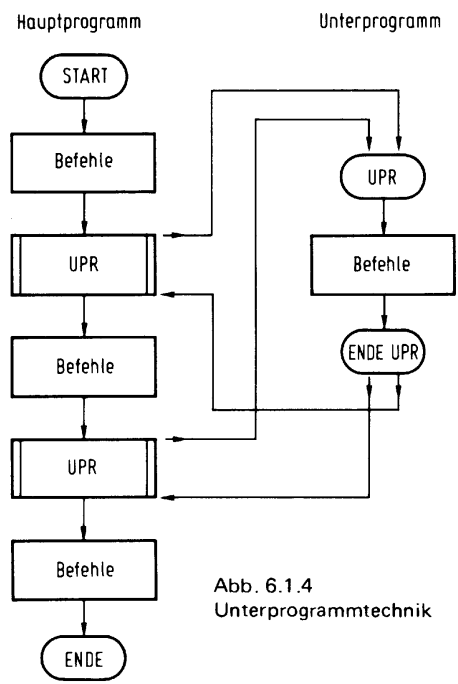


Abb. 6.1.4
Unterprogrammtechnik

eine Ramzelle zeigt. Bei einem Unterprogrammaufruf wird in zwei aufeinanderfolgende Speicherzellen, beginnend bei der Adresse des Stackpointers-1, der höherwertige Teil des Programmzählers abgespeichert und an der Stelle Stackpointer-2 wird der niederwertige Teil abgelegt. Danach wird der Stackpointer SP um zwei verringert. Der Programmzähler steht bei diesem Vorgang auf dem nächsten Befehl nach dem Aufruf-Befehl. Für das Unterprogramm gibt es einen Rücksprungbefehl. Er holt den niederwertigen Teil des Programmzählers von der Stelle SP, den höherwertigen Teil von SP+1 und anschließend wird der Stackpointer SP um zwei erhöht. Damit ist der Befehl genau die Umkehrung des Aufrufbefehls. Nun die Mnemonics der Unterprogrammbefehle:

CALL adresse CALL 1250h CD 50 12

Aufruf eines Unterprogramms auf Adresse 1250h. Der Programmzählerstand wird auf den Stack (so wird diese Art der Speicherung bezeichnet) gerettet. Der Stackpointer wird anschließend um zwei verringert.

RET RET C9

Rückkehr vom Unterprogramm in das Hauptprogramm. Die Rückkehradresse wird vom Stack zurückgeholt und der Stackpointer anschließend um zwei erhöht.

```
                        ; Unterprogrammtechnik

1200    31 13FF                 ld sp,13FFh     ;Stack pointer ans Ende
1203    3E 06           ampel:  ld a,00000110b  ;ROT
1205    D3 30                   out (30h),a
1207    CD 1222                 call warte
120A    3E 04                   ld a,00000100b  ;ROTGELB
120C    D3 30                   out (30h),a
120E    CD 1222                 call warte
1211    3E 03                   ld a,00000011b  ;GRUEN
1213    D3 30                   out (30h),a
1215    CD 1222                 call warte
1218    3E 05                   ld a,00000101b  ;GELB
121A    D3 30                   out (30h),a
121C    CD 1222                 call warte
121F    C3 1203                 jp ampel        ;dauernd wiederholen
                        ;
                        ; unterprogramm folgt nun
                        ;
1222    3E C8           warte:  ld a,200        ;aeussere Schleife 200 Mal
1224    32 1300                 ld (1300h),a    ;zaehler 1
1227    3E FF           war1:   ld a,255        ;innere Schleife 255 Mal
1229    32 1301                 ld (1301h),a    ;zaehler 2
122C    3A 1301         war2:   ld a,(1301h)    ;zaehler 2 innen
122F    D6 01                   sub 1
1231    32 1301                 ld (1301h),a    ;-1
1234    C2 122C                 jp nz,war2      ;bis = 0
1237    3A 1300                 ld a,(1300h)    ;aussen zaehler 1
123A    D6 01                   sub 1
123C    32 1300                 ld (1300h),a
123F    C2 1227                 jp nz,war1      ;innen zaehler neu belegen
1242    C9                      ret             ;Ruecksprung ins Hauptprogramm
                        ;
```

Abb. 6.1.5 Beispiel eines Unterprogramms

Wozu ist der Stackpointer gut, wenn es auch möglich wäre, die Rückkehradresse einfach in einem Register festzuhalten? Der Vorteil liegt darin, daß im Unterprogramm ein weiterer Unterprogrammaufruf erfolgen kann und in diesem wieder usw. Der Stackpointer wird dabei immer weiter verringert. Dies geht solange gut, bis der Speicherplatz für den Stack ausgeht, also der Bereich, den der Stackpointer mit seinen Adressen überstreicht. Es gibt auch Prozessoren, die die Rückkehradresse in ein Register legen, z. B. SCMP oder TMS 9900.

Zurück zu unserem Verkehrsampelproblem. *Abb. 6.1.5* zeigt die Lösung mit den Unterprogrammbefehlen. Dieses Programm kann auf unserem Mikrorechner auch tatsächlich ausgeführt werden.

Mit den bisherigen Befehlen lassen sich praktisch alle programmtechnischen Aufgaben lösen. Es ist sozusagen der Grundwortschatz, mit dem man sich verständlich machen kann. Der Z80 besitzt weitere Befehle, die es erlauben, die Programme eleganter zu schreiben. So ist es möglich, anstelle des Befehls ADD A,1 den Befehl INC A zu verwenden, der den Inhalt des Akkus um eins erhöht und nur einen Befehl für den Operationscode belegt (3Ch). Auch besitzt der Z80 Möglichkeiten, um direkt Arithmetik mit 16-Bit-Daten durchzuführen, was sehr wichtig für Adreßarithmetik ist. Ferner gibt es Befehle, mit denen eine Arithmetik für beliebig lange Zahlen gemacht werden kann. Der Z80 besitzt zur Abfrage von bedingten Anweisungen, wie auch der Befehl „JP NZ,adresse" eine war, ein Register mit dem Namen Statusregister. Es hat folgende Aufteilung:

7	6	5	4	3	2	1	0
S	Z	X	H	X	P/V	N	C

Die Bedeutung der einzelnen Bits:

C Carry-Flag. Es ist der Übertrag bei einer arithmetischen Operation. Kann aber auch mit Schiebebefehlen verändert werden. War ein Übertrag vorhanden, so wird es gesetzt. Der Übertrag ist der der vorzeichenlosen Arithmetik.

N Additions/Subtraktions-Flag. Es hat die Aufgabe einer ggf. nachfolgenden Dezimalkorrektur zu sagen, ob zuvor eine Addition oder Subtraktion ausgeführt wurde.

P/V Paritäts- oder Überlauf-Flag. Bei logischen Operationen gibt es die Parität des Akku A an. Ist das Bit Eins, so liegt eine gerade (even) Parität vor. Bei arithmetischen Operationen ist es das Überlaufbit bei Zweierkomplement-Arithmetik (vorzeichenbehaftet). Bei Überlauf ist das Bit gesetzt.

X Ist nicht verwendet.

H Halb-Überlauf. Wird für die Dezimalkorrektur benötigt und ist der Überlauf zwischen 3tem und 4tem Bit bei Arithmetik-Operationen.

Z Zero-Flag oder Null-Flag. Es wird bei arithmetischen und logischen Operationen gesetzt und ist Eins, wenn die bearbeitete Zelle 0 ist.

S Sign-Flag. Ist das Vorzeichenbit und wird bei arithmetischen und logischen Operationen gesetzt. Dabei wird das Bit auf Eins gesetzt, wenn Bit 7 der bearbeiteten Zelle Eins war, entspricht also bei einer Zweier-Komplement-Arithmetik exakt dem Vorzeichen.

```
                        ; Transport - Lade und
                        ; Speicherbefehle
02                              LD      (BC),A
12                              LD      (DE),A
77                              LD      (HL),A
70                              LD      (HL),B
71                              LD      (HL),C
72                              LD      (HL),D
73                              LD      (HL),E
74                              LD      (HL),H
75                              LD      (HL),L
36 20                           LD      (HL),' '        ;20H
DD 77 05                        LD      (IX+05),A
DD 70 05                        LD      (IX+05),B
DD 71 05                        LD      (IX+05),C
DD 72 05                        LD      (IX+05),D
DD 73 05                        LD      (IX+05),E
DD 74 05                        LD      (IX+05),H
DD 75 05                        LD      (IX+05),L
DD 36 05 20                     LD      (IX+05),20H
FD 77 05                        LD      (IY+05),A
FD 70 05                        LD      (IY+05),B
FD 71 05                        LD      (IY+05),C
FD 72 05                        LD      (IY+05),D
FD 73 05                        LD      (IY+05),E
FD 74 05                        LD      (IY+05),H
FD 75 05                        LD      (IY+05),L
FD 36 05 20                     LD      (IY+05),20H
32 84 05                        LD      (0584H),A
ED 43 84 05                     LD      (0584H),BC
ED 53 84 05                     LD      (0584H),DE
22 84 05                        LD      (0584H),HL
DD 22 84 05                     LD      (0584H),IX
FD 22 84 05                     LD      (0584H),IY
ED 73 84 05                     LD      (0584H),SP
0A                              LD      A,(BC)
1A                              LD      A,(DE)
7E                              LD      A,(HL)
DD 7E 05                        LD      A,(IX+05)
FD 7E 05                        LD      A,(IY+05)
3A 84 05                        LD      A,(0584H)
7F                              LD      A,A
78                              LD      A,B
79                              LD      A,C
7A                              LD      A,D
7B                              LD      A,E
7C                              LD      A,H
ED 57                           LD      A,I
7D                              LD      A,L
3E 20                           LD      A,' '   ;20H
ED 5F                           LD      A,R
```

Abb. 6.1.6 Transportbefehle

```
46                      LD      B,(HL)
DD 46 05                LD      B,(IX+05)
FD 46 05                LD      B,(IY+05)
47                      LD      B,A
40                      LD      B,B
41                      LD      B,C
42                      LD      B,D
43                      LD      B,E
44                      LD      B,H
45                      LD      B,L
06 20                   LD      B,' '   ;20H
ED 4B 84 05             LD      BC,(0584H)
01 84 05                LD      BC,0584H
4E                      LD      C,(HL)
DD 4E 05                LD      C,(IX+05)
FD 4E 05                LD      C,(IY+05)
4F                      LD      C,A
48                      LD      C,B
49                      LD      C,C
4A                      LD      C,D
4B                      LD      C,E
4C                      LD      C,H
4D                      LD      C,L
0E 20                   LD      C,' '   ;20H
56                      LD      D,(HL)
DD 56 05                LD      D,(IX+05)
FD 56 05                LD      D,(IY+05)
57                      LD      D,A
50                      LD      D,B
51                      LD      D,C
52                      LD      D,D
53                      LD      D,E
54                      LD      D,H
55                      LD      D,L
16 20                   LD      D,' '   ;20H
ED 5B 84 05             LD      DE,(0584H)
11 84 05                LD      DE,0584H
5E                      LD      E,(HL)
DD 5E 05                LD      E,(IX+05)
FD 5E 05                LD      E,(IY+05)
5F                      LD      E,A
58                      LD      E,B
59                      LD      E,C
5A                      LD      E,D
5B                      LD      E,E
5C                      LD      E,H
5D                      LD      E,L
1E 20                   LD      E,' '   ;20H
```

zu Abb. 6.1.6

```
66                      LD      H,(HL)
DD 66 05                LD      H,(IX+05)
FD 66 05                LD      H,(IY+05)
67                      LD      H,A
60                      LD      H,B
61                      LD      H,C
62                      LD      H,D
63                      LD      H,E
64                      LD      H,H
65                      LD      H,L
26 20                   LD      H,' '   ;20H
2A 84 05                LD      HL,(0584H)
21 84 05                LD      HL,0584H
ED 47                   LD      I,A
DD 2A 84 05             LD      IX,(0584H)
DD 21 84 05             LD      IX,0584H
FD 2A 84 05             LD      IY,(0584H)
FD 21 84 05             LD      IY,0584H
6E                      LD      L,(HL)
DD 6E 05                LD      L,(IX+05)
FD 6E 05                LD      L,(IY+05)
6F                      LD      L,A
68                      LD      L,B
69                      LD      L,C
6A                      LD      L,D
6B                      LD      L,E
6C                      LD      L,H
6D                      LD      L,L
2E 20                   LD      L,' '   ;20H
ED 4F                   LD      R,A
ED 7B 84 05             LD      SP,(0584H)
F9                      LD      SP,HL
DD F9                   LD      SP,IX
FD F9                   LD      SP,IY
31 84 05                LD      SP,0584H
```

zu Abb. 6.1.6

Die einzelnen Bits werden nicht immer alle bei allen Operationen gesetzt. Als Faustregel gilt die, bei den Flags angegebene Regel. Ein paar wichtige Fälle werden wir mit den Befehlen noch kennenlernen.
Die einzelnen Befehlsgruppen werden im folgenden besprochen.

1. Transportbefehle

Wir haben davon schon ein paar Befehle kennengelernt, z. B. LD A,(adresse).

Der Z80 besitzt aber eine ganze Reihe von solchen Befehlen, da er noch andere Register als A und B besitzt. *Abb. 6.1.6 a...c* zeigt alle möglichen Kombinationen des Befehls ‚LD'. Der Z80 hat die Möglichkeit, auch mit 16-Bit-Größen zu arbeiten. Dazu können die Register BC, DE, HL sowie IX, IY und natürlich SP verwendet werden. Diese Register

können direkt mit 16-Bit-Konstanten geladen werden, z. B. lädt der Befehl LD HL,1234h das Register H mit 12h und das Register L mit 34h. HL ist ein besonderes Registerpaar, es kann für eine 16-Bit-Arithmetik verwendet werden. Ebenfalls IX und IY. Diese drei Register sind dann praktisch ein neuer Akkumulator. Der Inhalt eines der anderen Registerpaare, also BC,DE und SP sowie auch HL (IX, IY) können dazu addiert werden. Nicht möglich ist es HL auf IX, IY oder umgekehrt zu addieren.

Eine andere Form der Adressierung ist die sogenannte indirekte Adresse. Dazu steht z. B. im Registerpaar HL eine Adresse. Mit der Speicherzelle, die durch diese Adresse bestimmt ist, kann dann gearbeitet werden. Beispiel:

In HL steht 1350h

In der Zelle 1350h steht der Wert 55h

Mit dem Befehl LD B,(HL) wird in das Register B der Wert 55h geladen.

Die Registerpaare IX und IY haben noch eine weitere Besonderheit. Zusätzlich kann ein Verschiebefaktor oder auch displacement genannt, angegeben werden. Er wird der Adresse, die im Registerpaar steht, aufaddiert und zuvor zu einer 16-Bit-Adresse ergänzt. Beispiel:

In IX steht der Wert 1310h

Mit LD C,(IX+40h)

wird der Inhalt der Speicherzelle 1350h in das Register C geladen.

Mit LD C,(IX+0FFH)

wird der Inhalt der Speicherzelle 130Fh in das Register C geladen.

Alle Transportbefehle verändern die Flags nicht. Damit ist es möglich, nach einem Transport noch den Zustand der vorherigen Operation, die die Flags verändert hat, zu sehen und eine bedingte Operation kann durchgeführt werden.

2. Austausch-Operationen

In *Abb. 6.1.7* sind Austausch-Befehle abgebildet. Sie sind eigentlich auch Transportbefehle, transportieren aber zwei oder mehr Operanden gleichzeitig. Der Z80 besitzt einen zweiten Registersatz. Die Registernamen werden mit einem Apostroph gekennzeichnet, der dem Registernamen hinten angestellt wird. Auf diese Register kann nicht direkt zugegriffen

```
                        ; Vertauschen eines Registerpaars
                        ; mit dem Inhalt des Stacks
E3                              EX      (SP),HL
DD E3                           EX      (SP),IX
FD E3                           EX      (SP),IY
                        ; Vertauschen der beiden Akkumulatoren
                        ; A und A', sowie der Flags
08                              EX      AF,AF'
                        ; Vertauschen von DE mit HL
EB                              EX      DE,HL
                        ; Vertauschen der beiden Registersaetze
                        ; BC,DE,HL
D9                              EXX
```

Abb. 6.1.7 Vertauschen von Registern

```
                    ; Transportiere den Inhalt der Zelle (HL) nach
                    ; Zelle (DE)
                    ; DE,HL,BC werden anschliessend decrementiert
ED A8                     LDD
                    ; Transportiere den Inhalt der Zelle (HL) nach
                    ; Zelle (DE)
                    ; DE,HL,BC werden anschliessend decrementiert
                    ; Wiederholung bis BC=0 ist
ED B8                     LDDR
                    ; Transportiere den Inhalt der Zelle (HL) nach
                    ; Zelle (DE)
                    ; DE,HL,werden incrementiert
                    ; BC wird anschliessend decrementiert
ED A0                     LDI
                    ; Transportiere den Inhalt der Zelle (HL) nach
                    ; Zelle (DE)
                    ; DE,HL,werden incrementiert
                    ; BC wird anschliessend decrementiert
                    ; Wiederholung bis BC=0 ist
ED B0                     LDIR
```

Abb. 6.1.8 Blocktransport-Befehle

werden. Der Registersatz muß erst aktiviert werden. Dazu gibt es zwei Befehle. Der erste Befehl EX AF,AF' vertauscht die beiden Akkumulatoren und die Statusregister. Danach wird immer mit dem zweiten Satz gearbeitet. Mit dem Befehl EXX werden die Register BC',DE',HL' gegen BC, DE und HL vertauscht.

3. Blocktransportbefehle

Abb. 6.1.8 zeigt die vier Befehle. Sie sind wohl die mächtigsten Operationen, die der Z80 besitzt. Mit ihnen kann ein ganzer Datenbereich verschoben werden. Beispiel:

```
LD HL,1000h
LD DE,2000h
LD BC,500h
LDIR
```

Der Bereich 1000h bis 14FFh wird nach 2000h bis 24 FFh transportiert. Mit diesem Befehl läßt sich auch das Löschen von Speicherbereichen durchführen. Beispiel:

```
LD HL,1000h
LD (HL),0
LD DE,1001h
LD BC,1FEh
LDIR
```

Der Bereich 1000h bis 11FFH wird mit dem Wert 0 belegt.

```
                        ; Das angegebene Registerpaar
                        ; wird mit dem oberen Stackwert
                        ; geladen
F1                              POP     AF
C1                              POP     BC
D1                              POP     DE
E1                              POP     HL
DD E1                           POP     IX
FD E1                           POP     IY
                        ; Der Inhalt des angegebenen Registerpaars
                        ; wird auf den Stack gespeichert
F5                              PUSH    AF
C5                              PUSH    BC
D5                              PUSH    DE
E5                              PUSH    HL
DD E5                           PUSH    IX
FD E5                           PUSH    IY
```

Abb. 6.1.9 Stack-Operationen

4. Stackoperationen

Der Stack, auch Keller genannt, ist vielleicht manchen Taschenrechner-Besitzern bekannt, und zwar denjenigen, die einen UPN-Rechner besitzen. Ein Stack ist ein Speicher, bei dem immer nur auf den zuletzt eingespeicherten Wert zugegriffen werden kann. Beim Z80 wird ein Stack mit Hilfe des Registerpaars SP aufgebaut. Dieses Registerpaar enthält dazu eine Adresse. Mit den Befehlen in *Abb. 6.1.9* kann auf den Stack zugegriffen werden. Der Stackpointer SP wird dabei automatisch verändert. Auch die Unterprogrammbefehle verwenden den Stack.

Als Beispiel soll das Registerpaar HL z. B. für eine Berechnung zwischengespeichert werden. Dies könnte mit dem Befehl LD (adresse),HL geschehen. Kürzer und eleganter geschieht es aber mit den Stackbefehlen

PUSH HL
. . diverse Operationen
POP HL

Mit dem letzten Befehl wird der alte Inhalt des Registerpaares HL wieder hergestellt. Wichtig ist bei den Befehlen, daß dazu immer zwei Operationen gehören. PUSH und POP. Danach hat der Stackpointer wieder den alten Wert. Die Operationen können aber beliebig geschachtelt werden. Nach dem PUSH-Befehl wird der Stackpointer um zwei verringert, nach dem POP-Befehl um zwei erhöht. Beispiel:

SP enthält den Wert 1234h
HL enthält den Wert 55AAh
PUSH HL

Danach enthält die Zelle 1233h den Wert 55h, also den MSB-Wert und die Zelle 1232h den Wert AAh, also LSB. Der Stackpointer steht dann auf 1232h.

```
                        ; Vergleich eines Operanden mit dem
                        ; Inhalt des Akku A
BE                              CP      (HL)
DD BE 05                        CP      (IX+05)
FD BE 05                        CP      (IY+05)
BF                              CP      A
B8                              CP      B
B9                              CP      C
BA                              CP      D
BB                              CP      E
BC                              CP      H
BD                              CP      L
FE 20                           CP      ' '     ;20H


                        ; Vergleich der Speicherstelle (HL) und
                        ; des Akku A, HL und BC wird decrementiert
ED A9                           CPD
                        ; Vergleich (HL) mit Akku A; HL,BC decrement
                        ; bis BC=0
ED B9                           CPDR
                        ; Vergleich (HL) mit Akku A; HL increment
                        ; BC decrement
ED A1                           CPI
                        ; Vergleich (HL) mit Akku A; HL increment
                        ; BC decrement bis BC=0
ED B1                           CPIR
```

Abb. 6.1.10 Vergleichsbefehle

Bei POP wird genau umgekehrt verfahren. Mit dem Befehl PUSH AF wird der Inhalt des Akkumulators A und das Statusregister auf den Stack geladen.

5. Vergleichsbefehle

Nun kommt eine sehr wichtige Gruppe. *Abb. 6.1.10*. Mit diesen Befehlen ist es möglich, Vergleiche von Operanden durchzuführen. Dabei werden auch die Flags gesetzt. Der Befehl CP wirkt dabei wie eine Subtraktion. Das Ergebnis der Subtraktion wird allerdings nicht abgespeichert, sondern nur die Flags. Verändert werden alle Flags. Beispiele:

CP 20h

Ist im Akku A der Wert 20h enthalten, so wird das Zero-Flag gesetzt, ansonsten wird es rückgesetzt. Ist der Wert des Akkus größer oder gleich dem Wert 20h, so wird das Carry-Flag zurückgesetzt, sonst wird es gesetzt, da bei der Subtraktion „a-20h" ein Übertrag stattfindet. Das Vorzeichen-Bit wird gemäß des Ergebnisses gesetzt.

Eine Besonderheit bieten die Blockvergleichsbefehle. Das Vorzeichen-Flag wird nach dem Ergebnis gesetzt. Das Null-Flag wird gesetzt, wenn gilt A = (HL). Das P/V-Flag wird hier besonders verwendet. Es wird gesetzt, wenn gilt BC-1 $<>$ 0, der Inhalt von BC vermindert um eins, ist ungleich Null. Das Carry-Flag wird nicht beeinflußt. Mit den Befehlen können Zeichen in einem Speicherbereich gesucht werden.

```
                        ; Addition mit Carry
                        ; nach Akku A
89                              ADC     A,C
8A                              ADC     A,D
8B                              ADC     A,E
8C                              ADC     A,H
8D                              ADC     A,L
CE 20                           ADC     A,' '   ;20H
                        ; Addition mit Carry
                        ; nach Registerpaar HL
ED 4A                           ADC     HL,BC
ED 5A                           ADC     HL,DE
ED 6A                           ADC     HL,HL
ED 7A                           ADC     HL,SP
                        ; Addition ohne Carry
                        ; nach Akku A
81                              ADD     A,C
82                              ADD     A,D
83                              ADD     A,E
84                              ADD     A,H
85                              ADD     A,L
C6 20                           ADD     A,' '   ;20H
                        ; Addition ohne Carry
                        ; nach Registerpaar HL
09                              ADD     HL,BC
19                              ADD     HL,DE
29                              ADD     HL,HL
39                              ADD     HL,SP
                        ; Addition ohne Carry
                        ; nach Registerpaar IX
DD 09                           ADD     IX,BC
DD 19                           ADD     IX,DE
DD 29                           ADD     IX,IX
DD 39                           ADD     IX,SP
                        ; Addition ohne Carry
                        ; nach Registerpaar IY
FD 09                           ADD     IY,BC
FD 19                           ADD     IY,DE
FD 29                           ADD     IY,IY
FD 39                           ADD     IY,SP
```

Abb. 6.1.11 Additionsbefehle

6. Arithmetik Befehle

Abb. 6.1.11 zeigt die Additionsbefehle. 16-Bit-Arithmetik ist ebenfalls möglich. Zu unterscheiden ist zwischen einer Addition mit Carry oder einer ohne. Die Flags werden in beiden Fällen entsprechend dem Ergebnis gesetzt, auch das Carry-Flag. Aber bei der Addition mit Carry wird der Zustand des alten Carry-Flags noch hinzuaddiert. Damit läßt sich eine Mehrfach-Genaue-Arithmetik realisieren. Beispielsweise sollen eine 24-Bit-Zahl zu einer anderen 24-Bit-Zahl addiert werden. In Register B C D steht die erste Zahl, in

Register E H L die zweite. Die höherwertige Ziffer steht dabei in B und bei der zweiten Zahl in E. Das Ergebnis soll in E H L stehen. Es gilt dann:

```
LD A,D      ; LSB zuerst
ADD A,L     ; ohne Carry addieren
LD L,A      ; Ergebnis LSB
LD A,C
ADC A,H     ; mit Carry
LD H,A
LD A,B
ADC A,E     ; MSB mit Carry
LD E,A      ; Ergebnis in EHL
```

```
                        ; Der angegebene Operand wird
                        ; vom Inhalt des Akkumulators
                        ; subtrahiert
96                              SUB     (HL)
DD 96 05                        SUB     (IX+05)
FD 96 05                        SUB     (IY+05)
97                              SUB     A
90                              SUB     B
91                              SUB     C
92                              SUB     D
93                              SUB     E
94                              SUB     H
95                              SUB     L
D6 20                           SUB     ' '     ;20H
                        ; Subtrahiere Operanden
                        ; von Akkumulator mit
                        ; Carry
9E                              SBC     A,(HL)
DD 9E 05                        SBC     A,(IX+05)
FD 9E 05                        SBC     A,(IY+05)
9F                              SBC     A,A
98                              SBC     A,B
99                              SBC     A,C
9A                              SBC     A,D
9B                              SBC     A,E
9C                              SBC     A,H
9D                              SBC     A,L
DE 20                           SBC     A,' '   ;20H
                        ; Subtrahiere Operanden
                        ; von Registerpaar HL
                        ; mit Carry
ED 42                           SBC     HL,BC
ED 52                           SBC     HL,DE
ED 62                           SBC     HL,HL
ED 72                           SBC     HL,SP
```

Abb. 6.1.12 Subtraktionsbefehle

```
                        ; Incrementieren eines Operanden
                        ; Der Wert des Operanden wird um
                        ; Eins erhoeht. Bei Registerpaarangabe
                        ; wird mit 16 Bits gearbeitet
34                              INC     (HL)
DD 34 05                        INC     (IX+05)
FD 34 05                        INC     (IY+05)
3C                              INC     A
04                              INC     B
03                              INC     BC
0C                              INC     C
14                              INC     D
13                              INC     DE
1C                              INC     E
24                              INC     H
23                              INC     HL
DD 23                           INC     IX
FD 23                           INC     IY
2C                              INC     L
33                              INC     SP
                        ; Decrementieren eines Operanden,
                        ; der Operandenwert wird um eins
                        ; verringert
                        ; Dabei werden Doppelregister als
                        ; 16-Bit Groessen behandelt
35                              DEC     (HL)
DD 35 05                        DEC     (IX+05)
FD 35 05                        DEC     (IY+05)
3D                              DEC     A
05                              DEC     B
0B                              DEC     BC
0D                              DEC     C
15                              DEC     D
1B                              DEC     DE
1D                              DEC     E
25                              DEC     H
2B                              DEC     HL
DD 2B                           DEC     IX
FD 2B                           DEC     IY
2D                              DEC     L
3B                              DEC     SP
```

Abb. 6.1.13 Increment- und Decrement-Befehle

Bei einer 16-Bit-Addition ist das einfacher. Beispiel: Es soll 1234h zum Inhalt von HL addiert werden

```
LD BC,1234h     ; z. B. mit BC durchführen
ADD HL,BC       ; ohne Carry Ergebnis in HL
```

Bei der Subtraktion ist das genausc. *Abb. 6.1.12* zeigt die Befehle. Die Flags werden auch entsprechend gesetzt.

```
                        ; Logisch Und-Verknuepfung
                        ; nach Akku A
A6                              AND     (HL)
DD A6 05                        AND     (IX+05)
FD A6 05                        AND     (IY+05)
A7                              AND     A
A0                              AND     B
A1                              AND     C
A2                              AND     D
A3                              AND     E
A4                              AND     H
A5                              AND     L
E6 20                           AND     ' '     ;20H
                        ; Logisch Oder-Verknuepfung von Operand
                        ; und Akku A
B6                              OR      (HL)
DD B6 05                        OR      (IX+05)
FD B6 05                        OR      (IY+05)
B7                              OR      A
B0                              OR      B
B1                              OR      C
B2                              OR      D
B3                              OR      E
B4                              OR      H
B5                              OR      L
F6 20                           OR      ' '     ;20H
                        ; Logische Exclusiv-Oder-Verknuepfung
                        ; eines Operanden mit dem Akku A
AE                              XOR     (HL)
DD AE 05                        XOR     (IX+05)
FD AE 05                        XOR     (IY+05)
AF                              XOR     A
A8                              XOR     B
A9                              XOR     C
AA                              XOR     D
AB                              XOR     E
AC                              XOR     H
AD                              XOR     L
EE 20                           XOR     ' '     ;20H
                        ;
                        ;
```

Abb. 6.1.14 Logische Verknüpfungen

Ein Sonderfall ist in *Abb. 6.1.13* dargestellt. Es sind die Increment- und Decrement-Befehle. Sie können Register oder auch Speicherplätze um den Wert 1 erhöhen oder erniedrigen und eignen sich daher besonders zur Bearbeitung von Zählern. Die Flags werden aber hier besonders behandelt. Alle Doppelregister-Befehle wie z. B. INC HL verändern die Flags überhaupt nicht, bei den Einzelregister-Befehlen werden nur Zero-Flag, Sign-Flag, N-Flag und H-Flag, nicht aber das Carry-Flag beeinflußt.

7. Logische Operationen

Neben arithmetischen Befehlen gibt es bei Computern auch Befehle, um logische Verknüpfungen durchzuführen. *Abb. 6.1.14* zeigt die Möglichkeiten. Es gibt dabei drei verschiedene Operationen, AND, OR und XOR, die den Verknüpfungen UND, ODER und EXCLUSIV-ODER entsprechen. Die Verknüpfungen werden in dem Akkumulator bitweise durchgeführt. Beispiel:

```
LD A,10110011b
AND 11011101b
```

Anschließend steht im Akku der Wert 10010001b.
Wird dann der Befehl

```
OR  11110000b
```

durchgeführt, so ergibt sich 11110001b im Akku. Folgt dann

```
XOR 11001100b
```

ist der Akkuinhalt schließlich 00111101b.

```
                    ; Carry-Flag komplementieren
3F                          CCF
                    ; Einer-Komplement des Akkus wird
                    ; gebildet
2F                          CPL
                    ; Dezimal-Korrektur wird ausgefuehrt
27                          DAA
                    ; Sperren des Interrupts
F3                          DI
                    ; Freigabe des Interrupts
FB                          EI

                    ; Stop. Warten auf RESET oder Interrupt
76                          HALT
                    ; Interrupt Mode setzen
                    ; IM 0 nach Reset voreingestellt
ED 46                       IM      0
ED 56                       IM      1
ED 5E                       IM      2

                    ; Bilde das Zweier-Komplement des Akku A
ED 44                       NEG
                    ; Keine Operation
00                          NOP

                    ; Setze das Carry-Flag
37                          SCF
```

Abb. 6.1.15 Einzelbefehle

8. Diverse Einzelbefehle

Abb. 6.1.15 zeigt eine Reihe einzelner Befehle. Mit dem Befehl CPL ist es zum Beispiel möglich, das Einerkomplement einer Zahl zu bilden. Beispiel:

```
Akku vorher:    11011011b
                CPL
nachher:        00100100b
```

Mit dem Zweierkomplementbefehl sieht das anders aus:

```
Akku vorher:    11011011b
                NEG
Akku nachher    00100101b
```

Der NOP-Befehl bewirkt nichts und wird gern als Verzögerung verwendet.

Der Befehl DAA kann zur Dezimal-Korrektur verwendet werden, um mit BCD-Zahlen zu arbeiten. Beispiel:

```
Akku: 49h       ; BCD ZAHL 4 9
ADD A,27h       ; BCD ZAHL 2 7
DAA
Akku: 76h       ; BCD ZAHL 7 6
```

Die Subtraktion ist auch möglich.

9. Rotationsbefehle

Mit der Rotation kann im Prinzip ein Schieberegister realisiert werden. Dazu gibt es verschiedene Varianten, die *Abb. 6.1.16* zeigt. Dort sind logische Operationen gezeigt, die das Register komplett, also im Kreis herum rotieren lassen. Die Befehle, die durch das Carry hindurchschieben, schieben eigentlich 9 Bits.

```
Beispiele:      Akku zuvor auf 10011001b   Carry = 0
    RLCA
    Akku        00110011b                  Carry = 1
    RLA
    Akku        01100111b                  Carry = 0
```

bei RRCA und RRA entsprechend umgekehrt.

Eine Besonderheit sind die Befehle RLD und RRD. Dort können Operationen auf Digits durchgeführt werden, also mit 4-Bit-Größen. Bei RLD werden die Bits 0 . . 3 des Akkus nach (HL) Bits 0 . . 3 transportiert, die alten Bits 0 . . 3 von (HL) (sprich Speicherzelle, die durch HL adressiert wurde), werden nach Bits 4 . . 7 von (HL) transportiert und diese alten Bits nach Bits 0 . . 3 des Akkus. Der Befehl RRD kehrt den Vorgang exakt um.

In *Abb. 6.1.17* sind noch weitere Schiebebefehle abgebildet. Bei den arithmetischen Schiebebefehlen wird so verfahren, als ob eine Division durch zwei bei den Rechts-Schiebebefehlen und eine Multiplikation mit 2 bei den Links-Schiebebefehlen durchgeführt wird. Das Carry-Flag wird als Überlaufbit behandelt.

```
                        ; Rotiere Links durch das Carry-Flag
CB 16                           RL      (HL)
DD CB 05 16                     RL      (IX+05)
FD CB 05 16                     RL      (IY+05)
CB 17                           RL      A
CB 10                           RL      B
CB 11                           RL      C
CB 12                           RL      D
CB 13                           RL      E
CB 14                           RL      H
CB 15                           RL      L
                        ; Rotiere den Inhalt des Akkumulators
                        ; links durch das Carry-Flag
17                              RLA
                        ; Rotiere Links circular
CB 06                           RLC     (HL)
DD CB 05 06                     RLC     (IX+05)
FD CB 05 06                     RLC     (IY+05)
CB 07                           RLC     A
CB 00                           RLC     B
CB 01                           RLC     C
CB 02                           RLC     D
CB 03                           RLC     E
CB 04                           RLC     H
CB 05                           RLC     L
                        ; Rotiere den Inhalt des
                        ; Akkumulators links circular
07                              RLCA
                        ; Rotiere eine Digitgroesse (4Bit)
                        ; links und rechts zwischen Akku A
                        ; und der Zelle (HL)
ED 6F                           RLD
                        ; Rotiere rechts durch das Carry-Flag
CB 1E                           RR      (HL)
DD CB 05 1E                     RR      (IX+05)
FD CB 05 1E                     RR      (IY+05)
CB 1F                           RR      A
CB 18                           RR      B
CB 19                           RR      C
CB 1A                           RR      D
CB 1B                           RR      E
CB 1C                           RR      H
CB 1D                           RR      L
                        ; Rotiere den Inhalt des Akkumulators
                        ; rechst durch das Carry-Flag
1F                              RRA
                        ; Rotiere rechts circular
CB 0E                           RRC     (HL)
DD CB 05 0E                     RRC     (IX+05)
FD CB 05 0E                     RRC     (IY+05)
CB 0F                           RRC     A
CB 08                           RRC     B
CB 09                           RRC     C
CB 0A                           RRC     D
CB 0B                           RRC     E
CB 0C                           RRC     H
CB 0D                           RRC     L
```

Abb. 6.1.16 Schiebebefehle Rotationen

```
                        ; Rotiere den Inhalt des Akku A
                        ; rechts circular
0F                              RRCA
                        ; Rotiere ein Digit (4Bits)
                        ; rechts und links zwischen
                        ; Akku A und Zelle (HL)
ED 67                           RRD
                        ; Arithmetische Schiebeoperation
                        ; nach links (nachfuellen mit 0)
CB 26                           SLA     (HL)
DD CB 05 26                     SLA     (IX+05)
FD CB 05 26                     SLA     (IY+05)
CB 27                           SLA     A
CB 20                           SLA     B
CB 21                           SLA     C
CB 22                           SLA     D
CB 23                           SLA     E
CB 24                           SLA     H
CB 25                           SLA     L
                        ; Arithmetische Schieboperation
                        ; nach rechst (duplizieren des
                        ; Vorzeichen Bits 7)
CB 2E                           SRA     (HL)
DD CB 05 2E                     SRA     (IX+05)
FD CB 05 2E                     SRA     (IY+05)
CB 2F                           SRA     A
CB 28                           SRA     B
CB 29                           SRA     C
CB 2A                           SRA     D
CB 2B                           SRA     E
CB 2C                           SRA     H
CB 2D                           SRA     L
                        ; Logisches Schieben nach rechts
                        ; mit 0 nachfuellen
CB 3E                           SRL     (HL)
DD CB 05 3E                     SRL     (IX+05)
FD CB 05 3E                     SRL     (IY+05)
CB 3F                           SRL     A
CB 38                           SRL     B
CB 39                           SRL     C
CB 3A                           SRL     D
CB 3B                           SRL     E
CB 3C                           SRL     H
CB 3D                           SRL     L
```

Abb. 6.1.17 Schiebebefehle

10. Bit-Operationen

Sehr komfortabel sind die Bitmanipulations-Befehle des Z80. *Abb. 6.1.18 a + b* zeigt die Befehle, um den Zustand eines Bits abzufragen. Das Zero-Flag wird dabei exakt so gesetzt,

```
                                ; Testen eines Bits
                                ; setzen des Zero-Flags
                                ; entsprechend des Wertes
CB 46                                   BIT     0,(HL)
DD CB 05 46                             BIT     0,(IX+05)
FD CB 05 46                             BIT     0,(IY+05)
CB 47                                   BIT     0,A
CB 40                                   BIT     0,B
CB 41                                   BIT     0,C
CB 42                                   BIT     0,D
CB 43                                   BIT     0,E
CB 44                                   BIT     0,H
CB 45                                   BIT     0,L
CB 4E                                   BIT     1,(HL)
DD CB 05 4E                             BIT     1,(IX+05)
FD CB 05 4E                             BIT     1,(IY+05)
CB 4F                                   BIT     1,A
CB 48                                   BIT     1,B
CB 49                                   BIT     1,C
CB 4A                                   BIT     1,D
CB 4B                                   BIT     1,E
CB 4C                                   BIT     1,H
CB 4D                                   BIT     1,L
CB 56                                   BIT     2,(HL)
DD CB 05 56                             BIT     2,(IX+05)
FD CB 05 56                             BIT     2,(IY+05)
CB 57                                   BIT     2,A
CB 50                                   BIT     2,B
CB 51                                   BIT     2,C
CB 52                                   BIT     2,D
CB 53                                   BIT     2,E
CB 54                                   BIT     2,H
CB 55                                   BIT     2,L
CB 5E                                   BIT     3,(HL)
DD CB 05 5E                             BIT     3,(IX+05)
FD CB 05 5E                             BIT     3,(IY+05)
CB 5F                                   BIT     3,A
CB 58                                   BIT     3,B
CB 59                                   BIT     3,C
CB 5A                                   BIT     3,D
CB 5B                                   BIT     3,E
CB 5C                                   BIT     3,H
CB 5D                                   BIT     3,L
CB 66                                   BIT     4,(HL)
DD CB 05 66                             BIT     4,(IX+05)
FD CB 05 66                             BIT     4,(IY+05)
CB 67                                   BIT     4,A
CB 60                                   BIT     4,B
```

Abb. 6.1.18 Einzelbittest-Befehle

```
CB 61                BIT   4,C
CB 62                BIT   4,D
CB 63                BIT   4,E
CB 64                BIT   4,H
CB 65                BIT   4,L
CB 6E                BIT   5,(HL)
DD CB 05 6E          BIT   5,(IX+05)
FD CB 05 6E          BIT   5,(IY+05)
CB 6F                BIT   5,A
CB 68                BIT   5,B
CB 69                BIT   5,C
CB 6A                BIT   5,D
CB 6B                BIT   5,E
CB 6C                BIT   5,H
CB 6D                BIT   5,L
CB 76                BIT   6,(HL)
DD CB 05 76          BIT   6,(IX+05)
FD CB 05 76          BIT   6,(IY+05)
CB 77                BIT   6,A
CB 70                BIT   6,B
CB 71                BIT   6,C
CB 72                BIT   6,D
CB 73                BIT   6,E
CB 74                BIT   6,H
CB 75                BIT   6,L
CB 7E                BIT   7,(HL)
DD CB 05 7E          BIT   7,(IX+05)
FD CB 05 7E          BIT   7,(IY+05)
CB 7F                BIT   7,A
CB 78                BIT   7,B
CB 79                BIT   7,C
CB 7A                BIT   7,D
CB 7B                BIT   7,E
CB 7C                BIT   7,H
CB 7D                BIT   7,L
```

zu Abb. 6.1.18

wie der Zustand des Bits war. War der Zustand des Bits eine Eins, so wird das Null-Flag rückgesetzt, um anzuzeigen, daß eine NICHT-NULL-Bedingung vorliegt.

Abb. 6.1.19 a + b zeigt Befehle, mit denen ein einzelnes Bit gesetzt werden kann. Beispiel:

Akku vorher 10000100b
SET 4,A
Akku nachher 10010100b

Mit den Befehlen aus *Abb. 6.1.20 a + b* können einzelne Bits auch wieder zurückgesetzt werden.

```
                ; Setzen eines Bits
CB C6                   SET     0,(HL)
DD CB 05 C6             SET     0,(IX+05)
FD CB 05 C6             SET     0,(IY+05)
CB C7                   SET     0,A
CB C0                   SET     0,B
CB C1                   SET     0,C
CB C2                   SET     0,D
CB C3                   SET     0,E
CB C4                   SET     0,H
CB C5                   SET     0,L
CB CE                   SET     1,(HL)
DD CB 05 CE             SET     1,(IX+05)
FD CB 05 CE             SET     1,(IY+05)
CB CF                   SET     1,A
CB C8                   SET     1,B
CB C9                   SET     1,C
CB CA                   SET     1,D
CB CB                   SET     1,E
CB CC                   SET     1,H
CB CD                   SET     1,L
CB D6                   SET     2,(HL)
DD CB 05 D6             SET     2,(IX+05)
FD CB 05 D6             SET     2,(IY+05)
CB D7                   SET     2,A
CB D0                   SET     2,B
CB D1                   SET     2,C
CB D2                   SET     2,D
CB D3                   SET     2,E
CB D4                   SET     2,H
CB D5                   SET     2,L
CB DE                   SET     3,(HL)
DD CB 05 DE             SET     3,(IX+05)
FD CB 05 DE             SET     3,(IY+05)
CB DF                   SET     3,A
CB D8                   SET     3,B
CB D9                   SET     3,C
CB DA                   SET     3,D
CB DB                   SET     3,E
CB DC                   SET     3,H
CB DD                   SET     3,L
CB E6                   SET     4,(HL)
DD CB 05 E6             SET     4,(IX+05)
FD CB 05 E6             SET     4,(IY+05)
CB E7                   SET     4,A
CB E0                   SET     4,B
CB E1                   SET     4,C
CB E2                   SET     4,D
CB E3                   SET     4,E
CB E4                   SET     4,H
CB E5                   SET     4,L
CB EE                   SET     5,(HL)
DD CB 05 EE             SET     5,(IX+05)
FD CB 05 EE             SET     5,(IY+05)
CB EF                   SET     5,A
CB E8                   SET     5,B
CB E9                   SET     5,C
CB EA                   SET     5,D
CB EB                   SET     5,E
CB EC                   SET     5,H
CB ED                   SET     5,L
CB F6                   SET     6,(HL)
DD CB 05 F6             SET     6,(IX+05)
FD CB 05 F6             SET     6,(IY+05)
CB F7                   SET     6,A
CB F0                   SET     6,B
CB F1                   SET     6,C
CB F2                   SET     6,D
CB F3                   SET     6,E
CB F4                   SET     6,H
CB F5                   SET     6,L
CB FE                   SET     7,(HL)
DD CB 05 FE             SET     7,(IX+05)
FD CB 05 FE             SET     7,(IY+05)
CB FF                   SET     7,A
CB F8                   SET     7,B
CB F9                   SET     7,C
CB FA                   SET     7,D
CB FB                   SET     7,E
CB FC                   SET     7,H
CB FD                   SET     7,L
```

Abb. 6.1.19
Einzelbitsetz-Befehle

```
                              ; Ruecksetzen des angegebenen Bits in
CB 86                                 RES     0,(HL)
DD CB 05 86                           RES     0,(IX+05)
FD CB 05 86                           RES     0,(IY+05)
CB 87                                 RES     0,A
CB 80                                 RES     0,B
CB 81                                 RES     0,C
CB 82                                 RES     0,D
CB 83                                 RES     0,E
CB 84                                 RES     0,H
CB 85                                 RES     0,L
CB 8E                                 RES     1,(HL)
DD CB 05 8E                           RES     1,(IX+05)
FD CB 05 8E                           RES     1,(IY+05)
CB 8F                                 RES     1,A
CB 88                                 RES     1,B
CB 89                                 RES     1,C
CB 8A                                 RES     1,D
CB 8B                                 RES     1,E
CB 8C                                 RES     1,H
CB 8D                                 RES     1,L
CB 96                                 RES     2,(HL)
DD CB 05 96                           RES     2,(IX+05)
FD CB 05 96                           RES     2,(IY+05)
CB 97                                 RES     2,A
CB 90                                 RES     2,B
CB 91                                 RES     2,C
CB 92                                 RES     2,D
CB 93                                 RES     2,E
CB 94                                 RES     2,H
CB 95                                 RES     2,L
CB 9E                                 RES     3,(HL)
DD CB 05 9E                           RES     3,(IX+05)
FD CB 05 9E                           RES     3,(IY+05)
CB 9F                                 RES     3,A
CB 98                                 RES     3,B
CB 99                                 RES     3,C
CB 9A                                 RES     3,D
CB 9B                                 RES     3,E
CB 9C                                 RES     3,H
CB 9D                                 RES     3,L
CB A6                                 RES     4,(HL)
DD CB 05 A6                           RES     4,(IX+05)
FD CB 05 A6                           RES     4,(IY+05)
CB A7                                 RES     4,A
CB A0                                 RES     4,B
CB A1                                 RES     4,C
CB A2                                 RES     4,D
CB A3                                 RES     4,E
CB A4                                 RES     4,H
CB A5                                 RES     4,L
```

Abb. 6.1.20 Einzelbitrücksetz-Befehle

```
CB AE                RES   5,(HL)
DD CB 05 AE          RES   5,(IX+05)
FD CB 05 AE          RES   5,(IY+05)
CB AF                RES   5,A
CB A8                RES   5,B
CB A9                RES   5,C
CB AA                RES   5,D
CB AB                RES   5,E
CB AC                RES   5,H
CB AD                RES   5,L
CB B6                RES   6,(HL)
DD CB 05 B6          RES   6,(IX+05)
FD CB 05 B6          RES   6,(IY+05)
CB B7                RES   6,A
CB B0                RES   6,B
CB B1                RES   6,C
CB B2                RES   6,D
CB B3                RES   6,E
CB B4                RES   6,H
CB B5                RES   6,L
CB BE                RES   7,(HL)
DD CB 05 BE          RES   7,(IX+05)
FD CB 05 BE          RES   7,(IY+05)
CB BF                RES   7,A
CB B8                RES   7,B
CB B9                RES   7,C
CB BA                RES   7,D
CB BB                RES   7,E
CB BC                RES   7,H
CB BD                RES   7,L
```

zu Abb. 6.1.20

11. Sprungbefehle

Abb. 6.1.21 zeigt verschiedene Sprungbefehle. Dabei hat der Z80 zwei unterschiedliche Arten. Die Sprünge mit absoluten Adressen und solche mit einer relativen Adresse. Bei den Sprüngen mit relativer Adresse ergeben sich zwei Vorteile. Zum einen ist der Operations-Code kürzer und ferner wird das Programmstück unabhängig von der Lage im Speicher. Dazu wird der angegebene 8-Bit-Wert als Zweierkomplement-Zahl aufgefaßt und zum aktuellen Stand des Programmzählers addiert. Beispiel:

```
            JR SKIP
            NOP
SKIP:       NOP
```

ist kodiert:

```
18 01       JR SKIP
00          NOP
00    SKIP: NOP
```

```
                        ; Unbedingte Spruenge Indirekt
E9                              JP      (HL)
DD E9                           JP      (IX)
FD E9                           JP      (IY)
                        ; Unbedingter Sprung Absolut
C3 84 05                        JP      0584H
                        ; Bedingte Spruenge Absolut
DA 84 05                        JP      C,0584H
FA 84 05                        JP      M,0584H
D2 84 05                        JP      NC,0584H
C2 84 05                        JP      NZ,0584H
F2 84 05                        JP      P,0584H
EA 84 05                        JP      PE,0584H
E2 84 05                        JP      PO,0584H
CA 84 05                        JP      Z,0584H
                        ; Bedingte Spruenge Relativ
38 FE                   L1:     JR      C,L1
30 FE                   L2:     JR      NC,L2
20 FE                   L3:     JR      NZ,L3
28 FE                   L4:     JR      Z,L4
                        ; Unbedingter Sprung Relativ
18 FE                   L5:     JR      L5

                        ; Das Register B wird um eins
                        ; verringert, der Sprung wird
                        ; ausgefuehrt bis das Register B
                        ; den Wert 0 besitzt
10 FE                   LP:     DJNZ    LP
```

Abb. 6.1.21 Sprungbefehle

Bei einem Sprung zurück:

```
LOOP:   NOP
        NOP
        JR LOOP
```

oder kodiert:

```
00    LOOP: NOP
00          NOP
18 FC       JR LOOP
```

Der Befehl DJNZ ist noch eine Delikatesse. Er decrementiert das Register B und springt in Abhängigkeit des Wertes.

Damit lassen sich elegant Schleifen aufbauen. Beispiel:

```
        LD B,5
LOOP:   NOP
        NOP
        DJNZ LOOP
```

Die Schleife wird fünf Mal durchlaufen.

```
                        ; Bedingter Unterprogrammaufruf
                        ; falls die Bedingung erfuellt ist
DC 84 05                        CALL    C,0584H
FC 84 05                        CALL    M,0584H
D4 84 05                        CALL    NC,0584H
C4 84 05                        CALL    NZ,0584H
F4 84 05                        CALL    P,0584H
EC 84 05                        CALL    PE,0584H
E4 84 05                        CALL    PO,0584H
CC 84 05                        CALL    Z,0584H
                        ; Unbedingter Unterprogrammaufruf
CD 84 05                        CALL    0584H



                        ; Restart auf Speicherzelle
                        ;l wie Unterprogrammaufruf
C7                              RST     00H
CF                              RST     08H
D7                              RST     10H
DF                              RST     18H
E7                              RST     20H
EF                              RST     28H
F7                              RST     30H
FF                              RST     38H

                        ; Ruecksprung aus einem Unterprogramm
C9                              RET
                        ; Bedingter Ruecksprung aus einem
                        ; Unterprogramm
D8                              RET     C
F8                              RET     M
D0                              RET     NC
C0                              RET     NZ
F0                              RET     P
E8                              RET     PE
E0                              RET     PO
C8                              RET     Z
                        ; Rueckkehr aus einem Interruptprogramm
                        ; das mit NMI aufgerufen wurde
ED 4D                           RETI
                        ; Rueckkehr aus einem Interruptprogramm
                        ; das mit INT aufgerufen wurde
ED 45                           RETN
```

Abb. 6.1.22 Unterprogramm-Befehle

12. Unterprogramm-Aufrufe

Abb. 6.1.22 zeigt alle Unterprogramm-Befehle. Dabei gibt es, sowohl bei den Aufrufen, als auch bei Rücksprüngen, bedingte Anweisungen.

```
                        ; Laden eines Registers von einem
                        ; IO-Geraet mit der Adresse, die
                        ; in dem Register C steht
ED 78                           IN      A,(C)
ED 40                           IN      B,(C)
ED 48                           IN      C,(C)
ED 50                           IN      D,(C)
ED 58                           IN      E,(C)
ED 60                           IN      H,(C)
ED 68                           IN      L,(C)


                        ; Laden des Akkumulators mit einem Wert
                        ; des IO-Geraets auf der angegebenen
                        ; Adresse
DB 20                           IN      A,(20H)


                        ; Ausgabe des Inhalt von Akku A an das
                        ; Port mit der Adresse die in C steht
                        ; (C).
ED 79                           OUT     (C),A
ED 41                           OUT     (C),B
ED 49                           OUT     (C),C
ED 51                           OUT     (C),D
ED 59                           OUT     (C),E
ED 61                           OUT     (C),H
ED 69                           OUT     (C),L


                        ; Der Port mit der angegebenen Adresse
                        ; wird mit dem Inhalt des Akkus geladen
D3 20                           OUT     (20H),A
```

Abb. 6.1.23 IO-Befehle

Die RST-Befehle sind ebenfalls Unterprogrammaufrufe, jedoch können sie nur auf 8 vorbestimmte Adressen ausgeführt werden und sind als Hilfsmittel bei der Interrupt-Verarbeitung verwendbar. Bei einem Interrupt wird ebenfalls nur ein Unterprogrammaufruf durchgeführt.

13. I/O-Befehle

Abb. 6.1.23 zeigt alle Standard-IO-Befehle. Besondere Bedeutung erlangen die IO-Blockbefehle aus *Abb. 6.1.24*. Mit diesen Befehlen ist es möglich, einen ganzen Speicherblock (maximal 256 Bytes lang) auszulagern oder zu laden.

```
                        ; Laden der Speicherstelle (HL) mit einem
ED AA                   ; Wert des Input-Ports mit Adresse (C)
                        ; HL, B decrementieren
                                IND
                        ; Laden der Speicherstelle (HL) mit einem
                        ; Wert des Input-Ports mit Adresse (C)
ED BA                   ; HL, B decrementieren, Wiederholung
                        ; bis B=0
                                INDR
                        ; Laden der Speicherstelle (HL) mit einem
ED A2                   ; Wert des Input-Ports mit Adresse (C)
                        ; HL Incrementieren, B decrementieren
                                INI
                        ; Laden der Speicherstelle (HL) mit einem
                        ; Wert des Input-Ports mit Adresse (C)
ED B2                   ; HL Incrementieren, B decrementieren
                        ; Wiederholung bis B=0
                                INIR


                        ; Der Ausgabeport mit Adresse (C) wird von
                        ; der Adresse (HL) geladen
                        ; HL und B werden decrementiert
ED AB                           OUTD
                        ; Der Ausgabeport mit Adresse (C) wird von
                        ; der Adresse (HL) geladen
                        ; HL und B werden decrementiert
                        ; Wiederholung bis B=0 ist
ED BB                           OTDR
                        ; Der Ausgabeport mit Adresse (C) wird von
                        ; der Adresse (HL) geladen
                        ; HL wird incrementiert
                        ; und B wird decrementiert
ED A3                           OUTI
                        ; Der Ausgabeport mit Adresse (C) wird von
                        ; der Adresse (HL) geladen
                        ; HL wird incrementiert
                        ; und B wird decrementiert
                        ; Wiederholung bis B=0 ist
ED B3                           OTIR
```

Abb. 6.1.24 Block-IO-Befehle

14. Interrupt-Verarbeitung

Eine sehr wichtige Fähigkeit bei Computern ist die Interrupt-Behandlung. Ein Interrupt ist ein durch ein Hardwaresignal ausgelöster Unterprogrammaufruf. Damit kann durch ein Ereignis eine Aktion im Rechner ausgelöst werden. Gäbe es keine Interruptverarbeitung, so müßte im laufenden Programm ständig abgefragt werden, ob dieses Ereignis vorliegt oder nicht. Wenn ein solches Ereignis zu jeder Zeit kommen kann, wird das Hauptprogramm durch die dauernden Abfragen sehr langsam.

Ein Interrupt wird immer dann verwendet, wenn das Ereignis es erfordert, sofort bearbeitet zu werden. Beispielsweise sollen zu einem beliebigen Zeitpunkt Daten über eine serielle Leitung von außen ankommen können und sie müssen im Speicher abgelegt werden. Der Computer soll aber normalerweise etwas anderes tun, zum Beispiel irgendwelche Berechnungen durchführen. Nun muß ohne Interruptverarbeitung während der Berechnung dauernd gefragt werden, ob etwas über die serielle Leitung kommt. Wenn nicht, kann die Berechnung weitergeführt werden, wenn ja, so muß das ankommende Datum im Speicher abgelegt werden. Bei einer Interrupt-Methode können die Berechnungen direkt, ohne zusätzliches Abfragen durchgeführt werden. Über eine spezielle Leitung, der Interrupt-Leitung, gelangt ein Signal von dem seriellen Interface direkt zur CPU. Trifft ein Zeichen im seriellen Interface ein, so wird über diese Leitung bei der CPU ein Interrupt ausgelöst. Dadurch wird das Interruptprogramm gestartet. Wo es liegt, kann meist vom seriellen Interface selbst angegeben werden. Dieses Interruptprogramm ist praktisch ein normales Unterprogramm. Nachdem es fertig bearbeitet wurde, wird wieder das Hauptprogramm ausgeführt. Ein paar wesentliche Gesichtspunkte sind beim Interruptprogramm zu beachten. Da es an beliebiger Stelle bei der Abarbeitung des Hauptprogrammes aufgerufen werden kann, muß dafür gesorgt werden, daß alle, im Interruptprogramm verwendeten Register, am Anfang gerettet werden und nach der Ausführung des Interruptprogramms wieder zurückgespeichert werden müssen. Ansonsten wären Zwischenergebnisse, die im Hauptprogramm in Registern abgelegt wurden, wie auch Status-Zustände, verändert, wenn der Interrupt beendet ist und dies darf natürlich nicht geschehen.

Der Z80 besitzt zwei Interrupt-Eingänge. Der -NMI-Eingang und der -INT-Eingang. Erscheint beim -NMI-Eingang eine Flanke von 1 nach 0, so wird ein Interrupt ausgeführt. Dabei steht das Interruptprogramm auf Adresse 66h. Dort kann zum Beispiel auch ein Sprung auf das eigentliche Interruptprogramm stehen. Ein Interruptprogramm wird mit dem Befehl RETN abgeschlossen. Daran können Peripherie-Bausteine das Ende erkennen. Dies ist aber nur für Spezialfälle interessant. Das Interruptprogramm wird normalerweise mit einer Sequenz

```
PUSH AF      . . . . . .
PUSH BC
PUSH DE
PUSH HL
EX AF,AF'
PUSH AF
EXX
PUSH BC
PUSH DE
PUSH HL
PUSH IX
PUSH IY
```

eingeleitet und endet mit einer Sequenz

```
POP IY
POB IX , . . . . . POP AF
```

Die Register die im Interruptprogramm nicht gebraucht werden, müssen natürlich auch nicht gerettet werden.

Eine andere Möglichkeit gibt es mit dem Befehl -INT. Er ist universeller in den Möglichkeiten. Er kann gesperrt werden (disable). Das bedeutet, daß die CPU dann nicht auf den Interrupt reagiert. Manchmal gibt es Fälle, in denen ein Programmstück in der Ausführung auch nicht durch einen Interrupt unterbrochen werden darf, zum Beispiel in einer zeitkritischen Routine. Der Befehl DI und EI steuert das beim Z80. Mit EI wird der -INT-Eingang freigegeben und mit DI wird er gesperrt. Hier reagiert die CPU nicht auf eine Flanke, sondern auf den statischen Pegel 0. Ist -INT auf 0 und ist er freigegeben, so wird der Interrupt ausgeführt. Dabei wird er sofort gesperrt, denn sonst würde ja wegen des statischen Verhaltens sofort wieder ein Interrupt erfolgen. Dann kann das Interruptprogramm ausgeführt werden. Das Interrupt-Signal muß innerhalb dieser Zeit wieder entfernt werden, z. B. kann es per Software zurückgesetzt werden. Vor Beenden des Interruptprogramms wird der Befehl EI gegeben und dann RETI. Mit EI wird erst nach der Ausführung des nächsten Befehls die Freigabe des Interrupts bewirkt und somit erst nach der Ausführung des Rücksprungbefehls durch RETI. Beim -INT-Eingang gibt es verschiedene Möglichkeiten, wie die CPU darauf reagiert. Dazu gibt es die Befehle IM 0, IM 1, IM 2, die einen Betriebsmode programmieren. Nach dem RESET ist IM 0 wirksam.

Im Mode 0, der indentisch mit dem Verhalten der 8080-CPU ist, kann ein Befehl per Hardware bei der Interruptannahme auf den Bus gelegt werden. Dieser Befehl wird dann ausgeführt. Damit der Interrupt wie ein Unterprogrammaufruf wirken kann, wird als Befehl ein Unterprogrammsprung verwendet. Dazu kann der CALL-Befehl verwendet werden, der allerdings drei Bytes Opcode benötigt. Einfacher ist die Verwendung der RST-Befehle, die dafür auch gedacht sind. Mit ihnen kann ein Unterprogrammaufruf in eine von 8 verschiedenen festen Zellen erfolgen. Dort kann z. B. ein Sprungbefehl stehen. Die Interrupt-Quittung erfolgt beim Z80 durch gleichzeitiges Aktivieren von -M 1 und -IORQ, diese beiden Signale können nur in diesem Fall beide auf 0 liegen. Die Bedeutung ist dabei tatsächlich Holen eines Operationscodes, aber nicht vom Speicher, den das wäre -MREQ, sondern von einem Peripheriegerät, was durch -IORQ angezeigt wird.

Der Mode 1 ist einfacher, hier erfolgt ein Aufruf auf Zelle 38h, unabhängig davon, was am Bus geschieht.

Der Mode 2 hingegen ist am flexibelsten. Dort kann ein indirekter Sprung an eine beliebige Stelle des Speichers ausgeführt werden. Dazu gibt es das Register I, das in 8 Bit die höherwertige Adresse einer Speicherzelle trägt. Bei der Interrupt-Quittung durch -M 1 und -IORQ wird eine weitere 8-Bit-Adresse auf den Datenbus gegeben, aber Bit 0 muß auf 0 liegen. Damit ergibt sich eine 16-Bit-Gesamtadresse. Aus der Zelle, die dadurch angesprochen wird, wird der niederwertige Teil einer Adresse geholt und aus der nächst höheren (Bit 0=1) wird die höherwertige Adresse geholt. Diese neue Adresse wird dann als Interruptprogrammstartadresse verwendet.

Damit ist es möglich, an einer Stelle im Speicher eine Adreßtabelle aufzubauen, in der die Adressen von maximal 128 Interruptroutinen stehen.

Viele Z80-Bausteine können die Interruptsteuerung selbständig durchführen.

6.2 Assembler

In diesem Abschnitt wird etwas über die Funktionsweise von Assemblern sowie deren Fähigkeiten gesagt.

Die Aufgabe eines Assemblers ist es, ein mit Mnemonics geschriebenes Programm in den Maschinencode zu übersetzen. Bei Sprüngen wird zum Beispiel noch ein Sprungziel angegeben. Dieses Sprungziel kann in symbolischer Form, z. B. als Name gegeben werden. Diese Namen müssen vom Assembler durch die Maschinenadresse ersetzt werden.

Wie bei der Übersetzung eines Programms vorgegangen wird, sei im folgenden, anhand des Programms aus *Abb. 6.2.1*, gezeigt.
Aufgabe ist es, das Programm von Hand zu übersetzen. Dazu wird in mehreren Schritten vorgegangen. Als erstes werden alle Operationscodes erzeugt. Dabei können Sprungadressen, die noch nicht bekannt sind, auch noch nicht eingesetzt werden. Ein Listing nach *Abb. 6.2.2* entsteht dabei. Die Teile mit ?? sind noch unbekannt. Nach diesem ersten Durchlauf (engl. PASS), muß ein erneuter Durchlauf erfolgen, in dem die noch fehlenden Adreßteile eingesetzt werden. Danach ergibt sich ein Listing nach *Abb. 6.2.3*. Ein Assembler

```
;************************************
;* Beispiel einer Handuebersetzung *
;************************************


start:  ld sp,13ffh      ;stack definieren
        call init        ;forwaerts referenz
        call main        ;forwaerts referenz
        jp start         ;rueckwaerts referenz

init:   ld a,5
        out (30h),a      ;port init
        ret

main:   ld b,5           ;schleifenzaehler
        ld a,1           ;ausgabewert
loop:   out (31h),a      ;dahin ausgeben
        djnz loop        ;rueckwaertsref
        in a,(31h)
        or a             ;test ob null
        jr z,skip        ;forwaerts referenz
        ld a,6
        out (31h),a
        jp over          ;forwaerts referenz
skip:   ld a,7
        out (31h),a
over:   ld a,8
        out (31h),a
        ret
```

Abb. 6.2.1 Beispiel einer Handübersetzung

```
                        ;***********************************
                        ;* Beispiel einer Handuebersetzung *
                        ;***********************************
0000    31 FF 13        start: ld sp,13ffh      ;stack definieren
0003    CD ?? ??               call init        ;forwaerts referenz
0006    CD ?? ??               call main        ;forwaerts referenz
0009    C3 00 00               jp start         ;rueckwaerts referenz

000C    3E 05           init:  ld a,5
000E    D3 30                  out (30h),a      ;port init
0010    C9                     ret

0011    06 05           main:  ld b,5           ;schleifenzaehler
0013    3E 01                  ld a,1           ;ausgabewert
0015    D3 31           loop:  out (31h),a      ;dahin ausgeben
0017    10 FC                  djnz loop        ;rueckwaertsref
0019    DB 31                  in a,(31h)
001B    B7                     or a             ;test ob null
001C    28 ??                  jr z,skip        ;forwaerts referenz
001E    3E 06                  ld a,6
0020    D3 31                  out (31h),a
0022    C3 ?? ??               jp over          ;forwaerts referenz
0025    3E 07           skip:  ld a,7
0027    D3 31                  out (31h),a
0029    3E 08           over:  ld a,8
002B    D3 31                  out (31h),a
002D    C9                     ret
```

Abb. 6.2.2 Pass 1

```
                        ;***********************************
                        ;* Beispiel einer Handuebersetzung *
                        ;***********************************

0000    31 FF 13        start: ld sp,13ffh      ;stack definieren
0003    CD 0C 00               call init        ;forwaerts referenz
0006    CD 11 00               call main        ;forwaerts referenz
0009    C3 00 00               jp start         ;rueckwaerts referenz

000C    3E 05           init:  ld a,5
000E    D3 30                  out (30h),a      ;port init
0010    C9                     ret

0011    06 05           main:  ld b,5           ;schleifenzaehler
0013    3E 01                  ld a,1           ;ausgabewert
0015    D3 31           loop:  out (31h),a      ;dahin ausgeben
0017    10 FC                  djnz loop        ;rueckwaertsref
0019    DB 31                  in a,(31h)
001B    B7                     or a             ;test ob null
001C    28 07                  jr z,skip        ;forwaerts referenz
001E    3E 06                  ld a,6
0020    D3 31                  out (31h),a
0022    C3 29 00               jp over          ;forwaerts referenz
0025    3E 07           skip:  ld a,7
0027    D3 31                  out (31h),a
0029    3E 08           over:  ld a,8
002B    D3 31                  out (31h),a
002D    C9                     ret
```

Abb. 6.2.3 Pass 2

```
MACRO-80 3.43   27-Jul-81       PAGE    1

                                .z80

                                ;***********************************
                                ;* Beispiel einer Handuebersetzung *
                                ;***********************************


0000'   31 13FF                 start:  ld sp,13ffh     ;stack definieren
0003'   CD 000C'                        call init       ;forwaerts referenz
0006'   CD 0011'                        call main       ;forwaerts referenz
0009'   C3 0000'                        jp start        ;rueckwaerts referenz

000C'   3E 05                   init:   ld a,5
000E'   D3 30                           out (30h),a     ;port init
0010'   C9                             ret

0011'   06 05                   main:   ld b,5          ;schleifenzaehler
0013'   3E 01                           ld a,1          ;ausgabewert
0015'   D3 31                   loop:   out (31h),a     ;dahin ausgeben
0017'   10 FC                           djnz loop       ;rueckwaertsref
0019'   DB 31                           in a,(31h)
001B'   B7                              or a            ;test ob null
001C'   28 07                           jr z,skip       ;forwaerts referenz
001E'   3E 06                           ld a,6
0020'   D3 31                           out (31h),a
0022'   C3 0029'                       jp over         ;forwaerts referenz
0025'   3E 07                   skip:   ld a,7
0027'   D3 31                           out (31h),a
0029'   3E 08                   over:   ld a,8
002B'   D3 31                           out (31h),a
002D'   C9                              ret

                                        end


MACRO-80 3.43   27-Jul-81       PAGE    S

Macros:

Symbols:
000C'   INIT            0015'   LOOP            0011'   MAIN
0029'   OVER            0025'   SKIP            0000'   START


No Fatal error(s)
```

Abb. 6.2.4 Assemblerausgabe

geht dabei genauso vor. Er muß den Quelltext zweimal durchlaufen, bevor er den Objektcode ablegen kann. *Abb. 6.2.4* zeigt die Ausgabe des Assemblers. Alle Adressen sind mit dem Zeichen „'" gekennzeichnet. Durch addieren einer Konstanten auf diesen Adreßteil

```
                        ;************************************
                        ;* Steueranweisungen bei Assembler  *
                        ;* ORG ANWEISUNG                    *
                        ;************************************

                                org 100h        ;start adresse
0100    00                      nop             ;wird auf 100h
0101    78                      ld a,b          ;abgelegt usw.
                                org 200h        ;neuer anfang
0200    4F                      ld c,a
0201    3C                      inc a
```

Abb. 6.2.5 ORG-Anweisung

```
                        ;************************************
                        ;* Operanden bei Assemblern         *
                        ;************************************

0000    3E 41                   ld a,'A'
0002    3E 50                   ld a,78+2
0004    3E 3D                   ld a,'A'-4
0006    3E 93                   ld a,10010011b
0008    3E 17                   ld a,23d
000A    3E 17                   ld a,23
000C    3E 3C                   ld a,74o
000E    3E 74                   ld a,74h
0010    3E BF                   ld a,10110101b or 00001111b
0012    3E B0                   ld a,10111010b and 11110000b
0014    3E 67                   ld a,10101011b xor 11001100b
0016    3E 09                   ld a,'9' and 0fh
```

Abb. 6.2.6 Operandenangabe beim Assembler

kann das Programm verschoben werden und ist dann auf einer anderen Start-Adresse lauffähig. Die Adreßteile werden von diesem Assembler in der Form MSB,LSB ausgegeben und nicht in der Form, wie sie später im Speicher stehen, nämlich LSB, MSB. Am Ende der Übersetzung wird die sogenannte Symboltabelle ausgegeben, dies ist ein Verzeichnis der verwendeten Marken (engl. Labels) und deren zugehöriger Wert.

Das Assemblerprogramm beginnt mit dem Befehl „.z80". Er bewirkt, daß der Assembler die Z80-Befehle versteht, denn er kann auch 8080-Befehle verstehen, die etwas anders aussehen, obwohl sie eine Untermenge der Z80-Befehle sind. Das Programm endet mit dem Befehl „end", der dem Assembler sagt, daß hier das Programm beendet ist. Diese Befehle, die keine Befehle des Z80 sind, nennt man Pseudobefehle. Sie versteht nur der Assembler und sie sind zu dessen Steuerung gedacht.

Der Assembler besitzt eine Vielzahl von Pseudobefehlen, um die Programmierung zu erleichtern. Ein paar davon werden wir jetzt kennenlernen. *Abb. 6.2.5* zeigt die Anwendung des ORG-Befehls. Damit kann der Adreßzähler auf einen Wert gesetzt werden. Soll der Code ab Adresse 100h beginnen, so wird ein ORG 100h vorangestellt. Der Adreßzähler wird benötigt, um Sprungmarken berechnen zu können. Der Assembler hat auch

```
                                ;************************************
                                ;* Steueranweisungen bei Assembler  *
                                ;* EQU ANWEISUNG                    *
                                ;************************************

0020                            blank   equ     ' '     ;definitionen
000A                            zehn    equ     10      ;zur klaren
00FE                            port    equ     0feh    ;programmgestaltung
1234                            konst   equ     1234h

0000    3E 20                           ld a,blank      ;damit auch leicht
0002    D6 0A                           sub zehn        ;aenderbar
0004    D3 FE                           out (port),a
0006    21 1234                         ld hl,konst
```

Abb. 6.2.7 EQU-Anweisung

```
                                ;************************************
                                ;* Steueranweisungen bei Assembler  *
                                ;* SPEICHERBELGUNGSANWEISUNGEN      *
                                ;************************************

0000                            tabelle:
0000    01 02 03 04                     defb 1,2,3,4
0004    61 62 63 64                     defb 'a','b','c','d'

0008                            adressen:
0008    1000                            defw 1000h
000A    2010                            defw 2010h

000C                            texte:
000C    61 6C 70 68                     defm 'alpha TEXT'
0010    61 20 54 45
0014    58 54
0016    77 69 72 64                     defm 'wird abgelegt'
001A    20 61 62 67
001E    65 6C 65 67
0022    74

                                ; bei adressen z.B. Anwendung interrupt
                                ; adresstabelle

                                        org 1000h       ;dort sei start
1000    0524                            defw int1prg
1002    13FF                            defw int2prg
1004    1400                            defw int3prg

0524                            int1prg equ 524h        ;z.B. dort int1
13FF                            int2prg equ 13ffh       ;int2
1400                            int3prg equ 1400h       ;int3
```

Abb. 6.2.8 Speicherbelegungs-Anweisung

die Möglichkeit, in verschiedenen Zahlensystemen zu arbeiten. Es besteht auch die Möglichkeit, unterschiedliche Verknüpfungen bei Operanden durchzuführen, so gibt es die vier Grundrechenarten und logische Verknüpfungen. In *Abb. 6.2.6* sind ein paar Operationen aufgelistet.

Marken als symbolische Operanden haben wir bei Sprungzielen schon kennengelernt. Symbolische Operanden können aber auch direkt definiert werden. *Abb. 6.2.7* zeigt wie. Durch die Verwendung von Namen wird ein Programm übersichtlicher und die Bedeutung von Werten kann veranschaulicht werden.

Neben Befehlen müssen aber auch Daten im Speicher abgelegt werden können. Dazu gibt es eine Reihe von Pseudobefehlen, die das leisten. In *Abb. 6.2.8* sind sie dargestellt. Dabei werden drei Gruppen unterschieden. Befehle zur Ablage von Bytes (DEFB), zur Ablage von 16-Bit-Werten (DEFW) und zur Ablage von Texten (DEFM). Mit DEFB können

```
                                ;************************************
                                ;* Steueranweisungen bei Assembler  *
                                ;* MAKROBEFEHLE 1                   *
                                ;************************************

                                save    macro
                                        push af
                                        push bc
                                        push de
                                        push hl
                                        endm

                                restore macro
                                        pop hl
                                        pop de
                                        pop bc
                                        pop af
                                        endm

                                        org 1200h

1200                            uprg:   save
1200    F5              +               push af
1201    C5              +               push bc
1202    D5              +               push de
1203    E5              +               push hl
1204    21 1000                         ld hl,1000h
1207    7E                              ld a,(hl)
1208    D6 03                           sub 3
120A    77                              ld (hl),a
                                        restore
120B    E1              +               pop hl
120C    D1              +               pop de
120D    C1              +               pop bc
120E    F1              +               pop af
120F    C9                              ret
```

Abb. 6.2.9
Makro-Anweisungen

aber auch Texte abgelegt werden. Ferner gibt es noch den nicht dargestellten Befehl DEFS. Mit ihm kann ein Speicherblock reserviert werden. Der Befehl DEFS 200 hält 200 Speicherzellen frei. Dabei wird einfach der Adreßzähler des Assemblers um 200 erhöht.

Eine Besonderheit, die nicht alle Assembler besitzen, ist die Behandlung von MAKROS. Wird eine Sequenz von Befehlen sehr oft benötigt, so verwendet man einen Unterprogrammaufruf. Doch es gibt eine ähnliche Aufgabenstellung beim Schreiben eines Programms, wo ein Unterprogrammaufruf nicht in Frage kommt. *Abb. 6.2.9* zeigt einen solchen Fall. Alle Register (nur der 8080-Satz AF,BC,DE und HL) sollen öfters auf den Stack gespeichert und dann wieder zurückgeholt werden. Um sich die Schreibarbeit zu sparen und auch vor Fehlern sicherer zu sein, kann ein Makro definiert werden. Danach gibt es in unserem Beispiel zwei neue Befehle. Den Befehl SAVE und den Befehl RESTORE. Sie können, wie die anderen Mnemonics, von da an im Assemblerprogramm verwendet werden. Bei der Übersetzung wird aber der Originalcode, der in der Makrodefinition steht, übernommen und in das fertige Programmlisting einkopiert. Alle mit + gekennzeichneten Stellen sind auf diese Weise entstanden. Sie wurden vom Assembler erzeugt.

Ein anderes Beispiel für die Makrobearbeitung ist in *Abb. 6.2.10* gezeigt. Es wird dort ein Makro mit dem Namen SUBHL erzeugt. Dieser Makro besitzt aber einen Parameter, der den Namen „WERT" trägt. Bei der Verwendung des Makros kann an Stelle des Para-

```
                              ;************************************
                              ;* Steueranweisungen bei Assembler  *
                              ;* MAKROBEFEHLE 2                   *
                              ;************************************

                              subhl   macro wert
                                      push de
                                      ld de,wert
                                      xor a
                                      sbc hl,de
                                      pop de
                                      endm


                                      org 1200h
1200    2A 1000                       ld hl,(1000h)   ;laden mit inhalt 1000h
                                      subhl 395       ;subtrahieren
1203    D5              +             push de
1204    11 018B         +             ld de,395
1207    AF              +             xor a
1208    ED 52           +             sbc hl,de
120A    D1              +             pop de
120B    22 1000                       ld (1000h),hl   ;ablegen
                                      subhl 412       ;nochmals subtr
120E    D5              +             push de
120F    11 019C         +             ld de,412
1212    AF              +             xor a
1213    ED 52           +             sbc hl,de
1215    D1              +             pop de
1216    22 1002                       ld (1002h),hl   ;ablegen
```

Abb. 6.2.10 Weiteres Beispiel Makro-Anweisungen

meters eine beliebige Konstante geschrieben werden. An allen Stellen, an denen in der Makrodefinition der Parameter „WERT" verwendet wird, wird dann die Konstante eingesetzt.

Weitere Pseudobefehle im Assembler gestatten die bedingte Übersetzung, bei denen in Abhängigkeit von Parametern z. B. ein Programmteil übersetzt oder nicht übersetzt wird, doch würde die ausführliche Behandlung an dieser Stelle den Inhalt des Buches sprengen.

Am Schluß dieses Abschnitts sei noch der Begriff EDITOR kurz erwähnt, da er auch immer im Zusammenhang mit Assemblern genannt wird. Mit dem Editor ist es möglich, die Quellprogramme in den Computer einzugeben und zu verbessern. Danach steht der Quelltext der Assemblierung zur Verfügung. Liegt ein Fehler im Programm vor, z. B. ein falsch geschriebener Befehl oder eine nirgendwo vorkommende Marke, dies zeigt der Assembler durch eine Fehlermeldung an, so muß der Quelltext neu editiert werden, um dann anschließend erneut übersetzt zu werden.

6.3 Strukturierte Programmierung

Der Begriff der strukturierten Programmierung ist heute bei den Programmierern ein Schlagwort. So wollen auch wir ein bißchen darüber hören, um die Vorteile schätzen zu lernen. Die strukturierte Programmierung ist eine Vorgehensweise, die zu leicht änderbaren und klaren Programmen führt.

In der Anfangszeit der Programmierung wurde großer Wert auf besonders trickreiche Programmierung gelegt und die Programmierer haben viel Zeit darauf verwandt, Programme kompakt und optimiert zu gestalten. Genausoviel Zeit oder noch viel mehr haben sie jedoch bei der Fehlersuche gebraucht. Heute ist man davon ganz abgekommen. Die Programme sollen möglichst gut lesbar und klar verständlich im Aufbau sein. Es wurde ein Satz von Regeln entworfen, nach denen man beim Programmieren vorgehen muß, um eine klare Prgrammstruktur zu erhalten. Diese Regeln wurden ursprünglich für höhere Programmiersprachen entwickelt, jedoch lassen sie sich genauso auf Assemblerprogramme anwenden. Diese Regeln werden im folgenden behandelt.

Die einfachste Programmstruktur ist ein lineares Programm. Dort folgt eine Anweisung auf die nächste, ohne daß irgendwelche Sprünge ausgeführt werden. *Abb. 6.3.1* zeigt ein Flußdiagramm dazu. Bei einem Flußdiagramm gibt es ein Start und ein Endesymbol. Dies markiert einen logischen Abschnitt. In dem rechteckigen Kasten werden die Anweisungen hingeschrieben.

Mit linearer Struktur allein können aber keine sinnvollen Programme geschrieben werden. Eine Entscheidungsmöglichkeit wird gebraucht. *Abb. 6.3.2* zeigt die erste Form einer strukturierten Entscheidung. Nach Abfrage einer Bedingung wird entweder der Ja-Teil oder der Nein-Teil ausgeführt. Danach wird mit einem gemeinsamen Programmabschnitt fortgefahren. In dem Flußdiagramm ist die Entscheidung als Raute abgebildet. Als Beispiel eines Assemblerprogramms ist das Listing in *Abb. 6.3.3* gezeigt. Nach der Entscheidung folgt ein Ja-Teil und ein Nein-Teil. Nach dem Ja-Teil erfolgt im Assembler-

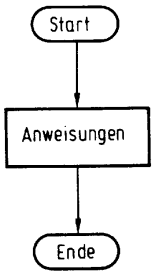


Abb. 6.3.1 Lineare Sequenz

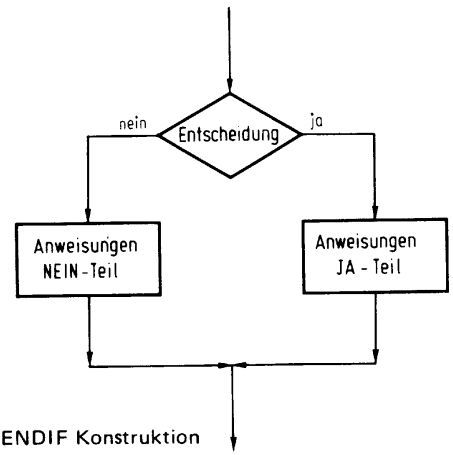


Abb. 6.3.2 Flußdiagramm IF ELSE ENDIF Konstruktion

```
                        ; strukturierte Programmierung
                        ; Befehlsanordnungen

                        ; IF  ....  ELSE ....  ENDIF

                        ; if nonzero
0000'   28 05           jr z,elsepart       ;ueberspringen wenn bedingung
0002'   00               nop                ;nicht erfuellt
0003'   00               nop                ;--- beliebige instruktionen
0004'   00               nop
                        ; else
0005'   18 03           jr endif            ;uebersprung else teil
0007'                   elsepart:           ;einsprung
0007'   00               nop
0008'   00               nop                ;--- beliebige instruktionen
0009'   00               nop
                        ; endif
000A'                   endif:
000A'   00              nop
000B'   00              nop                 ;--- folge befehle
000C'   00              nop
```

Abb. 6.3.3 Assemblercodierung IF ELSE ENDIF

programm ein unbedingter Sprung hinter den Nein-Teil. Die Bedingung ist in dem Fall, daß das Zero-Flag nicht gesetzt ist.

Eine weitere Form einer Bedingungsstruktur zeigt *Abb. 6.3.4*, hier fehlt einfach der Nein-Teil. Im Fall Ja wird eine Anweisungssequenz ausgeführt, sonst nichts. *Abb. 6.3.5* zeigt die Realisierung im Assembler.

Nun gibt es aber auch andere Sprungstrukturen. *Abb. 6.3.6* zeigt die sogenannten WHILE-Bedingung. Es wird zuerst eine Abfrage durchgeführt. Ist sie nicht erfüllt, wird abgebrochen, ist sie erfüllt, so wird eine Sequenz von Anweisungen durchgeführt und dann erneut die Anweisung ausgeführt. Diese Form wird auch als Schleife bezeichnet. *Abb. 6.3.7* zeigt, wie sie im Assembler verwirklicht werden kann.

```
                        ; IF ... ENDIF

                        ;if non zero
0000'   28 03           jr z,skip           ;ueberspringen falls zero
0002'   00               nop
0003'   00               nop                ;--- befehlssequenz
0004'   00               nop
                        ;endif
0005'                   skip:               ;uebersprungmarke
0005'   00             nop
0006'   00             nop                 ;--- folge befehle
0007'   00             nop
```

Abb. 6.3.5 Assemblercodierung IF ENDIF

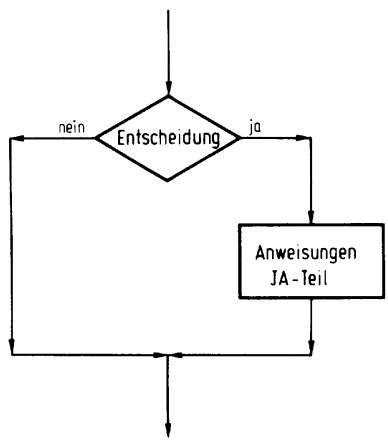


Abb. 6.3.4 Flußdiagramm IF ENDIF Konstruktion

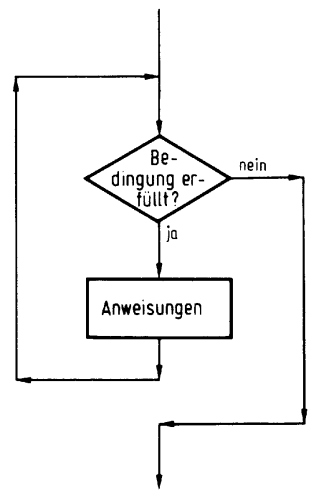


Abb. 6.3.6 Flußdiagramm WHILE ENDWHILE Konstruktion

```
                        ; WHILE ... ENDWHILE

                        ; while zero
0000'                   repw:               ;ruecksprung
0000'   20 05           jr nz,skipw         ;nicht zero dann stop
0002'   00               nop
0003'   00               nop                ;--- befehlssequenz
0004'   00               nop
                        ; endwhile
0005'   18 F9           jr repw             ;immer zurueck hier
0007'                   skipw:              ;ende teil
0007'   00              nop
0008'   00              nop                 ;--- folge befehle
0009'   00              nop
```

Abb. 6.3.7 Assemblercodierung WHILE ENDWHILE

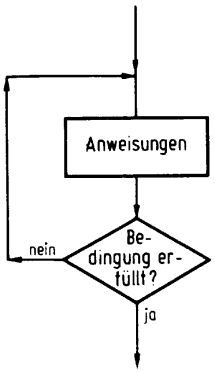


Abb. 6.3.8 Flußdiagramm REPEAT UNTIL Konstruktion

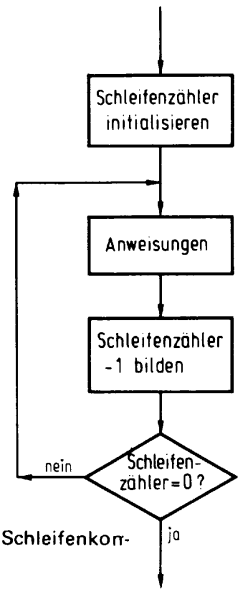


Abb. 6.3.10 Flußdiagramm der Schleifenkonstruktion

```
                        ; REPEAT ... UNTIL

                        ; repeat
0000'                   rep:                ;einsprung fuer rueckkehr
0000'   00               nop
0001'   00               nop                ;--- befehlssequenz
0002'   00               nop
                        ; until carry
0003'   30 FB           jr nc,rep           ;nicht erfuellt dann zurueck
0005'   00             nop
0006'   00             nop                 ;--- folge befehle
0007'   00             nop
```

Abb. 6.3.9 Assemblercodierung REPEAT UNTIL

```
                            org 0


                        ; DO register,anzahl ... ENDDO

                        ; DO c,5
0000'   0E 05           ld c,5
0002'                   lpp:                ;schleifenruecksprung
0002'   00               nop
0003'   00               nop                ;--- befehlssequenz
0004'   00               nop
                        ; ENDDO
0005'   0D              dec c
0006'   20 FA          jr nz,lpp           ;bis c=0 wiederholen
0008'   00             nop
0009'   00             nop                 ;--- folge befehle
000A'   00             nop
```

Abb. 6.3.11 Assemblercodierung der DO-Schleife mit dem C-Register

In *Abb. 6.3.8* ist eine ähnliche Form, die als REPEAT UNTIL bekannt ist, abgebildet. Es wird zunächst eine Anweisungssequenz ausgeführt und dann die Bedingung abgeprüft. Ist die Bedingung erfüllt, so wird abgebrochen, ist sie nicht erfüllt, so wird erneut die Anweisungssequenz durchgeführt. *Abb. 6.3.9* zeigt das Assemblerprogramm. Der Unterschied zwischen WHILE und REPEAT besteht darin, das bei REPEAT die Anweisungssequenz mindestens einmal durchlaufen wird und bei WHILE der Abbruch erfolgen kann, ohne daß die Anweisungssequenz auch nur einmal durchlaufen wurde.

Eine oft benötigte Schleifenstruktur ist die, bei der ein Zähler, der sogenannte Schleifenzähler mit auftritt. *Abb. 6.3.10* zeigt das Flußdiagramm. Dabei ist die Struktur im Prinzip identisch mit der Form REPEAT UNTIL. *Abb. 6.3.11* zeigt das Assemblerprogramm mit einem 8-Bit-Zähler, *Abb. 6.3.12* zeigt die Realisierung, wenn das Register B verwendet wird und in *Abb. 6.3.13* ist die Lösung mit einem 16-Bit-Zähler beschrieben. Häufig wird auch die Konstruktion einer Endlosschleife mit mehreren Abbruchkriterien gebraucht. *Abb. 6.3.14* zeigt die Lösung. Die Abfragen sind diesmal inmitten der Programmabschnitte. In *Abb. 6.3.15* ist das dazugehörige Assemblerprogramm beschrieben.

```
                        ; DO b,6
0000'   06 06           ld b,6
0002'                   dolp1:          ;schleifenruecksprung
0002'   00               nop
0003'   00               nop
0004'   00               nop
                        ; ENDDO
0005'   10 FB           djnz dolp1      ;sonderfall direktbefehl
0007'   00              nop
0008'   00              nop
0009'   00              nop

                                end
```

Abb. 6.3.12 Assemblercodierung der DO-Schleife mit dem B-Register

```
                        ; DO registerpaar,startwert ... ENDDO

                        ; do de,1000
0000'   11 03E8         ld de,1000
0003'                   dolp:           ;ruecksprungmarke
0003'   00               nop
0004'   00               nop            ;--- befehlssequenz
0005'   00               nop
                        ; enddo
0006'   1B              dec de
0007'   7B              ld a,e
0008'   B2              or d            ;test ob 0
0009'   20 F8           jr nz,dolp      ;nein dann nochmals
000B'   00              nop
000C'   00              nop             ;--- befehls folge
000D'   00              nop
```

Abb. 6.3.13 Assemblercodierung der DO-Schleife mit dem DE-Register

Diese einzelnen Module können zu einem gesamten Programm geformt werden. Dabei kann anstelle des Begriffs Anweisung wieder irgendeine der Formen stehen. Sie können damit beliebig verschachtelt werden. Beispiel:

```
loop
 Anweisung
 if zero
  Anweisung
 else
  Anweisung
  repeat
   Anweisung
  until no carry
  Anweisung
 endif
 exitif carry
 Anweisung
endloop
Anweisung
```

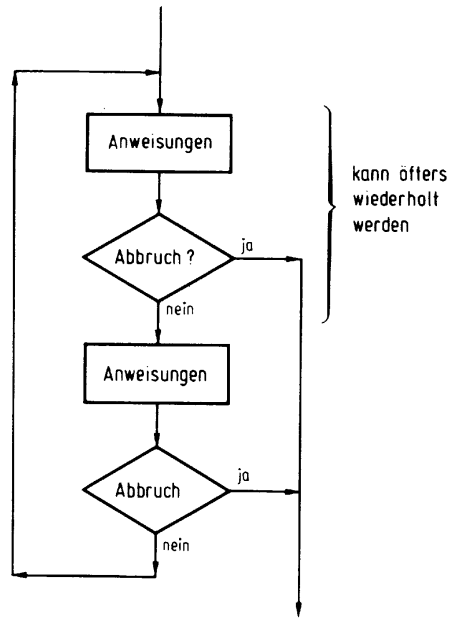


Abb. 6.3.14 Flußdiagramm mit LOOP ENDLOOP, Konstruktion

```
                        ; LOOP , EXIT ,  ENDLOOP

                        ; loop
0000'                   loop:               ;schleifenruecksprung
0000'   00               nop
0001'   00               nop                ;--- befehlssequenz
0002'   00               nop
                         ;exitif zero
0003'   28 0A            jr z,fina
0005'   00               nop
0006'   00               nop                ;--- befehlssequenz
0007'   00               nop
                         ;exitif no carry
0008'   30 05            jr nc,fina
000A'   00               nop
000B'   00               nop                ;--- befehlssequenz
000C'   00               nop
                        ; endloop
000D'   18 F1           jr loop             ;ruecksprung
000F'                   fina:               ;ende der schleife
000F'   00              nop
0010'   00              nop                 ;--- folge befehle
0011'   00              nop

                        end
```

Abb. 6.3.15 Assemblercodierung mit LOOP ENDLOOP

```
                        ; geschachtelte konstruktion

                        ;loop
0000'                   lpp:
0000'   00               nop
0001'   00               nop
0002'   00               nop
                         ;if zero
0003'   20 05            jr nz,sk1
0005'   00                nop
0006'   00                nop
0007'   00                nop
                         ;else
0008'   18 0B            jr sk2
000A'                    sk1:
000A'   00                nop
000B'   00                nop
000C'   00                nop
                          ;repeat
000D'                     rp1:
000D'   00                 nop
000E'   00                 nop
000F'   00                 nop
                          ;until no carry
0010'   38 FB             jr c,rp1
0012'   00                nop
0013'   00                nop
0014'   00                nop
                         ;endif
0015'                    sk2:
                         ;exitif carry
0015'   38 05            jr c,fina
0017'   00               nop
0018'   00               nop
0019'   00               nop
                        ;endloop
001A'   18 E4           jr lpp
001C'                   fina:
001C'   00              nop
001D'   00              nop
001E'   00              nop
```

Abb. 6.3.16 Geschachtelte Konstruktion

IF . . ELSE . . ENDIF wurde als Abkürzung für die Entscheidungskonstruktion verwendet, LOOP ENDLOOP zusammen mit EXITIF für die Endlosschleife und REPEAT UNTIL für die Wiederholungsschleife. Die Anweisungen innerhalb der Konstruktionen werden jeweils um einen Platz eingerückt, was besonders gut die Struktur des Programms darstellt. An Stelle des Begriffs Anweisung können beliebige Z80-Befehle stehen. Dabei dürfen sie keine Sprünge mehr enthalten, wohl aber z. B. Unterprogrammaufrufe. *Abb. 6.3.16* zeigt das Assemblerprogramm dazu.

Beim Entwurf eines Programms kann man so vorgehen, daß zuerst mit Hilfe der Begriffe IF ELSE ENDIF etc. eine grobe Struktur des Programms aufgezeichnet wird. Dann kann entweder direkt codiert werden, oder es werden die einzelnen Flußdiagramme anstelle der Struktur-Bezeichnungen gezeichnet. Es kann aber auch mit dem Zeichnen des Flußdiagramms begonnen werden. Wichtig ist dann, daß nur die vorgezeigten Strukturen verwendet werden und keine anderen, da sich sonst eventuell ein nicht strukturiertes Programm ergibt. Im nächsten Abschnitt ist noch ein komplettes Beispiel abgedruckt, nämlich das Monitorprogramm, das nach diesen Regeln entworfen wurde.

6.4 Das Monitorprogramm – Befehle und Aufbau

Nach dem der Rechner eingeschaltet wird, passiert erst einmal gar nichts, denn er braucht ein Programm. Wie aber kann ein solches Programm eingegeben werden; dazu braucht man wiederum ein Programm und nun steckt man in einer Zwickmühle. Der Computer muß also bereits vor dem Einschalten ein Programm fest gespeichert besitzen. Dieses Programm kann bei unserem Rechner in EPROMs abgelegt werden. Nach dem Start wird es sofort ausgeführt. Es hat die Aufgabe, eine Eingabe z. B. über die Tastatur eines Datensichtgerätes zu akzeptieren und muß Aktionen ausführen können. Ein solches Programm wird auch Monitorprogramm genannt. Mit Hilfe des Monitorprogrammes können eigene Programme eingegeben und gestartet werden. Ebenfalls ist es möglich, sich Register der CPU anzusehen, mit denen auch getestet werden kann.

Unser Monitorprogramm läßt sich in vielen Kombinationen der Rechnerzusammenstellung einsetzen.

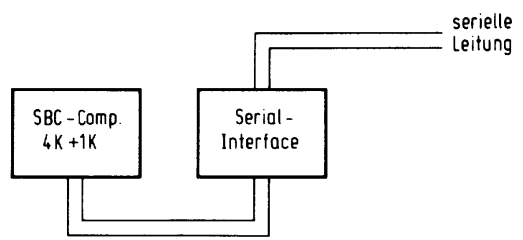


Abb. 6.4.1
Minimalkonfiguration

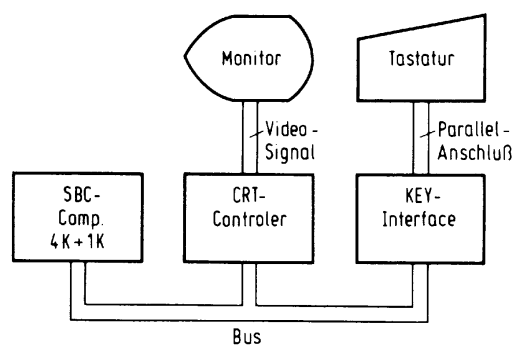


Abb. 6.4.2
Konfiguration mit
CRT-Karte

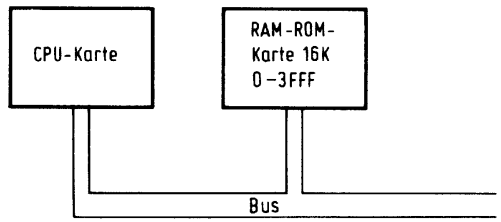


Abb. 6.4.3
Verwendung der
CPU-Karte

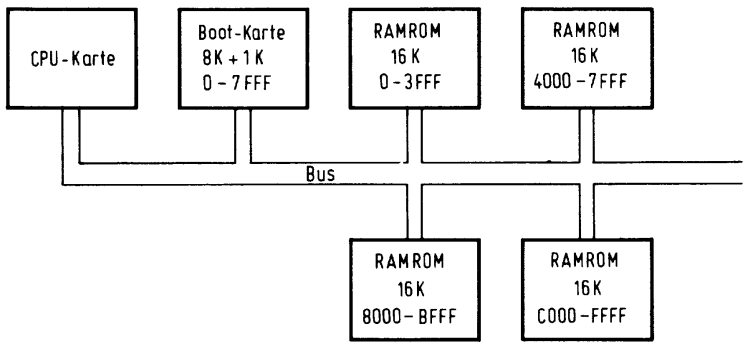


Abb. 6.4.4
Vollausbau des
Speichers

Abb. 6.4.1 zeigt die einfachste Variante. Die Karte SBC wird zusammen mit der Karte SER verwendet. Über eine serielle Leitung wird dann ein fertiges Datensichtgerät angeschlossen. Steht kein Datensichtgerät zur Verfügung, so kann die Konfiguration nach *Abb. 6.4.2* verwendet werden. Anstelle der SBC-Karte kann aber auch die große CPU-Karte + eine RAM/ROM-Karte verwendet werden, wie *Abb. 6.4.3* zeigt, oder es wird bei einem Voll-Ausbau auf 64K wie in *Abb. 6.4.4* vorgegangen. Bei der Verwendung der Boot-Karte muß jedoch eine etwas andere Form des Monitors verwendet werden.

Abb. 6.1 zeigt ein Datensichtgerät, das mit dem Computer verbunden wurde und in *Abb. 6.2* ist der Bildschirm vergrößert dargestellt. Auf dem Computer wurden ein paar Monitorbefehle ausgeführt.

Im folgenden werden die einzelnen Befehle des Monitors besprochen. *Abb. 6.4.5* zeigt das Programmlisting des Monitor-Programms.

Da der Monitor mit unterschiedlichen Konfigurationen arbeiten kann, muß ihm gesagt werden, welche davon vorliegt. Dies geschieht mit dem Dil-Schalter auf der Platine KEY. Der Monitor fragt dazu diesen Schalter ab. Ist die Karte KEY nicht vorhanden, so muß der Wert OFFh von dem Port 69h eingelesen werden können. Dies wird erreicht, in dem Widerstände mit einem Wert von ca. 1 KOHM am Datenbus angelötet werden und das andere Ende mit +5 V verbunden wird. Ist ein Port nicht existent, so wird ein High-Pegel eingelesen. Die Bedeutung des Wertes OFFh ist genau die Konfiguration, die dann vorliegt, nämlich ein Betrieb ohne Tastatur und ohne CRT-Karte. Die Daten werden dann über die serielle Schnittstelle geleitet. (Fortsetzung des Textes auf Seite 252).

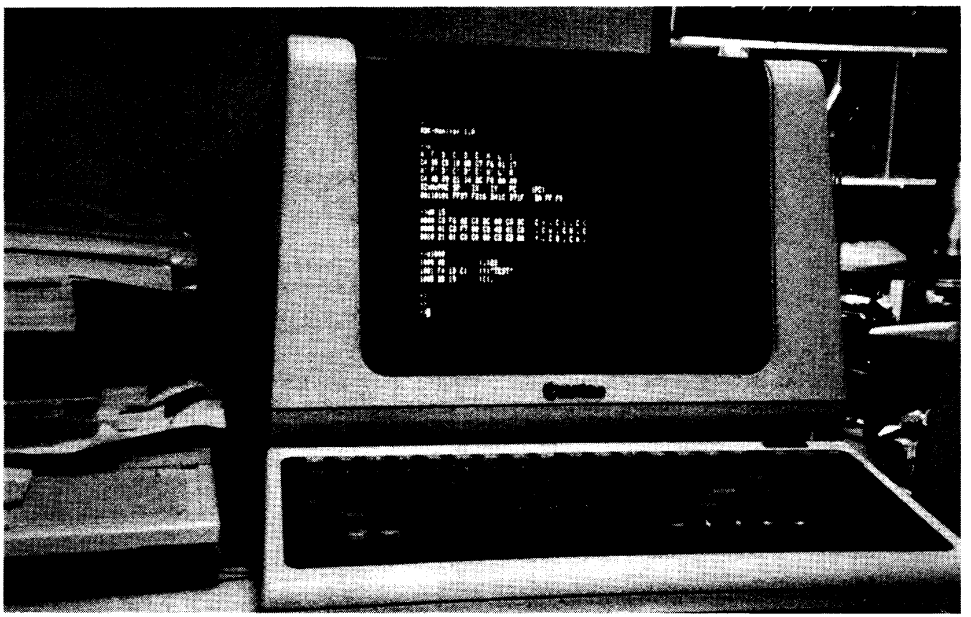


Abb. 6.1 Monitorausgabe auf einem Datensichtgerät

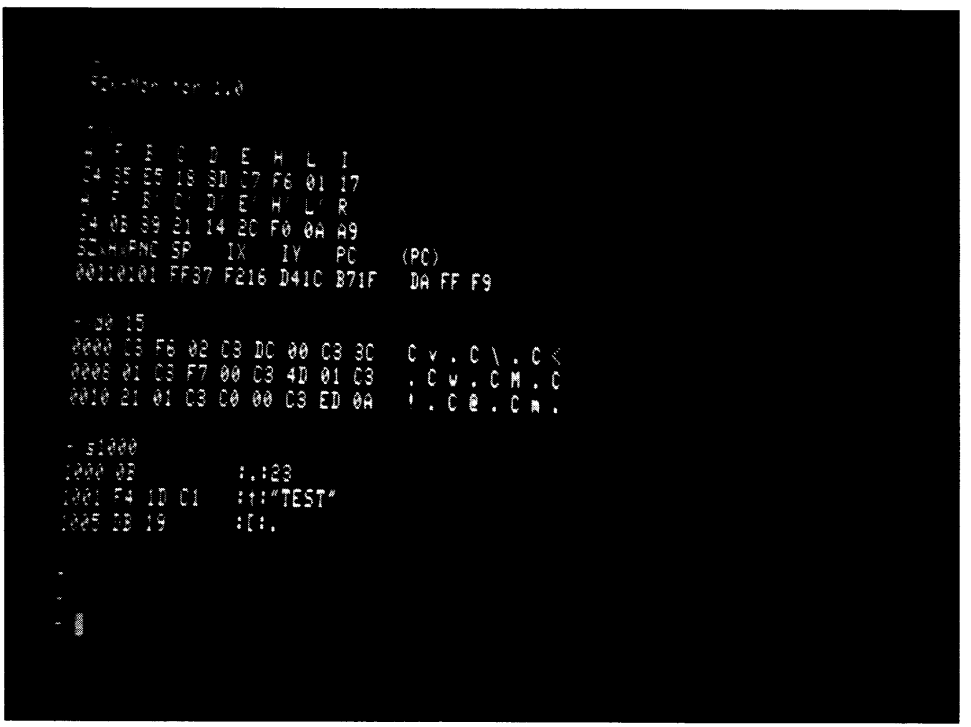


Abb. 6.2 Bildausschnitt des Datensichtgerätes

```
MACRO-80 3.43   27-Jul-81       PAGE    1


                        .z80
0000'                   cseg

                        ;**************************************
                        ;* Monitorprogramm von                *
                        ;* Rolf-Dieter Klein      820423      *
                        ;*                        V1.1        *
                        ;* mit terminal ringbuffer  820509    *
                        ;**************************************
                        ;
                        ; bereiche
                        ; SBC     0=monloc
                        ;      13ffh=endram
                        ;      1300h=endusr
                        ;
                        ; CP/M
                        ;    0f000h=monloc
                        ;    0ffffh=endram
                        ;    0efffh=endusr
                        ;
0000                    monloc  equ     0       ;rom bereich
13FF                    endram  equ     13ffh   ;last ram
1300                    endusr  equ     1300h   ;end user
                        ;
1200                    ringpo  equ     1200h   ;im R/W bereich 256 lang
11FF                    ringct  equ     1200h-1
11FE                    ringls  equ     1200h-2
                        ;
                        org monloc

                        ; Vektorbeschreibung
                        ; ci,ri
                        ; wert einer eingabe falls da in akku
                        ; sonst wird solange gewartet
                        ;
                        ; co,po,lo
                        ; ausgabe eines wertes ueber register c
                        ;
                        ; csts
                        ; status zeichen da=0ffh sonst 0
                        ;
                        ; iochk
                        ; aktuelle io-konfig in register a
                        ;
                        ; ioset
                        ; neuer wert in register c
                        ;
                        ; memck
                        ; hoechste speicherzelle fuer benutzer-
                        ; programme in register b (msb) a (lsb)
                        ;
                        ; restart
                        ; sprung fuer warm boot des monitors
                        ;
                        ; mini,maxi1,maxi2
                        ; standard - Aufuf fuer Floppy
                        ; HL = Adresse von der aus daten
                        ;      geladen oder gespeichert
                        ;      werden sollen
```

Abb. 6.4.5 Listing des Monitorprogramms

```
                MACRO-80 3.43   27-Jul-81       PAGE    1-1

                                ;
                                ; D  = Track Register
                                ; E  = Sektor Register
                                ;
                                ; B  = Befehl   0 = restore
                                ;               1 = read
                                ;               2 = write
                                ;
                                ; C  = Laufwerk 0,1,2,3
                                ;

                                ; Sprungtabelle fuer Standard-Eingabe

0000'   C3 031B'                jp start        ;hauptprogramm Start
0003'   C3 0101'                jp ci           ;console eingabe nach a
0006'   C3 0161'                jp ri           ;eingabe von reader
0009'   C3 011C'                jp co           ;ausgabe reg c nach con
000C'   C3 0172'                jp poo          ;ausgabe auf punch
000F'   C3 0146'                jp lo           ;ausgabe auf drucker
0012'   C3 00E5'                jp csts         ;status console 0ffh=da
0015'   C3 0B12'                jp iochk        ;i/o kanal zurueck
0018'   C3 0B11'                jp ioset        ;i/o kanal setzen
001B'   C3 0393'                jp memck        ;hoechster speicher reg b,a
001E'   C3 0A98'                jp rstart       ;warm start monitor BRK
                                ;
0021'   C3 003B'                jp mini         ;minifloppy vektor
0024'   C3 003E'                jp maxi1        ;soft maxi
0027'   C3 0041'                jp maxi2        ;reserve z.B. Platte
                                ;
                                org monloc+30h  ;fuer Interrupt-Experimente
0030'   C3 1300                 jp rst30jmp     ;in ram bereich

                                org monloc+38h  ;fuer rom versionen und SBC
0038'   C3 0A98'                jp rstart       ;hier BREAKPOINT

                                ;****** nicht implementierte funktionen

003B'                           mini:
003B'   C3 0A98'                jp rstart
003E'                           maxi1:
003E'   C3 0A98'                jp rstart
0041'                           maxi2:
0041'   C3 0A98'                jp rstart

                                ;
                                ; definitionen der i/o - Ports
                                ;

                                ; UART-Ports fuer Consol-Uart
00FC                            udat    equ     0fch            ;6551 UART
00FD                            usta    equ     0fdh
00FE                            ucmd    equ     0feh
00FF                            ucon    equ     0ffh

                                ; Uart-Ports fuer Printer-Port
00EC                            udat2   equ     0ech            ;6551 UART
00ED                            usta2   equ     0edh
00EE                            ucmd2   equ     0eeh
00EF                            ucon2   equ     0efh
                                ;
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-2


                        ; Ports fuer Kassetteninterface
00CA                    cmdcas  equ     0cah            ;6850 UART
00CB                    datcas  equ     0cbh

                        ; prom programmier-Ports
0080                    promd   equ     80h             ;daten bi-dir
0081                    proma1  equ     81h             ;lsb adr
0082                    proma2  equ     82h             ;spez
                        ; proma2
                        ; 7     6     5     4     3     2     1     0
                        ; ena   led   trg   a12   a11   a10   a9    a8
                        ;
                        ; bei read
                        ; 0     1     0
                        ;
                        ; bei write
                        ; 1     0     0
                        ; 1     0     1
                        ; 1     0     0
                        ;
                        ; proma1 lesen
                        ; x     x     x     x     x     x     x     bsy
                        ; bsy=1 bei programmieren sonst 0
                        ;
                        ;
                        ; crt-controler ef9364
0060                    crtdat  equ     60h     ;daten
0061                    crtctr  equ     61h     ;control
                        ;
                        ; 7     6     5     4     3     2     1     0
                        ;-st    x     x     x    en     c2    c1    c0
                        ;

                        ; tastaturports
0068                    keyd    equ     68h     ;daten bit7=0 dann rdy
0069                    keys    equ     69h     ;switch optionen loescht
                                                ;keyd
                        ; schalterbelegung
                        ;
                        ; 7     6     5     4     3     2     1     0
                        ; crt   kbd   bd    cas   bd    bd    bd    bd
                        ;
                        ; 0 = crt da
                        ;       0 = keyboard da
                        ;             0 = baudrate gueltig
                        ;                   0 = auf co schalten
                        ;                         baudrate
                        ; von 1=50,75,109.92,134.58,150,300,600,1200
                        ;     1800,2400,3600,4800,7200,9600,19200 = 16
                        ; 0=external (modifizieren der SER-Karte)
                        ; ist der port nicht vorhanden, so wird 9600
                        ; und nicht crt,nicht kbd angenommen
                        ; ist der Port nicht vorhanden, so muss beim Daten-
                        ; bus mit Pullupwiderstaenden gearbeitet werden,
                        ; da sonst der status undefiniert ist. Es muss
                        ; 0ffh eingelesen werden koennen auch ohne Port.

                        org monloc+66h          ;fuer NMI-Experimente
0066'   C3 1303        jp nmijmp               ;in rambereich liegt
                                                ;der sprung
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-3


                                ;
                                ; IO-Routinen

                                ; Consol-Uart

0069'                           coninit:
0069'   3E 1E                  ld a,1eh        ;9600 BAUD 1 stop 8 bits
006B'   D3 FF                  out (ucon),a
006D'   3E 0B                  ld a,00001011b  ;no par, dtr-
006F'   D3 FE                  out (ucmd),a
0071'   0E 0C                  ld c,0ch        ;FF CLEAR SCREEN
0073'   CD 0B8C'               call c93co      ;auch wenn nicht da
0076'   DB 69                  in a,(keys)     ;tastatur port test
0078'   00                     nop
0079'   00                     nop             ;fuer aendern
007A'   00                     nop             ;und einfuegen
007B'   32 132A                ld (keyboa),a   ;MERKEN HIER STATUS
007E'   CB 6F                  bit 5,a         ;bd=1 dann ok
0080'   C0                     ret nz
0081'   E6 0F                  and 0fh         ;baudrate holen
0083'   F6 10                  or 10h          ;1stop 8 data
0085'   D3 FF                  out (ucon),a    ;neu def
0087'   C9                     ret

                                ; routine fuer Terminal-Emulator
                                ; benoetigt UART und KEY
                                ; endlosschleife
                                ; terminal mit ringbuffer
                                ; ----------------------------------------
                                ; damit kann SBC + CRT + KEY als
                                ; eigenes Terminal verwendet werden

0088'                           terminal:
0088'   CD 00EC'               call csts1      ;uart - > crt
008B'   20 12                  jr nz,auscr
008D'   CD 00F5'               call kbds       ;key - > uart
0090'   20 16                  jr nz,ausua
0092'   CD 00C8'               call get        ;in akku wert
0095'   30 02                  jr nc,ausc1
0097'   18 EF                  jr terminal
0099'                           ausc1:          ;exec ausgabe
0099'   4F                     ld c,a
009A'   CD 012D'               call co2        ;und auf crt geben
009D'   18 E9                  jr terminal
009F'                           auscr:
009F'   CD 0108'               call ci1
00A2'   4F                     ld c,a
00A3'   CD 00B1'               call put        ;ueber ring
00A6'   18 E0                  jr terminal
00A8'                           ausua:
00A8'   CD 0111'               call ci2        ;von key
00AB'   4F                     ld c,a
00AC'   CD 0123'               call co1        ;nach uart
00AF'   18 D7                  jr terminal
                                ;
00B1'                           put:
00B1'   E5                     push hl
00B2'   C5                     push bc
00B3'   21 1200                ld hl,ringpo    ;pointer laden
00B6'   3A 11FF                ld a,(ringct)
00B9'   4F                     ld c,a
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-4


00BA'   06 00               ld b,0
00BC'   09                  add hl,bc               ;+
00BD'   C1                  pop bc
00BE'   71                  ld (hl),c       ;ablegen
00BF'   3A 11FF             ld a,(ringct)
00C2'   3C                  inc a           ;256
00C3'   32 11FF             ld (ringct),a   ;next free pos
00C6'   E1                  pop hl
00C7'   C9                  ret

00C8'                       get:
00C8'   3A 11FF             ld a,(ringct)   ;vergleich
00CB'   47                  ld b,a
00CC'   3A 11FE             ld a,(ringls)   ;beide gleich
00CF'   B8                  cp b            ;dann keine daten da
00D0'   20 02               jr nz,get1
00D2'   37                  scf             ;set carry
00D3'   C9                  ret
00D4'                       get1:
00D4'   E5                  push hl
00D5'   21 1200             ld hl,ringpo
00D8'   4F                  ld c,a
00D9'   3C                  inc a
00DA'   32 11FE             ld (ringls),a
00DD'   06 00               ld b,0
00DF'   09                  add hl,bc
00E0'   7E                  ld a,(hl)
00E1'   E1                  pop hl
00E2'   37                  scf
00E3'   3F                  ccf
00E4'   C9                  ret                     ;no carry data avail


00E5'                       csts:           ;status test
00E5'   3A 132A             ld a,(keyboa)   ;DEVICE
00E8'   CB 77               bit 6,a         ;0=keyboard da
00EA'   28 09               jr z,kbds
00EC'                       csts1:          ;nur usart
00EC'   DB FD               in a,(usta)
00EE'   E6 08               and 00001000b   ;input port
00F0'   C8                  ret z           ;=0 ok keine daten
00F1'   3E FF               ld a,0ffh       ;0ffh wenn da
00F3'   B7                  or a            ;flags setzen
00F4'   C9                  ret
                            ;
00F5'                       kbds:
00F5'   DB 68               in a,(keyd)     ;bit7=0 dann da
00F7'   CB 7F               bit 7,a
00F9'   28 02               jr z,kb1        ;ja da
00FB'   AF                  xor a
00FC'   C9                  ret             ;nein
00FD'                       kb1:
00FD'   3E FF               ld a,0ffh
00FF'   B7                  or a
0100'   C9                  ret             ;ja
                            ;


                            ;
0101'                       ci:
0101'   3A 132A             ld a,(keyboa)
0104'   CB 77               bit 6,a         ;kbd=da=0
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-5


0106'   28 09                   jr z,kbdin
0108'                           ci1:
0108'   DB FD                   in a,(usta)     ;pruefen on data da
010A'   E6 08                   and 00001000b   ;status
010C'   28 FA                   jr z,ci1        ;warten bis da
010E'   DB FC                   in a,(udat)     ;holen des zeichens
0110'   C9                      ret
                                ;

                                ;
0111'                           ci2:            ;eingang 2
0111'                           kbdin:
0111'   DB 68                   in a,(keyd)     ;test
0113'   CB 7F                   bit 7,a
0115'   20 FA                   jr nz,kbdin     ;bis =0
0117'   F5                      push psw        ;merken
0118'   DB 69                   in a,(keys)     ;damit loeschen status
011A'   F1                      pop psw
011B'   C9                      ret



011C'                           co:
011C'   3A 132A                ld a,(keyboa)
011F'   CB 7F                   bit 7,a         ;=0 dann crt
0121'   28 0A                   jr z,co2        ;ausgabe auf CRT
0123'                           co1:
0123'   DB FD                   in a,(usta)     ;status lesen
0125'   E6 10                   and 00010000b   ;test ausgabe fertig
0127'   28 FA                   jr z,co1        ;warten solange
0129'   79                      ld a,c          ;holen des ausgabewertes
012A'   D3 FC                   out (udat),a    ;und ausgeben
012C'   C9                      ret
                                ;
012D'                           co2:
012D'   C5                      push bc
012E'   CD 0B8C'                call c93co      ;retten und crt
0131'   C1                      pop bc
0132'   79                      ld a,c          ;auch in akku
0133'   C9                      ret
                                ;

                                ; Drucker-Port

0134'                           loinit:
0134'   3E 1E                   ld a,1eh        ;9600 BAUD 1 stop 8 bits
0136'   D3 EF                   out (ucon2),a
0138'   3E 0B                   ld a,00001011b  ;no par, dtr-
013A'   D3 EE                   out (ucmd2),a
013C'   C9                      ret

013D'                           li:
013D'   DB ED                   in a,(usta2)    ;pruefen on data da
013F'   E6 08                   and 00001000b   ;status
0141'   28 FA                   jr z,li         ;warten bis da
0143'   DB EC                   in a,(udat2)    ;holen des zeichens
0145'   C9                      ret

0146'                           lo:
0146'   3A 132A                ld a,(keyboa)
0149'   CB 67                   bit 4,a         ;cas umschaltung
```

Abb. 6.4.5 Listing des Monitorprogramms

```
       MACRO-80 3.43   27-Jul-81       PAGE    1-6


014B'   CA 0123'              jp z,co1        ;echtes CO verwenden
014E'                        lo1:
014E'   DB ED                 in a,(usta2)   ;status lesen
0150'   E6 10                 and 00010000b  ;test ausgabe fertig
0152'   28 FA                 jr z,lo1       ;warten solange
0154'   79                    ld a,c         ;holen des ausgabewertes
0155'   D3 EC                 out (udat2),a  ;und ausgeben
0157'   C9                    ret


                             ; Cassetten-Port

0158'                        casinit:
0158'   3E 53                 ld a,53h       ;*1 baudrate
015A'   D3 CA                 out (cmdcas),a
015C'   3E 50                 ld a,50h
015E'   D3 CA                 out (cmdcas),a
0160'   C9                    ret

0161'                        ri:
0161'   3A 132A               ld a,(keyboa)
0164'   CB 67                 bit 4,a
0166'   CA 0108'              jp z,ci1       ;echtes CI verwenden dann
0169'                        ri1:
0169'   DB CA                 in a,(cmdcas)  ;test cassettenstatus
016B'   E6 01                 and 00000001b  ;data da
016D'   28 FA                 jr z,ri1       ;warten bis da
016F'   DB CB                 in a,(datcas)  ;wert holen
0171'   C9                    ret

0172'                        poo:
0172'   3A 132A               ld a,(keyboa)
0175'   CB 67                 bit 4,a
0177'   CA 0123'              jp z,co1       ;echtes co verwenden
017A'                        poo1:
017A'   DB CA                 in a,(cmdcas)
017C'   E6 02                 and 00000010b  ;warten bis ausgabe
017E'   28 FA                 jr z,poo1      ;fertig ist
0180'   79                    ld a,c         ;data holen
0181'   D3 CB                 out (datcas),a ;und ausgeben
0183'   C9                    ret

                             ;
0184'                        bell:          ;FERTIG MELDUNG
                                            ;z.B. fuer CAS
0184'   0E 07                 ld c,7
0186'   CD 011C'              call co        ;ggf soundgen danach
0189'   C9                    ret



                             ; allgemeine unterprogramme

                             ; Bestimmung der Laenge eines Z80-Befehls
                             ; um Pseudo-Disassemblierung
                             ; durchfuehren zu koennen

                             ; hl -> zeigt aus den Befehl
                             ; b nach dem Aufruf die Anzahl der Bytes
                             ;   die der Z80-Befehl belegt
```

```
MACRO-80 3.43   27-Jul-81       PAGE    1-7


018A'                           length:
018A'   06 00                   ld b,0          ;bytezaehler auf 0 stellen
018C'   E5                      push hl
018D'   D1                      pop de          ;zeiger retten nach de
018E'   7E                      ld a,(hl)       ;1.befehlsbyte holen
018F'   E6 DF                   and 0dfh        ;dd,fd-Befehle
0191'   FE DD                   cp 0ddh
0193'   CA 0200'               jp z,tab5       ;sprung auf die tabelle 5
0196'   7E                      ld a,(hl)
0197'   FE CB                   cp 0cbh         ;cb-Befehle
0199'   CA 01FA'               jp z,tab3       ;tabelle 3
019C'   FE ED                   cp 0edh         ;ed-Befehle
019E'   CA 01EE'               jp z,tab2       ; tabelle 2
01A1'   7E                      ld a,(hl)
01A2'   FE C3                   cp 0c3h         ;jmp-Befehle
01A4'   CA 0237'               jp z,b3         ;3 bytes
01A7'   FE CD                   cp 0cdh
01A9'   CA 0237'               jp z,b3         ;3 bytes
01AC'   E6 EF                   and 0efh        ;neue klasse
01AE'   FE 22                   cp 022h
01B0'   CA 0237'               jp z,b3         ;3 bytes
01B3'   FE 2A                   cp 02ah
01B5'   CA 0237'               jp z,b3         ;3 bytes
01B8'   E6 CF                   and 0cfh
01BA'   FE 01                   cp 1            ;ld xx,nn
01BC'   CA 0237'               jp z,b3         ;3 bytes
01BF'   E6 C7                   and 0c7h
01C1'   FE C2                   cp 0c2h         ;jp cond
01C3'   CA 0237'               jp z,b3         ;3 bytes
01C6'   FE C4                   cp 0c4h         ;call cond
01C8'   CA 0237'               jp z,b3         ;3 bytes
01CB'   7E                      ld a,(hl)       ;2 byte befehle
01CC'   E6 F7                   and 0f7h
01CE'   FE 10                   cp 10h          ;DJNZ,JR
01D0'   CA 0238'               jp z,b2
01D3'   FE D3                   cp 0d3h         ;in,out
01D5'   CA 0238'               jp z,b2
01D8'   E6 E7                   and 0e7h
01DA'   FE 20                   cp 20h          ;jr cond
01DC'   CA 0238'               jp z,b2
01DF'   E6 C7                   and 0c7h
01E1'   FE 06                   cp 6            ; ld n
01E3'   CA 0238'               jp z,b2
01E6'   FE C6                   cp 0c6h         ; op n
01E8'   CA 0238'               jp z,b2
01EB'   C3 0239'               jp b1           ;rest 1-byte-befs

                                ;
01EE'                           tab2:
01EE'   23                      inc hl          ;next bef holen
01EF'   7E                      ld a,(hl)
01F0'   E6 C7                   and 0c7h
01F2'   FE 43                   cp 43h          ;ld(nn),xx, ld xx,(nn)
01F4'   CA 0236'               jp z,b4
01F7'   C3 0238'               jp b2           ;sonst zwei bytes
                                ;

01FA'                           tab3:
01FA'   C3 0238'               jp b2           ;alles 2 byte-befs

01FD'                           tab4:
```

```
MACRO-80 3.43   27-Jul-81       PAGE    1-8


01FD'   C3 0236'            jp b4           ;alles 4 byte-befs

0200'                       tab5:           ;dd,fd-tab
0200'   23                  inc hl          ;next wert
0201'   7E                  ld a,(hl)
0202'   FE CB               cp 0cbh         ;dd,fd-cb tabelle
0204'   CA 0236'            jp z,b4         ;immer 4 bytes
0207'   FE 21               cp 21h          ;ld ii,nn
0209'   CA 0236'            jp z,b4
020C'   E6 FE               and 0feh
020E'   FE 34               cp 34h
0210'   CA 0237'            jp z,b3         ;inc dec (ii)
0213'   E6 F8               and 0f8h
0215'   FE 70               cp 70h          ;ld (ii),r
0217'   CA 0237'            jp z,b3
021A'   7E                  ld a,(hl)
021B'   E6 CF               and 0cfh
021D'   FE 06               cp 6            ;ld (ii),n
021F'   CA 0236'            jp z,b4
0222'   E6 C7               and 0c7h
0224'   FE 02               cp 2            ;ld (nn),ii
0226'   CA 0236'            jp z,b4
0229'   7E                  ld a,(hl)
022A'   D6 40               sub 40h
022C'   E6 87               and 87h
022E'   FE 06               cp 6
0230'   CA 0237'            jp z,b3         ;ld r,(ii)
0233'   C3 0238'            jp b2           ;rest 2 bytes

0236'   04                  b4:     inc b
0237'   04                  b3:     inc b
0238'   04                  b2:     inc b
0239'   04                  b1:     inc b
023A'   EB                  ex de,hl        ;alter pointer
023B'   C9                  ret

                            ; ---------------------------------
                            ;

                            ; hilfsroutinen
023C'                       echo:           ;holt ein zeichen und
023C'   CD 0101'            call ci         ;gibt echo aus
023F'   E6 7F               and 07fh        ;no paritaet
0241'   3C                  inc a
0242'   F8                  ret m           ;ignore rubs
0243'   3D                  dec a
0244'   C8                  ret z           ;ignore nulls
0245'   FE 4E               cp 'N'
0247'   C8                  ret z
0248'   FE 6E               cp 'n'
024A'   28 10               jr z,ec1
024C'   FE 0D                cp 0dh
024E'   C8                    ret z         ;ignore crs
024F'   C5                    push bc
0250'   4F                    ld c,a
0251'   CD 011C'              call co               ;echo
0254'   79                    ld a,c
0255'   C1                    pop bc
0256'   FE 40                cp 'A'-1      ;convert to upper
0258'   D8                    ret c
0259'   FE 7B                cp 'z'+1
```

Abb. 6.4.5 Listing des Monitorprogramms

```
      MACRO-80 3.43   27-Jul-81       PAGE    1-9


025B'   D0                    ret nc
025C'                         ec1:
025C'   E6 5F                 and 05fh
025E'   C9                    ret


                              ; expr holt zeichen von der console
                              ; interpretiert diese als zahl und
                              ; liefert das ergebnis in register hl
                              ; falls keine zahl da war ergibt sich
                              ; carry als wert das zeichen steht
                              ; dann a wie auch das terminalzeichen


025F'                         expr:
025F'   21 0000               ld hl,0         ;startwert 0
0262'   CD 023C'              call echo       ;zeichen holen
0265'   47                    ld b,a          ;retten
0266'   CD 028C'              call nibble     ;test ob zahl
0269'   78                    ld a,b          ;alten wert
026A'   30 02                 jr nc,ex1       ;weiter wenn ok
026C'   37                     scf            ;carry=kein guelt. zahl
026D'   C9                     ret            ;in a zeichenwert
026E'                         ex1:            ;loop
026E'   47                     ld b,a
026F'   CD 028C'               call nibble    ;test in schleife
0272'   38 0B                  jr c,ex2       ;ok exit ausfuehren
0274'   29                     add hl,hl
0275'   29                     add hl,hl
0276'   29                     add hl,hl
0277'   29                     add hl,hl      ;hl * 16 -> hl
0278'   B5                     or l           ;dazu maskieren
0279'   6F                     ld l,a         ;wert von akku
027A'   CD 023C'               call echo      ;next zeichen holen
027D'   18 EF                 jr ex1          ;endloop
027F'                         ex2:            ;ende teil
027F'   78                    ld a,b          ;alten wert
0280'   37                    scf
0281'   3F                    ccf             ;no carry
0282'   C9                    ret             ;in a terminatorzeichen
                              ;
                              ; input no par
0283'                         rix:
0283'   CD 0161'              call ri
0286'   E6 7F                 and 07fh
0288'   C9                    ret
                              ;

0289'                         ribble:         ;einsprung von reader
0289'   CD 0283'              call rix
028C'                         nibble:         ;test ob zahl   0..9,a..f
028C'   D6 30                 sub '0'
028E'   D8                    ret c
028F'   FE 17                 cp 'G'-'0'
0291'   3F                    ccf
0292'   D8                    ret c
0293'   FE 0A                 cp 10
0295'   3F                    ccf
0296'   D0                    ret nc
0297'   D6 07                 sub 'A'-'9'-1
0299'   FE 0A                 cp 10
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-10

029B'   C9                      ret             ;carry dann nicht im bereich

                                ;
                                ;
                                ;
                                ; prhl : ausgabe des Hl-Inhaltes HEX
                                ; prac : ausgabe des akkuinhaltes auf console


029C'                           prhl:
029C'   7C                      ld a,h
029D'   CD 02A1'                call prac
02A0'   7D                      ld a,l          ;direkt weiter
02A1'                           prac:
02A1'   F5                      push psw
02A2'   1F                      rra             ;nach rechts schieben
02A3'   1F                      rra
02A4'   1F                      rra
02A5'   1F                      rra             ;damit msb teil 0..3
02A6'   CD 02AA'                call outh       ;teil ausgeben
02A9'   F1                      pop psw
02AA'                           outh:           ;doppelt nutzen
02AA'   E6 0F                   and 0fh
02AC'   C6 30                   add a,'0'       ;konvertieren
02AE'   FE 3A                   cp '9'+1        ;
02B0'   38 02                   jr c,ak1        ;ausgabe
02B2'   C6 07                    add a,'A'-'9'-1
02B4'                           ak1:
02B4'   4F                      ld c,a
02B5'   C3 011C'                jp co           ;ausgabe
                                ;
02B8'                           peol:
02B8'   0E 0D                   ld c,0dh
02BA'   CD 0172'                call poo
02BD'   0E 0A                   ld c,0ah
02BF'   C3 0172'                jp poo          ;CRLF
                                ;
02C2'                           punhl:
02C2'   7C                      ld a,h
02C3'   CD 02C7'                call punac
02C6'   7D                      ld a,l          ;direkt weiter
02C7'                           punac:
02C7'   F5                      push psw
02C8'   1F                      rra             ;nach rechts schieben
02C9'   1F                      rra
02CA'   1F                      rra
02CB'   1F                      rra             ;damit msb teil 0..3
02CC'   CD 02D0'                call puth       ;teil ausgeben
02CF'   F1                      pop psw
02D0'                           puth:           ;doppelt nutzen
02D0'   E6 0F                   and 0fh
02D2'   C6 30                   add a,'0'       ;konvertieren
02D4'   FE 3A                   cp '9'+1        ;
02D6'   38 02                   jr c,pak1       ;ausgabe
02D8'   C6 07                    add a,'A'-'9'-1
02DA'                           pak1:
02DA'   4F                      ld c,a
02DB'   C3 0172'                jp poo          ;ausgabe
                                ;
                                ; read von ri
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-11


02DE'                           sbyte:          ;nach akku holen
02DE'   C5                      push bc         ;retten
02DF'   CD 0289'                call ribble
02E2'   07                      rlca
02E3'   07                      rlca
02E4'   07                      rlca
02E5'   07                      rlca
02E6'   4F                      ld c,a          ;temp
02E7'   CD 0289'                call ribble
02EA'   B1                      or c            ;dazu maskieren
02EB'   C1                      pop bc          ;in akku wert
02EC'   C9                      ret
                                ;

                                ; ausgabe eines textes auf der console
                                ; hl -> start des textes, 0=ende
02ED'                           print:          ;loop
02ED'   7E                       ld a,(hl)      ;zeichen holen
02EE'   B7                       or a
02EF'   C8                       ret z          ;exitif zero
02F0'   4F                       ld c,a         ;ausgeben
02F1'   CD 011C'                 call co
02F4'   23                       inc hl         ;next
02F5'   18 F6                   jr print        ;endloop

02F7'                           crlf:
02F7'   0E 0D                   ld c,0dh
02F9'   CD 011C'                call co
02FC'   0E 0A                   ld c,0ah
02FE'   CD 011C'                call co
0301'   C9                      ret
                                ;
0302'                           txt1:
0302'   0D 0A                   defb 0dh,0ah
0304'   52 44 4B 2D            defm 'RDK-Monitor 1.0'
0308'   4D 6F 6E 69
030C'   74 6F 72 20
0310'   31 2E 30
0313'   0D 0A 00                defb 0dh,0ah,0
                                ;
0316'                           txt2:
0316'   0D 0A                   defb 0dh,0ah
0318'   2D 3E                   defm '->'
031A'   00                      defb 0
                                ;

031B'                           start:          ;start sp laden
031B'   31 13FF                 ld sp,stack
031E'   CD 0069'                call coninit    ;alle ports initialisieren
0321'   CD 0134'                call loinit
0324'   CD 0158'                call casinit
0327'   3E 00                   ld a,0
0329'   32 11FE                 ld (ringls),a   ;ringbuffer:=leer
032C'   32 11FF                 ld (ringct),a   ;beide pointer
032F'   21 0302'                ld hl,txt1      ;start meldung ausgeben
0332'   CD 02ED'                call print
0335'   3E FF                   ld a,0ffh       ;ins1,ins2 flag no TRACE
0337'   32 1324                 ld (ins1),a
033A'   32 1326                 ld (ins2),a
                                ;
033D'                           lpm:            ;hauptschleife
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-12


033D'   11 033D'              ld de,lpm       ;ruecksprungadr
0340'   D5                    push de         ;auf stack legen
0341'   21 0316'              ld hl,txt2      ;meldung
0344'   CD 02ED'              call print      ;ausgeben
0347'                         lpm1:
0347'   CD 023C'              call echo       ;holen zeichen
034A'   28 FB                 jr z,lpm1       ;=0 dann neuen holen
034C'   D6 41                 sub 'A'
034E'   F8                    ret m           ;<A
034F'   FE 1A                 cp 'Z'-'A'+1    ;>Z
0351'   D0                    ret nc
0352'   87                    add a,a         ;*2
0353'   06 00                 ld b,0
0355'   4F                    ld c,a
0356'   21 035F'              ld hl,tbl       ;commando tabelle
0359'   09                    add hl,bc       ;alle buchstaben
035A'   5E                    ld e,(hl)
035B'   23                    inc hl
035C'   56                    ld d,(hl)
035D'   EB                    ex de,hl        ;nach hl sprungadr
035E'   E9                    jp (hl)         ;ausfuehren


035F'                         tbl:            ;tabelle aller befehle
035F'   0399'                 defw    error   ;a
0361'   0399'                 defw    error   ;b
0363'   0399'                 defw    error   ;c
0365'   0793'                 defw    displa  ;d  display
0367'   0399'                 defw    error   ;e
0369'   0516'                 defw    fill    ;f  fill
036B'   0A15'                 defw    goto    ;g  goto
036D'   0399'                 defw    error   ;h
036F'   0399'                 defw    error   ;i
0371'   0399'                 defw    error   ;j
0373'   0399'                 defw    error   ;k
0375'   0399'                 defw    error   ;l
0377'   0548'                 defw    move    ;m  move
0379'   05EE'                 defw    nulls   ;n  ausgabe nuls
037B'   0399'                 defw    error   ;o
037D'   0421'                 defw    promer  ;p  prom prg
037F'   0696'                 defw    querry  ;q  io-ports
0381'   0574'                 defw    read    ;r  lesen cas
0383'   06CE'                 defw    subst   ;s  substitute
0385'   0088'                 defw    terminal ;t  terminal mode
0387'   0399'                 defw    error   ;u
0389'   039F'                 defw    verify  ;v  verify
038B'   05FA'                 defw    write   ;w  schreiben cas
038D'   0827'                 defw    regist  ;x  register
038F'   0399'                 defw    error   ;y
0391'   0399'                 defw    error   ;z
                              ;



0393'                         memck:          ;letzte userzelle
0393'   21 1300              ld hl,endusr    ;fest
0396'   44                    ld b,h
0397'   7D                    ld a,l
0398'   C9                    ret
```

Abb. 6.4.5 Listing des Monitorprogramms

```
      MACRO-80 3.43   27-Jul-81       PAGE    1-13

0399'                         error:          ;allgemeiner fehler
0399'   0E 2A                ld c,'*'
039B'   CD 011C'              call co
039E'   C9                    ret             ;ende

039F'                         verify:         ;vergleichen von bereichen
                                              ;V<expr> <expr> <expr>
                                              ; von    bis    mit
039F'   CD 025F'              call expr
03A2'   DA 0399'              jp c,error
03A5'   E5                    push hl
03A6'   DD E1                 pop ix          ;start
03A8'   CD 025F'              call expr
03AB'   DA 0399'              jp c,error
03AE'   E5                    push hl
03AF'   FD E1                 pop iy          ;ende
03B1'   CD 025F'              call expr
03B4'   DA 0399'              jp c,error      ;mit in hl
03B7'                         verlp:          ;LOOP
03B7'   DD 7E 00               ld a,(ix+0)    ;start
03BA'   BE                     cp (hl)        ;vergleichen
03BB'   28 32                  jr z,versk     ;skip
03BD'   E5                      push hl
03BE'   DD E5                   push ix
03C0'   FD E5                   push iy
03C2'   CD 02F7'                call crlf
03C5'   DD E5                   push ix
03C7'   D1                      pop de         ;vergleichen
03C8'   EB                      ex de,hl
03C9'   E5                      push hl
03CA'   CD 029C'                call prhl      ;adr aus
03CD'   E1                      pop hl
03CE'   0E 20                   ld c,' '
03D0'   CD 011C'                call co        ;aaaa_
03D3'   7E                      ld a,(hl)
03D4'   CD 02A1'                call prac
03D7'   0E 20                   ld c,' '
03D9'   CD 011C'                call co        ;aaaa_nn_
03DC'   1A                      ld a,(de)
03DD'   CD 02A1'                call prac      ;aaaa_nn_nn
03E0'   0E 20                   ld c,' '
03E2'   CD 011C'                call co        ;aaaa_nn_nn_
03E5'   1A                      ld a,(de)
03E6'   AE                      xor (hl)       ;differenz
03E7'   CD 09EB'                call bitaus    ;aaaa_nn_nn_bbbbbbbb
03EA'   FD E1                   pop iy
03EC'   DD E1                   pop ix
03EE'   E1                      pop hl
03EF'                         versk:
03EF'   E5                    push hl                 ;retten
03F0'   DD E5                 push ix                 ;von
03F2'   D1                    pop de
03F3'   FD E5                 push iy
03F5'   E1                    pop hl          ;hl bis
03F6'   AF                    xor a
03F7'   ED 52                 sbc hl,de
03F9'   7D                    ld a,l
03FA'   B4                    or h
03FB'   E1                    pop hl
03FC'   28 05                 jr z,verfin     ;ende
03FE'   DD 23                 inc ix          ;next
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-14


0400'   23                      inc hl          ;bis adr
0401'   18 B4                  jr verlp        ;ENDLOOP
0403'                           verfin:
0403'   CD 02F7'               call crlf
0406'   C9                     ret
                                ;


0407'                           txtb1:
0407'   50 52 47 20            defm 'PRG YES=Y '
040B'   59 45 53 3D
040F'   59 20
0411'   00                      defb 0

0412'                           txtb2:
0412'   42 55 52 4E            defm 'BURNED'
0416'   45 44
0418'   00                      defb 0

0419'                           txtb3:
0419'   50 52 47 20            defm 'PRG ERR'
041D'   45 52 52
0420'   00                      defb 0

                                ;
0421'                           promer:         ;prom programer
0421'   CD 023C'               call echo       ;pr, pw , read werite
0424'   FE 52                  cp 'R'
0426'   CA 04E9'               jp z,prrd
0429'   FE 57                  cp 'W'
042B'   C2 0399'               jp nz,error     ;write
                                                ;pw <expr> <expr> (expr)
                                                ;last optional prganfang so 0
042E'   CD 025F'               call expr       ;start adresse
0431'   DA 0399'               jp c,error
0434'   E5                     push hl
0435'   CD 025F'               call expr       ;endadr
0438'   DD E1                  pop ix          ;start in ix, endadr iy
043A'   DA 0399'               jp c,error
043D'   E5                     push hl
043E'   FD E1                  pop iy
0440'   D9                     exx
0441'   21 0000                ld hl,0         ;PROM adr
0444'   D9                     exx
0445'   FE 0D                  cp 0dh          ;last
0447'   28 0C                  jr z,prs1
0449'   CD 025F'               call expr       ;prom adr
044C'   DA 0399'               jp c,error
044F'   E5                     push hl
0450'   D9                     exx
0451'   E1                     pop hl
0452'   E5                     push hl
0453'   D1                     pop de          ;de' reserve
0454'   D9                     exx
0455'                           prs1:
0455'   3E 40                  ld a,01000000b
0457'   D3 82                  out (proma2),a  ;init neutral
0459'   CD 02F7'               call crlf
045C'   21 0407'               ld hl,txtb1     ;fragen ob ja
045F'   CD 02ED'               call print      ;einlegen proms hier
0462'   CD 023C'               call echo
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81        PAGE    1-15


0465'   FE 59                  cp 'Y'
0467'   C2 0399'               jp nz,error      ;no
046A'   CD 02F7'               call crlf        ;ok start
046D'   DD E5                  push ix
046F'   E1                     pop hl           ;start
0470'   FD E5                  push iy
0472'   D1                     pop de           ;ende
0473'   3E 80                  ld a,10000000b   ;tristate neutral
0475'                          prlp1:           ;main loop iy to ix
0475'   7E                     ld a,(hl)        ;datenwert
0476'   D3 80                  out (promd),a
0478'   D9                     exx              ;' reg
0479'   7D                     ld a,l           ;adr lsb
047A'   D9                     exx
047B'   D3 81                  out (proma1),a
047D'   06 32                  ld b,50          ;50*1ms
047F'                          prlp2:
047F'   D9                     exx
0480'   7C                     ld a,h
0481'   D9                     exx
0482'   E6 1F                  and 00011111b ;adr
0484'   F6 80                  or  10000000b
0486'   D3 82                  out (proma2),a
0488'   F6 20                  or  00100000b ;prg
048A'   D3 82                  out (proma2),a
048C'   E6 DF                  and 11011111b
048E'   D3 82                  out (proma2),a
0490'                          prlp3:           ;warten bis ok
0490'   DB 81                  in a,(proma1)
0492'   E6 01                  and 00000001b
0494'   20 FA                  jr nz,prlp3
0496'   10 E7                  djnz prlp2       ;alle 100
                               ; test ob endadr erreicht
0498'   E5                     push hl
0499'   AF                     xor a
049A'   ED 52                  sbc hl,de
049C'   7D                     ld a,l
049D'   B4                     or h
049E'   E1                     pop hl           ;nur flags
049F'   28 06                  jr z,prgfin      ;ende program
04A1'   23                     inc hl
04A2'   D9                     exx
04A3'   23                     inc hl
04A4'   D9                     exx
04A5'   18 CE                  jr prlp1
04A7'                          prgfin:
04A7'   3E 40                  ld a,01000000b   ;nun test ob prg ok
04A9'   D3 82                  out (proma2),a   ;ix,iy gueltig
04AB'   DD E5                  push ix
04AD'   E1                     pop hl
04AE'   FD E5                  push iy
04B0'   D1                     pop de           ;hl start de ende
04B1'   D9                     exx
04B2'   D5                     push de
04B3'   E1                     pop hl
04B4'   D9                     exx
04B5'                          prglp4:          ;leseschleife
04B5'   D9                     exx
04B6'   7D                     ld a,l
04B7'   D3 81                  out (proma1),a
04B9'   7C                     ld a,h
```

Abb. 6.4.5 Listing des Monitorprogramms

```
      MAÇRO-80 3.43   27-Jul-81       PAGE    1-16

04BA'   D9                    exx             ;'reg
04BB'   E6 1F                 and 00011111b
04BD'   F6 40                 or  01000000b
04BF'   D3 82                 out (proma2),a  ;read
04C1'   DB 80                 in a,(promd)    ;holen wert
04C3'   BE                    cp (hl)         ;vergleichen
04C4'   20 19                 jr nz,prgerr    ;prom error **
04C6'   E5                    push hl
04C7'   AF                    xor a
04C8'   ED 52                 sbc hl,de
04CA'   7C                    ld a,h
04CB'   B5                    or l
04CC'   E1                    pop hl
04CD'   28 06                 jr z,prgf1      ;final
04CF'   23                    inc hl
04D0'   D9                    exx
04D1'   23                    inc hl
04D2'   D9                    exx
04D3'   18 E0                jr prglp4
04D5'                        prgf1:
04D5'   21 0412'             ld hl,txtb2     ;burned
04D8'   CD 02ED'             call print
04DB'   CD 0184'             call bell
04DE'   C9                   ret
                             ;
04DF'                        prgerr:
04DF'   21 0419'             ld hl,txtb3     ;error
04E2'   CD 02ED'             call print
04E5'   CD 0184'             call bell
04E8'   C9                   ret

04E9'                        prrd:           ;pr <expr> <expr>
04E9'   3E 40                ld a,01000000b
04EB'   D3 82                out (proma2),a  ;neutr
04ED'   CD 025F'             call expr
04F0'   DA 0399'             jp c,error
04F3'   E5                   push hl
04F4'   CD 025F'             call expr
04F7'   D1                   pop de
04F8'   DA 0399'             jp c,error
04FB'   EB                   ex de,hl        ;hl start de ende
04FC'                        prrlp1:
04FC'   7D                    ld a,l
04FD'   D3 81                 out (proma1),a
04FF'   7C                    ld a,h
0500'   E6 1F                 and 00011111b
0502'   F6 40                 or  01000000b
0504'   D3 82                 out (proma2),a
0506'   DB 80                 in a,(promd)   ;data
0508'   77                    ld (hl),a
0509'   E5                    push hl
050A'   AF                    xor a
050B'   ED 52                 sbc hl,de
050D'   7D                    ld a,l
050E'   B4                    or h
050F'   E1                    pop hl
0510'   28 03                 jr z,prrf      ;final
0512'   23                    inc hl         ;next
0513'   18 E7                jr prrlp1
0515'                        prrf:           ;final
0515'   C9                   ret             ;ok
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-17


                                ;
                                ;
0516'                           fill:           ;f<expr> <expr> <expr>
0516'   CD 025F'                call expr       ;startadresse
0519'   DA 0399'                jp c,error
051C'   E5                      push hl
051D'   DD E1                   pop ix
051F'   CD 025F'                call expr       ;endadresse
0522'   DA 0399'                jp c,error
0525'   E5                      push hl
0526'   FD E1                   pop iy
0528'   CD 025F'                call expr       ;fuellwert
052B'   DA 0399'                jp c,error
052E'   45                      ld b,l          ;merken
052F'   FD E5                   push iy
0531'   E1                      pop hl
0532'   DD E5                   push ix
0534'   D1                      pop de          ;hl end de start
0535'   AF                      xor a
0536'   ED 52                   sbc hl,de       ;differenz 0..n
                                                ;0= 1byte etc
0538'   E5                      push hl         ;merken
0539'   DD E5                   push ix
053B'   E1                      pop hl          ;start
053C'   E5                      push hl
053D'   D1                      pop de
053E'   13                      inc de          ;fuer blocktrans
053F'   78                      ld a,b          ;value holen
0540'   77                      ld (hl),a       ;mindestens einen
0541'   C1                      pop bc          ;laenge
0542'   78                      ld a,b
0543'   B1                      or c            ;=0 ready
0544'   C8                      ret z
0545'   ED B0                   ldir            ;loesche n-1
0547'   C9                      ret             ;fertig



0548'                           move:           ;m<expr> <expr> <expr>
                                                ;  von    bis    nach
0548'   CD 025F'                call expr
054B'   DA 0399'                jp c,error
054E'   E5                      push hl
054F'   DD E1                   pop ix          ;start merken
0551'   CD 025F'                call expr
0554'   DA 0399'                jp c,error
0557'   E5                      push hl
0558'   FD E1                   pop iy          ;ende in iy
055A'   CD 025F'                call expr
055D'   DA 0399'                jp c,error      ;nach in HL z.Z.
0560'   E5                      push hl         ;merken ziel
0561'   FD E5                   push iy
0563'   E1                      pop hl
0564'   DD E5                   push ix
0566'   D1                      pop de          ;ende-anfang = laenge-1
0567'   AF                      xor a
0568'   ED 52                   sbc hl,de       ;differenz
056A'   23                      inc hl          ;+1 dann 1..n
056B'   E5                      push hl
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-18


056C'   C1                      pop bc          ;count blocktrans
056D'   DD E5                   push ix
056F'   E1                      pop hl          ;quelle
0570'   D1                      pop de          ;ziel von stack
0571'   ED B0                   ldir            ;und transport ausf.
0573'   C9                      ret
                                ;



0574'                           read:           ;lesen intelhex-cas
0574'   CD 025F'                call expr       ;GET BIAS fuer offsetload
0577'   DD 21 0000              ld ix,0         ;BIAS in ix
057B'   38 03                   jr c,rd11       ;no bias
057D'   E5                       push hl
057E'   DD E1                    pop ix         ;bias
0580'                           rd11:
0580'   CD 02F7'                call crlf       ;new line
0583'                           rd1:            ;HAUPTSCHLEIFE
0583'   CD 0283'                call rix        ;test ':'
0586'   FE 3A                   cp ':'
0588'   20 F9                   jr nz,rd1       ;warten darauf
                                ; ja gefunden dan start file
058A'   CD 02DE'                call sbyte      ;LAENGE holen
058D'   FE 00                   cp 0            ;=0 dann ende
058F'   CA 05C4'                jp z,rdfin      ;gefunden
0592'   57                      ld d,a          ;CHECKSUM=counter
0593'   47                      ld b,a          ;auch in LOOPCNT
0594'   CD 02DE'                call sbyte      ;bc erhalten
0597'   67                      ld h,a          ;msb
0598'   CD 02DE'                call sbyte      ;lsb
059B'   6F                      ld l,a
059C'   84                      add a,h
059D'   82                      add a,d
059E'   57                      ld d,a          ;neue checksum
059F'   CD 02DE'                call sbyte      ;0=abs
05A2'   FE 00                   cp 0
05A4'   C2 0399'                jp nz,error     ;FILE TYPE ERR
05A7'   C5                      push bc
05A8'   DD E5                   push ix
05AA'   C1                      pop bc
05AB'   09                      add hl,bc       ;+BIAS
05AC'   C1                      pop bc          ;=startadresse
05AD'                           rdlp:           ;DO b,anzah
05AD'   CD 02DE'                 call sbyte     ;ablegen
05B0'   77                       ld (hl),a      ;und dann testen
05B1'   BE                       cp (hl)        ;ungleich dann error
05B2'   C2 05D1'                 jp nz,memerr   ;KEIN SPEICHER DORT
05B5'   82                       add a,d        ;checksum
05B6'   57                       ld d,a
05B7'   23                       inc hl         ;next hl
05B8'   10 F3                   djnz rdlp
05BA'   CD 02DE'                call sbyte      ;checksum
05BD'   82                      add a,d
05BE'   C2 05C8'                jp nz,chkerr    ;checksumerror
05C1'   C3 0583'                jp rd1          ;NEXT wiederholen
                                ;
                                ;
                                ;
```

Abb. 6.4.5 Listing des Monitorprogramms

```
      MACRO-80 3.43   27-Jul-81       PAGE    1-19


05C4'                           rdfin:
05C4'   CD 0184'                call bell       ;melden fertig
05C7'   C9                      ret
                                ;
05C8'                           chkerr:         ;checksum error
05C8'   21 05DA'                ld hl,txtchk
05CB'   CD 02ED'                call print
05CE'   C3 0399'                jp error
                                ;
05D1'                           memerr:         ;MEMORY ERROR
05D1'   21 05E5'                ld hl,txtmem
05D4'   CD 02ED'                call print
05D7'   C3 0399'                jp error
                                ;
05DA'                           txtchk:
05DA'   20 43 48 45             defm ' CHECKSUM '
05DE'   43 4B 53 55
05E2'   4D 20
05E4'   00                      defb 0

05E5'                           txtmem:
05E5'   20 4D 45 4D             defm ' MEMORY '
05E9'   4F 52 59 20
05ED'   00                      defb 0

05EE'                           nulls:          ;punch nulls
05EE'   06 30                   ld b,48
05F0'                           null:
05F0'   C5                       push bc
05F1'   0E 00                    ld c,0
05F3'   CD 0172'                 call poo
05F6'   C1                       pop bc
05F7'   10 F7                   djnz null
05F9'   C9                      ret

05FA'                           write:          ;schreiben intelhex-cas
05FA'   CD 025F'                call expr       ;start
05FD'   DA 0399'                jp c,error
0600'   E5                      push hl         ;retten
0601'   CD 025F'                call expr       ;endadr
0604'   DD E1                   pop ix          ;nach ix
0606'   DA 0399'                jp c,error      ;hl=ende ix=start
                                ;
0609'   E5                      push hl
060A'   DD E5                   push ix
060C'   CD 05EE'                call nulls      ;vornullen
060F'   DD E1                   pop ix
0611'   E1                     pop hl
0612'   E5                      push hl
0613'   FD E1                   pop iy          ;end auch in iy
                                ;
0615'                           wtlp:           ;hauptschleife
0615'   CD 02B8'                 call peol      ;crlf
0618'   0E 3A                    ld c,':'
061A'   CD 0172'                 call poo       ;dann laenge berechnen
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-20

061D'   FD E5                   push iy
061F'   E1                      pop hl
0620'   DD E5                   push ix
0622'   D1                      pop de          ;end-anfang
0623'   AF                      xor a
0624'   ED 52                   sbc hl,de       ;differenz
0626'   7C                      ld a,h          ;muss 0 sein
0627'   06 18                   ld b,24
0629'   B7                      or a
062A'   20 0D                   jr nz,wt5       ;std <>0 dann ok
062C'   11 0017                 ld de,24-1      ;24 bytes max
062F'   AF                      xor a
0630'   ED 52                   sbc hl,de       ;dx-23
0632'   CB 7C                   bit 7,h         ;<0 dann weniger
0634'   28 03                   jr z,wt5
0636'   19                      add hl,de       ;+23 = anzahl
0637'   45                      ld b,l          ;
0638'   04                      inc b           ;1..23
0639'                           wt5:            ;24 . b=counter
0639'   50                      ld d,b          ;CHECKSUM:=start
063A'   78                      ld a,b
063B'   C5                      push bc
063C'   CD 02C7'                call punac      ;ausgabe laenge
063F'   DD E5                   push ix         ;start
0641'   E1                      pop hl
0642'   7D                      ld a,l
0643'   82                      add a,d
0644'   84                      add a,h
0645'   57                      ld d,a
0646'   CD 02C2'                call punhl      ;adr
0649'   3E 00                   ld a,0
064B'   CD 02C7'                call punac      ;type=0
064E'   C1                      pop bc          ;checksum in d
064F'                           wtl1:           ;loop
064F'   C5                       push bc
0650'   DD 7E 00                ld a,(ix)
0653'   82                       add a,d
0654'   57                       ld d,a
0655'   DD 7E 00                ld a,(ix)
0658'   CD 02C7'                 call punac
065B'   DD 23                    inc ix         ;next mem
065D'   C1                       pop bc
065E'   10 EF                   djnz wtl1       ;ausgeben alles
0660'   AF                      xor a
0661'   92                      sub d           ;complement ausgeben
0662'   CD 02C7'                call punac
0665'   DD E5                   push ix
0667'   D1                      pop de
0668'   FD E5                   push iy
066A'   E1                      pop hl
066B'   1B                      dec de          ;-1 wegen inc vorher
066C'   AF                      xor a
066D'   ED 52                   sbc hl,de       ;end-anf
066F'   7D                      ld a,l
0670'   B4                      or h
0671'   C2 0615'               jp nz,wtlp      ;bis alles
                                ;
0674'   CD 02B8'               call peol
0677'   0E 3A                  ld c,':'
0679'   CD 0172'               call poo
067C'   AF                     xor a
```

Abb. 6.4.5 Listing des Monitorprogramms

```
      MACRO-80 3.43   27-Jul-81       PAGE    1-21


067D'   CD 02C7'              call punac      ;0 laenge
0680'   21 0000              ld hl,0
0683'   CD 02C2'              call punhl      ;adresse
0686'   21 0000              ld hl,0         ;type=0
0689'   CD 02C2'              call punhl
068C'   CD 02B8'              call peol
068F'   CD 05EE'              call nulls      ;ende
0692'   CD 0184'              call bell       ;melden
0695'   C9                    ret



                              ;



0696'                         querry:         ;QI eingabe   QO aus I/O Port
0696'   CD 023C'              call echo       ;next zeichen
0699'   FE 49                cp 'I'          ;QI
069B'   28 18                 jr z,qui
069D'   FE 4F                 cp 'O'
069F'   C2 0399'              jp nz,error     ;QO
06A2'   CD 025F'              call expr       ;QO<expr> <expr>
06A5'   DA 0399'              jp c,error      ;fehler
06A8'   4D                    ld c,l
06A9'   C5                    push bc
06AA'   CD 025F'              call expr
06AD'   C1                    pop bc
06AE'   DA 0399'              jp c,error
06B1'   7D                    ld a,l
06B2'   ED 79                 out (c),a       ;ausgeben
06B4'   C9                    ret
                              ;
06B5'                         qui:
06B5'   CD 025F'              call expr       ;QI<expr>
06B8'   0E 20                 ld c,' '
06BA'   CD 011C'              call co
06BD'   4D                    ld c,l
06BE'   ED 78                 in a,(c)        ;laden
06C0'   F5                    push psw
06C1'   CD 02A1'              call prac
06C4'   0E 20                 ld c,' '
06C6'   CD 011C'              call co
06C9'   F1                    pop psw
06CA'   CD 09EB'              call bitaus     ;ausgabe bits
06CD'   C9                    ret
                              ;

                              ;
06CE'                         subst:          ;substitute   eingabe
06CE'   CD 025F'              call expr       ;adresse holen
06D1'   DA 0399'              jp c,error
06D4'                         suul:           ;loop main
06D4'   CD 02F7'              call crlf
06D7'                         suula:          ;innere schleife
06D7'   CD 029C'              call prhl       ;adr ausgeben
06DA'   CD 018A'              call length     ;laenge des befehls
06DD'   C5                    push bc         ;laenge
06DE'   E5                    push hl         ;retten pointer
06DF'                         suul1:
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-22


06DF'   0E 20                    ld c,' '
06E1'   CD 011C'                 call co
06E4'   7E                       ld a,(hl)
06E5'   CD 02A1'                 call prac
06E8'   23                       inc hl          ;
06E9'   10 F4                   djnz suul1
06EB'   D1                      pop de          ;pointer alt
06EC'   C1                      pop bc
06ED'   3E 04                   ld a,4
06EF'   90                      sub b
06F0'   47                      ld b,a          ;berechnen restlaenge
06F1'   07                      rlca
06F2'   E6 0F                   and 0fh         ;*3
06F4'   80                      add a,b
06F5'   47                      ld b,a
06F6'   B7                      or a
06F7'   28 07                   jr z,sunl1      ;no loop
06F9'                            sulp2:
06F9'   0E 20                     ld c,' '
06FB'   CD 011C'                  call co
06FE'   10 F9                    djnz sulp2
0700'                           sunl1:
                                ;
0700'   0E 3A                   ld c,':'        ;ASCII umrahmung
0702'   CD 011C'                call co
0705'   1A                      ld a,(de)
0706'   E6 7F                   and 07fh
0708'   FE 20                   cp ' '
070A'   38 0A                   jr c,snl        ;kein ascii
070C'   FE 7F                   cp 7fh
070E'   28 06                   jr z,snl
0710'   4F                       ld c,a
0711'   CD 011C'                 call co        ;ausgeben ascii
0714'   18 05                   jr snl1
0716'                           snl:
0716'   0E 2E                    ld c,'.'
0718'   CD 011C'                 call co
071B'                           snl1:
071B'   0E 3A                   ld c,':'
071D'   CD 011C'                call co         ; :.:
0720'                           suul2:          ;schleife 2
0720'   E5                      push hl         ;merken alten wert
0721'   CD 025F'                call expr       ;test ob zeichen zahl
0724'   38 0F                   jr c,sunz       ;no zahl
0726'   C1                      pop bc          ;dummy
                                                ;zahl da in hl
0727'   47                      ld b,a          ;retten
0728'   7D                      ld a,l
0729'   12                      ld (de),a       ;abspeichern
072A'   13                      inc de
072B'   78                      ld a,b
072C'   FE 20                   cp ' '
072E'   CA 0720'                jp z,suul2      ;schleife 2
0731'   EB                      ex de,hl        ;alter wert
0732'   C3 06D4'                jp suul         ;next hauptschl.
                                ;
0735'                           sunz:           ;keine zahl in akku wert
0735'   E1                      pop hl          ;alter wert
0736'   FE 08                   cp 8h           ;BS
0738'   CA 0753'                jp z,backgo     ;rueck
073B'   FE 0D                   cp 0dh
```


Abb. 6.4.5 Listing des Monitorprogramms

```
      MACRO-80 3.43   27-Jul-81       PAGE    1-23


073D'   CA 0758'            jp z,lppa       ;next befehl
0740'   FE 20               cp ' '
0742'   CA 06D4'            jp z,suul       ;loop aussen
0745'   FE 22               cp '"'
0747'   CA 075D'            jp z,text       ;texteingabe
074A'   FE 2E               cp '.'
074C'   C2 0399'            jp nz,error
074F'   CD 02F7'            call crlf
0752'   C9                  ret             ;ende
                            ;
0753'                       backgo:
0753'   1B                  dec de
0754'   EB                  ex de,hl
0755'   C3 06D4'            jp suul         ;zurueck
                            ;
0758'                       lppa:
0758'   EB                  ex de,hl        ;alter wert
0759'   23                  inc hl
075A'   C3 06D4'            jp suul         ;hauptschleife
                            ;
075D'                       text:           ;texteingabe
075D'   CD 0101'             call ci        ;alle zeichen nc echo
0760'   FE 08                 cp 08
0762'   28 1C                jr z,back      ;ohne echo
0764'   4F                   ld c,a
0765'   CD 011C'             call co
0768'   FE 22                cp '"'         ;ende terminator
076A'   28 04                jr z,conte     ;exit
076C'   12                   ld (de),a
076D'   13                   inc de
076E'   18 ED               jr text
0770'                       conte:
0770'   CD 0101'            call ci
0773'   4F                  ld c,a
0774'   CD 011C'            call co
0777'   FE 20               cp ' '
0779'   CA 0720'            jp z,suul2
077C'   EB                  ex de,hl
077D'   C3 06D4'            jp suul
                            ;
0780'                       back:
0780'   0E 08               ld c,8          ;ausgabe 8,20h,8
0782'   CD 011C'            call co         ;loeschen des
0785'   0E 20               ld c,' '        ;alten zeichens
0787'   CD 011C'            call co
078A'   0E 08               ld c,8
078C'   CD 011C'            call co
078F'   1B                  dec de
0790'   C3 075D'            jp text         ;neu einsprung
                            ;



0793'                       displa:         ;ausgabe eines bereiches
                                            ;aaaa bb bb ,,, bb  a a  ,,,  a
                                            ;jeweils 8 pro zeile
0793'   CD 025F'            call expr       ;start adresse
0796'   DA 0399'            jp c,error
0799'   E5                  push hl
079A'   DD E1               pop ix          ;merken in ix
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-24


079C'   FD 21 0000          ld iy,0         ;endadresse
07A0'   FE 0D               cp 0dh          ;start
07A2'   28 06               jr z,di1
07A4'   CD 025F'             call expr
07A7'   E5                   push hl
07A8'   FD E1                pop iy
07AA'                       di1:
07AA'   CD 02F7'            call crlf       ;ausgabe vorbereiten
07AD'                       dilp:           ;schleife exit wenn ix=iy
                                            ;oder csts
07AD'   CD 00E5'             call csts
07B0'   FE FF                cp 0ffh
07B2'   20 0B                jr nz,di2
07B4'   CD 0101'              call ci
07B7'   FE 03                 cp 3          ;CTRL C stoppt
07B9'   20 04                 jr nz,di2
07BB'   CD 02F7'              call crlf
07BE'   C9                    ret           ;in main zurueck
07BF'                       di2:
07BF'   DD E5               push ix
07C1'   E1                  pop hl          ;schleifenzaehler
07C2'   CD 029C'            call prhl       ;aaaa
07C5'   06 08               ld b,8          ;hex loop
07C7'                       dilp2:
07C7'   C5                    push bc
07C8'   0E 20                 ld c,' '
07CA'   CD 011C'              call co
07CD'   7E                    ld a,(hl)
07CE'   CD 02A1'              call prac     ;und ausgeben
07D1'   23                    inc hl
07D2'   C1                    pop bc
07D3'   10 F2               djnz dilp2
07D5'   0E 20               ld c,' '
07D7'   CD 011C'            call co
07DA'   0E 20               ld c,' '
07DC'   CD 011C'            call co
07DF'   DD E5               push ix
07E1'   E1                  pop hl
07E2'   06 08               ld b,8
07E4'                       dilp3:
07E4'   C5                    push bc
07E5'   0E 20                 ld c,' '
07E7'   CD 011C'              call co
07EA'   7E                    ld a,(hl)     ;wert bereich test
07EB'   E6 7F                 and 07fh      ;nur untere bits
07ED'   FE 20                 cp ' '
07EF'   38 0A                 jr c,dis1
07F1'   FE 7F                 cp 7fh
07F3'   28 06                 jr z,dis1
07F5'   4F                     ld c,a
07F6'   CD 011C'               call co
07F9'   18 05                 jr dis2
07FB'                         dis1:
07FB'   0E 2E                  ld c,'.'
07FD'   CD 011C'               call co
0800'                         dis2:
0800'   23                    inc hl        ;next adresse
0801'   C1                    pop bc
0802'   10 E0               djnz dilp3
0804'   FD E5               push iy
0806'   D1                  pop de
```


Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43  27-Jul-81        PAGE    1-25


0807'   DD E5                   push ix
0809'   E1                     pop hl          ;untere bits vgl.
080A'   7B                     ld a,e
080B'   E6 F8                  and 11111000b   ;nur alle 8
080D'   5F                     ld e,a
080E'   7D                     ld a,l
080F'   E6 F8                  and 11111000b
0811'   6F                     ld l,a
0812'   AF                     xor a           ;ix-iy=0?
0813'   ED 52                  sbc hl,de
0815'   7D                     ld a,l
0816'   B4                     or h            ;=0
0817'   20 04                  jr nz,di3
0819'   CD 02F7'                call crlf       ;dann ende
081C'   C9                      ret
081D'                          di3:            ;sonst ausgabe
081D'   11 0008                ld de,8         ;8 weiter
0820'   DD 19                  add ix,de
0822'   CD 02F7'                call crlf
0825'   18 86                  jr dilp         ;endloop



0827'                          regist:         ;ausgabe oder aendern
                                               ;der register
0827'   CD 023C'              call echo       ;0dh dann ausgabe
082A'   FE 0D                  cp 0dh
082C'   CA 08B7'              jp z,allaus     ;sonst nur teilbereich
                                               ;name in akku
082F'   5F                     ld e,a          ;merken
0830'   CD 023C'              call echo       ;zweiter teil
0833'   57                     ld d,a          ;nun suchen
0834'   21 0990'              ld hl,reg1
0837'   CD 088C'              call sucre
083A'   30 32                  jr nc,re1
083C'   21 09B5'              ld hl,reg2
083F'   CD 088C'              call sucre
0842'   30 2A                  jr nc,re1
0844'   21 09DA'              ld hl,reg3
0847'   CD 088C'              call sucre
084A'   DA 0399'              jp c,error      ;fehler
084D'   0A                     ld a,(bc)       ;regpaar
084E'   6F                     ld l,a
084F'   03                     inc bc
0850'   0A                     ld a,(bc)
0851'   67                     ld h,a
0852'   0B                     dec bc
0853'   C5                     push bc
0854'   0E 20                  ld c,' '
0856'   CD 011C'              call co
0859'   CD 029C'              call prhl
085C'   0E 2D                  ld c,'-'
085E'   CD 011C'              call co
0861'   CD 025F'              call expr       ;nach hl
0864'   C1                     pop bc
0865'   DA 0399'              jp c,error      ;fehler
0868'   7D                     ld a,l
0869'   02                     ld (bc),a
086A'   03                     inc bc
086B'   7C                     ld a,h
086C'   02                     ld (bc),a
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-26


086D'   C9                      ret             ;ende
                                ;
086E'                           rel:            ;bc -> alter wert
086E'   0A                      ld a,(bc)
086F'   C5                      push bc
0870'   F5                      push psw        ;akku
0871'   0E 20                   ld c,' '
0873'   CD 011C'                call co
0876'   F1                      pop psw         ;datenwert
0877'   CD 02A1'                call prac
087A'   0E 2D                   ld c,'-'
087C'   CD 011C'                call co
087F'   CD 025F'                call expr       ;nach hl
0882'   C1                      pop bc          ;in bc adresse neu
0883'   DA 0399'                jp c,error      ;fehler
0886'   7D                      ld a,l
0887'   02                      ld (bc),a       ;und ablegen dorthin
0888'   CD 02F7'                call crlf
088B'   C9                      ret

088C'                           sucre:          ;sucht in (hl) nach de
                                                ;found dann bc
088C'   E5                      push hl
088D'   DD E1                   pop ix
088F'                           sulp:
088F'   DD 7E 00                 ld a,(ix+0)
0892'   B7                       or a
0893'   28 20                   jr z,ssu1       ;EXIT1
0895'   DD 7E 00                 ld a,(ix+0)
0898'   BB                       cp e
0899'   20 0E                   jr nz,ss2
089B'   DD 7E 01                  ld a,(ix+1)
089E'   BA                        cp d
089F'   20 08                    jr nz,ss2
08A1'   DD 4E 02                  ld c,(ix+2)
08A4'   DD 46 03                  ld b,(ix+3)
08A7'   18 0A                     jr ssu2       ;EXIT2
08A9'                           ss2:
08A9'   DD 23                   inc ix
08AB'   DD 23                   inc ix
08AD'   DD 23                   inc ix
08AF'   DD 23                   inc ix
08B1'   18 DC                  jr sulp
                               ;
08B3'                          ssu2:
08B3'   AF                     xor a
08B4'   C9                     ret             ;found

08B5'                          ssu1:           ;fehler nc found
08B5'   37                     scf
08B6'   C9                     ret

08B7'                          allaus:         ;register ausgeben
08B7'   CD 02F7'               call crlf
08BA'   21 0990'               ld hl,reg1
08BD'   CD 08DF'               call ausub
08C0'   21 0990'               ld hl,reg1
08C3'   CD 08FA'               call auswt
08C6'   21 09B5'               ld hl,reg2
08C9'   CD 08DF'               call ausub
08CC'   21 09B5'               ld hl,reg2
```


Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-27


08CF'   CD 08FA'            call auswt
08D2'   21 09DA'            ld hl,reg3
08D5'   CD 090F'            call ausub2
08D8'   21 09DA'            ld hl,reg3
08DB'   CD 094C'            call auswt2
08DE'   C9                  ret

08DF'                       ausub:          ;hl-> tabelle 0=ende
08DF'   7E                  ld a,(hl)
08E0'   B7                  or a
08E1'   28 13               jr z,au1        ;ende
08E3'   4F                   ld c,a
08E4'   CD 011C'             call co
08E7'   23                   inc hl
08E8'   4E                   ld c,(hl)
08E9'   CD 011C'             call co
08EC'   0E 20                ld c,' '
08EE'   CD 011C'             call co
08F1'   23                   inc hl
08F2'   23                   inc hl
08F3'   23                   inc hl
08F4'   18 E9               jr ausub
08F6'                       au1:
08F6'   CD 02F7'            call crlf
08F9'   C9                  ret

08FA'                       auswt:
08FA'   7E                  ld a,(hl)
08FB'   B7                  or a
08FC'   28 F8               jr z,au1        ;mit verwenden
08FE'   23                   inc hl
08FF'   23                   inc hl         ;auf pointer
0900'   5E                   ld e,(hl)
0901'   23                   inc hl
0902'   56                   ld d,(hl)
0903'   23                   inc hl
0904'   1A                   ld a,(de)      ;holen
0905'   CD 02A1'             call prac
0908'   0E 20                ld c,' '
090A'   CD 011C'             call co
090D'   18 EB               jr auswt

090F'                       ausub2:
090F'   E5                  push hl
0910'   21 0A07'            ld hl,txtbit    ;ausgabe bitstream
0913'   CD 02ED'            call print
0916'   0E 20               ld c,' '
0918'   CD 011C'            call co
091B'   E1                  pop hl
091C'                       auslp2:         ;loop
091C'   7E                  ld a,(hl)
091D'   B7                  or a
091E'   28 1D               jr z,au2        ;ende
0920'   4F                   ld c,a
0921'   CD 011C'             call co
0924'   23                   inc hl
0925'   4E                   ld c,(hl)
0926'   CD 011C'             call co
0929'   0E 20                ld c,' '
092B'   CD 011C'             call co
092E'   0E 20                ld c,' '
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-28


0930'   CD 011C'              call co
0933'   0E 20                 ld c,' '
0935'   CD 011C'              call co         ;mehr leerplatz
0938'   23                    inc hl
0939'   23                    inc hl
093A'   23                    inc hl
093B'   18 DF                jr auslp2
093D'                         au2:
093D'   0E 20                 ld c,' '
093F'   CD 011C'              call co
0942'   21 0A10'              ld hl,txtpcw    ;inhalt pc
0945'   CD 02ED'              call print
0948'   CD 02F7'              call crlf
094B'   C9                    ret

094C'                         auswt2:
094C'   E5                    push hl
094D'   3A 130C               ld a,(afsto)
0950'   CD 09EB'              call bitaus     ;nur akku
0953'   0E 20                 ld c,' '
0955'   CD 011C'              call co
0958'   E1                    pop hl
0959'                         auslp3:         ;loop
0959'   7E                    ld a,(hl)
095A'   B7                    or a
095B'   CA 0975'              jp z,au11       ;crlf
095E'   23                    inc hl
095F'   23                    inc hl          ;auf pointer
0960'   4E                    ld c,(hl)
0961'   23                    inc hl
0962'   46                    ld b,(hl)
0963'   23                    inc hl
0964'   0A                    ld a,(bc)       ;holen
0965'   5F                    ld e,a
0966'   03                    inc bc
0967'   0A                    ld a,(bc)
0968'   57                    ld d,a
0969'   EB                    ex de,hl
096A'   CD 029C'              call prhl
096D'   EB                    ex de,hl
096E'   0E 20                 ld c,' '
0970'   CD 011C'              call co
0973'   18 E4                jr auslp3       ;loop
0975'                         au11:
0975'   0E 20                 ld c,' '
0977'   CD 011C'              call co
097A'   2A 1328               ld hl,(pcsto)   ;holen befehl
097D'   CD 018A'              call length     ;laenge in b
                                              ;hl bleibt
0980'                         au12:           ;schleife
0980'   0E 20                 ld c,' '
0982'   CD 011C'              call co
0985'   7E                    ld a,(hl)       ;wert
0986'   CD 02A1'              call prac       ;bc bleibt
0989'   23                    inc hl
098A'   10 F4                 djnz au12
098C'   CD 02F7'              call crlf
098F'   C9                    ret
                              ;

                              ;
```


Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-29


0990'                           reg1:           ;liste registername,adresse
0990'   41 20                   defm 'A '       ;einfache register
0992'   130D                   defw afsto+1    ;msb
0994'   46 20                   defm 'F '
0996'   130C                   defw afsto
0998'   42 20                   defm 'B '
099A'   130B                   defw bcsto+1
099C'   43 20                   defm 'C '
099E'   130A                   defw bcsto
09A0'   44 20                   defm 'D '
09A2'   1309                   defw desto+1
09A4'   45 20                   defm 'E '
09A6'   1308                   defw desto
09A8'   48 20                   defm 'H '
09AA'   1307                   defw hlsto+1
09AC'   4C 20                   defm 'L '
09AE'   1306                   defw hlsto
09B0'   49 20                   defm 'I '
09B2'   1314                   defw isto
09B4'   00                     defb 0

09B5'                           reg2:           ;'register
09B5'   41 27                   defm "A'"       ;einfache register
09B7'   131D                   defw afssto+1   ;msb
09B9'   46 27                   defm "F'"
09BB'   131C                   defw afssto
09BD'   42 27                   defm "B'"
09BF'   131B                   defw bcssto+1
09C1'   43 27                   defm "C'"
09C3'   131A                   defw bcssto
09C5'   44 27                   defm "D'"
09C7'   1319                   defw dessto+1
09C9'   45 27                   defm "E'"
09CB'   1318                   defw dessto
09CD'   48 27                   defm "H'"
09CF'   1317                   defw hlssto+1
09D1'   4C 27                   defm "L'"
09D3'   1316                   defw hlssto
09D5'   52 20                   defm 'R '
09D7'   1312                   defw rsto
09D9'   00                     defb 0

09DA'                           reg3:
09DA'   53 50                   defm 'SP'
09DC'   131E                   defw spsto
09DE'   49 58                   defm 'IX'
09E0'   130E                   defw ixsto
09E2'   49 59                   defm 'IY'
09E4'   1310                   defw iysto
09E6'   50 43                   defm 'PC'
09E8'   1328                   defw pcsto
09EA'   00                     defb 0

09EB'                           bitaus:         ;akku als bitstream
                                                ;MSB .. LSB ausgeben
09EB'   06 08                   ld b,8          ;8 bits
09ED'                           bi1:            ;DO B,8
09ED'   07                       rlca
09EE'   38 0B                    jr c,biei1     ;eins
09F0'   F5                         push psw
09F1'   C5                         push bc
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-30


09F2'   0E 30                    ld c,'0'       ;NULL
09F4'   CD 011C'                 call co
09F7'   C1                       pop bc
09F8'   F1                       pop psw
09F9'   18 09                   jr bi2
09FB'                           bieil:
09FB'   F5                       push psw
09FC'   C5                       push bc
09FD'   0E 31                    ld c,'1'       ;EINS
09FF'   CD 011C'                 call co
0A02'   C1                       pop bc
0A03'   F1                       pop psw
0A04'                           bi2:
0A04'   10 E7                   djnz bi1        ;ENDDO
0A06'   C9                      ret

0A07'                           txtbit:
0A07'   53 5A 78 48             defm 'SZxHxPNC'
0A0B'   78 50 4E 43
0A0F'   00                      defb 0

0A10'                           txtpcw:
0A10'   28 50 43 29             defm '(PC)'
0A14'   00                      defb 0


0A15'                           goto:           ;sprung zu einem
                                                ;anwenderprogramm
0A15'   CD 025F'                call expr       ;sprungadresse
0A18'   30 03                   jr nc, gt1
0A1A'   2A 1328                  ld hl,(pcsto)  ;alten pc verwenden
0A1D'                           gt1:
0A1D'   FE 0D                   cp 0dh
0A1F'   28 05                   jr z,gt3
0A21'   FE 2C                   cp ','          ;kein komma dann err
0A23'   C2 0399'                jp nz,error
0A26'                           gt3:
0A26'   E5                      push hl
0A27'   FE 0D                   cp 0dh          ;endezeichen
0A29'   28 2F                   jr z,g1         ;bei brk JMP einbau
0A2B'   3E C3                    ld a,0c3h
0A2D'   32 0038                  ld (38h),a
0A30'   21 0A98'                 ld hl,rstart
0A33'   22 0039                  ld (38h+1),hl
0A36'   CD 025F'                 call expr
0A39'   38 1F                    jr c,g1        ;ende
0A3B'   F5                       push psw       ;merke terminator
0A3C'   22 1320                  ld (brk1),hl
0A3F'   7E                       ld a,(hl)
0A40'   32 1324                  ld (ins1),a
0A43'   3E FF                    ld a,0ffh      ;RST 7
0A45'   77                       ld (hl),a
0A46'   F1                       pop psw        ;terminator
0A47'   FE 0D                    cp 0dh
0A49'   28 0F                    jr z,g1        ;end sonst weiter
0A4B'   CD 025F'                 call expr      ;immer ende
0A4E'   38 0A                    jr c,g1
0A50'   22 1322                  ld (brk2),hl
0A53'   7E                       ld a,(hl)
0A54'   32 1326                  ld (ins2),a
0A57'   3E FF                    ld a,0ffh
```


Abb. 6.4.5 Listing des Monitorprogramms

```
      MACRO-80 3.43   27-Jul-81       PAGE    1-31


0A59'   77                    ld (hl),a
0A5A'                         g1:
0A5A'   E1                    pop hl
                              ; goex ausfuehren
0A5B'                         goex:           ;sprung aufuehren
                                              ;sprungziel in HL
                                              ;wird auf usr stack gebracht
                                              ;dann mit ret springen
0A5B'   3A 1314               ld a,(isto)
0A5E'   ED 47                 ld i,a
0A60'   3A 1312               ld a,(rsto)
0A63'   ED 4F                 ld r,a
0A65'   D9                    exx
0A66'   08                    ex af,af'
0A67'   2A 131C               ld hl,(afssto)
0A6A'   E5                    push hl
0A6B'   F1                    pop psw
0A6C'   2A 1316               ld hl,(hlssto)
0A6F'   ED 5B 1318            ld de,(dessto)
0A73'   ED 4B 131A            ld bc,(bcssto)
0A77'   D9                    exx
0A78'   08                    ex af,af'       ;in hl sprung adr
0A79'   ED 5B 130C            ld de,(afsto)
0A7D'   D5                    push de
0A7E'   F1                    pop psw
0A7F'   ED 7B 131E            ld sp,(spsto)   ;neuer stack
0A83'   E5                    push hl         ;ret adresse
0A84'   2A 1306               ld hl,(hlsto)
0A87'   ED 5B 1308            ld de,(desto)
0A8B'   ED 4B 130A            ld bc,(bcsto)
0A8F'   DD 2A 130E            ld ix,(ixsto)
0A93'   FD 2A 1310            ld iy,(iysto)
0A97'   C9                    ret             ;SPRUNG AUF DAS AUSF.PRG.


                              ; bei breakpoint erfolgt hier der einsprung

0A98'                         rstart:         ;register retten
0A98'   22 1306               ld (hlsto),hl
0A9B'   ED 53 1308            ld (desto),de
0A9F'   ED 43 130A            ld (bcsto),bc
0AA3'   DD 22 130E            ld (ixsto),ix
0AA7'   FD 22 1310            ld (iysto),iy
0AAB'   E1                    pop hl          ;RET-ADRESSE
0AAC'   2B                    dec hl          ;RST -1 =adr
0AAD'   ED 73 131E            ld (spsto),sp   ;aktueller stand
0AB1'   22 1328               ld (pcsto),hl   ;und ablegen
0AB4'   31 13FF               ld sp,stack
0AB7'   F5                    push psw
0AB8'   E1                    pop hl
0AB9'   22 130C               ld (afsto),hl
0ABC'   D9                    exx
0ABD'   08                    ex af,af'
0ABE'   22 1316               ld (hlssto),hl
0AC1'   ED 53 1318            ld (dessto),de
0AC5'   ED 43 131A            ld (bcssto),bc
0AC9'   F5                    push psw
0ACA'   E1                    pop hl
0ACB'   22 131C               ld (afssto),hl
0ACE'   D9                    exx
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-32


0ACF'   08                      ex af,af'
0AD0'   ED 57                   ld a,i
0AD2'   32 1314                 ld (isto),a
0AD5'   ED 5F                   ld a,r
0AD7'   32 1312                 ld (rsto),a
0ADA'   3A 1324                 ld a,(ins1)     ;flag testen
0ADD'   FE FF                   cp 0ffh
0ADF'   28 04                   jr z,rs1
0AE1'   2A 1320                  ld hl,(brk1)   ;adresse holen
0AE4'   77                       ld (hl),a      ;alter wert zur.
0AE5'                           rs1:
0AE5'   3A 1326                 ld a,(ins2)
0AE8'   FE FF                   cp 0ffh
0AEA'   28 04                   jr z,rs2
0AEC'   2A 1322                  ld hl,(brk2)   ;adresse holen
0AEF'   77                       ld (hl),a
0AF0'                           rs2:
0AF0'   CD 02F7'                call crlf
0AF3'   21 0B08'                ld hl,txtbrk
0AF6'   CD 02ED'                call print
0AF9'   2A 1328                 ld hl,(pcsto)
0AFC'   CD 029C'                call prhl
0AFF'   CD 02F7'                call crlf       ;meldung ausgeben
0B02'   CD 0184'                call bell       ;melden fertig
0B05'   C3 033D'                jp lpm

0B08'                           txtbrk:
0B08'   2D 42 52 45             defm '-BREAK->'
0B0C'   41 4B 2D 3E
0B10'   00                      defb 0

0B11'                           ioset:
0B11'   C9                      ret


0B12'                           iochk:
0B12'   AF                      xor a
0B13'   C9                      ret

                                ; CRT-Controler Interface-Routinen
                                ;
                                ; wait wartet bc mal 12 ys+15 = 27ys
0B14'                           wait:           ;2 +1
0B14'   CB 38                   srl b           ;msb
0B16'   CB 19                   rr c            ;lsb
0B18'   03                      inc bc          ;savety
0B19'                           wait1:
0B19'   DB FD                    in a,(usta)    ;11->6.5ys
0B1B'   E6 08                    and 00001000b  ;test uart 3.5
0B1D'   C4 0B27'                 call nz,eintrg ;aufruf dann 5
0B20'   0B                       dec bc         ;rest 12ys
0B21'   78                       ld a,b
0B22'   B1                       or c
0B23'   C2 0B19'                jp nz,wait1
0B26'   C9                      ret

                                ;
0B27'                           eintrg:         ;eintrag in ringbuffer
0B27'   C5                      push bc
0B28'   CD 0108'                call ci1
0B2B'   4F                      ld c,a
```


Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-33


0B2C'   CD 0B81'            call put        ;wenn in warteschleife
0B2F'   C1                  pop bc
0B30'   C9                  ret             ;daten auftreten
                            ;  mit test csts routine zeiten verschoben
                            ;  aber in sichere richtung
                            ;
                            ; exec schreibt ein cmdwort nach crt
                            ; data in a , cmd in c

0B31'                       exec:
0B31'   D3 60               out (crtdat),a
0B33'   79                  ld a,c
0B34'   F6 80               or 10000000b    ;st geben
0B36'   D3 61               out (crtctr),a
0B38'   E6 7F               and 01111111b
0B3A'   D3 61               out (crtctr),a
0B3C'   F6 80               or 10000000b
0B3E'   D3 61               out (crtctr),a  ;___--___
0B40'   C9                  ret

                            ; zeichenausgabe des wertes in register c
                            ; ohne bereichspruefung

0B41'                       c93ch:
0B41'   79                  ld a,c          ;wert holen
0B42'   C5                  push bc
0B43'   0E 0F               ld c,0fh        ;daten SCHREIBEN
0B45'                       cbein:          ;zweiter eingang
0B45'   CD 0B31'            call exec       ;und ablegen
0B48'   01 000A             ld bc,130/12    ;delay noetig
0B4B'   CD 0B14'            call wait
0B4E'   3A 132C             ld a,(posx)     ;test noetig
0B51'   FE 3F               cp 63           ;letzte spalte
0B53'   C2 0B81'            jp nz,c1
0B56'   3A 132E             ld a,(posy)
0B59'   FE 0F               cp 15
0B5B'   C2 0B78'            jp nz,c2        ;letzte zeile
0B5E'   01 02B3             ld bc,8300/12   ;scroll
0B61'   CD 0B14'            call wait
0B64'   3E 20               ld a,' '        ;dann loeschen
0B66'   0E 0D               ld c,1101b      ;letzte reihe
0B68'   CD 0B31'            call exec       ;da falscher wert
0B6B'   01 02B3             ld bc,8300/12   ;nochmals warten
0B6E'   CD 0B14'            call wait
0B71'   3E 00               ld a,0
0B73'   32 132C             ld (posx),a     ;neue pos
0B76'   C1                  pop bc
0B77'   C9                  ret
                            ;
0B78'                       c2:
0B78'   3A 132E             ld a,(posy)
0B7B'   3C                  inc a           ;erhoehung
0B7C'   E6 0F               and 00001111b
0B7E'   32 132E             ld (posy),a
0B81'                       c1:
0B81'   3A 132C             ld a,(posx)
0B84'   3C                  inc a
0B85'   E6 3F               and 00111111b
0B87'   32 132C             ld (posx),a     ;auch neu
0B8A'   C1                  pop bc          ;ende
0B8B'   C9                  ret
```

Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-34


                                ;

                                ; eigentliche zeicheninterpreterroutine
                                ; befehle
                                ;
                                ; 0dh = CR  0ch,1ah = clear  0bh = cursor up
                                ; 0ah = LF  8 = cursor left    9 = cursor right
                                ;  5 = home

0B8C'                           c93co:
0B8C'   79                      ld a,c          ;analyse des zeichens
0B8D'   FE 20                   cp ' '          ;lesbares zeichen ?
0B8F'   DA 0B95'                jp c,steu       ;nein dann steuerzeichen
0B92'   C3 0B41'                jp c93ch        ;sonst ausgeben
                                ;
0B95'                           steu:           ;Steuerzeichenbehandlung

0B95'   FE 0D                   cp 0dh          ;CR
0B97'   20 13                   jr nz,steu1     ;nein dann weiter
0B99'   C5                       push bc
0B9A'   0E 01                    ld c,1         ;kein write return
0B9C'   CD 0B31'                 call exec
0B9F'   01 015E                  ld bc,4200/12  ;warten
0BA2'   CD 0B14'                 call wait
0BA5'   3E 00                    ld a,0
0BA7'   32 132C                  ld (posx),a    ;neue pos
0BAA'   C1                       pop bc
0BAB'   C9                       ret
                                ;
0BAC'                           steu1:
0BAC'   FE 0A                   cp 0ah          ;LF
0BAE'   20 1D                   jr nz,steu2
0BB0'   C5                       push bc
0BB1'   3E 20                    ld a,' '       ;bei scroll loeschen
0BB3'   0E 0A                    ld c,1010b     ;mit blanks
0BB5'   CD 0B31'                 call exec
0BB8'   01 02B3                  ld bc,8300/12  ;warten
0BBB'   CD 0B14'                 call wait
0BBE'   3A 132E                  ld a,(posy)
0BC1'   FE 0F                    cp 15
0BC3'   28 06                    jr z,srr1      ;dann ok
0BC5'   3C                        inc a
0BC6'   E6 0F                     and 00001111b ;savety
0BC8'   32 132E                   ld (posy),a   ;sonst +1
0BCB'                            srr1:
0BCB'   C1                       pop bc
0BCC'   C9                       ret
                                ;
0BCD'                           steu2:
0BCD'   FE 0C                   cp 0ch
0BCF'   28 04                   jr z,s1         ;weiter ok
0BD1'   FE 1A                    cp 1ah         ;auch CLEAR
0BD3'   20 18                    jr nz,steu3    ;nein
0BD5'                           s1:
0BD5'   C5                       push bc
0BD6'   3E 20                    ld a,' '       ;leerzeichen loe
0BD8'   0E 08                    ld c,1000b
0BDA'   CD 0B31'                 call exec
0BDD'   01 2AF8                  ld bc,11000    ;132ms/12
0BE0'   CD 0B14'                 call wait
0BE3'   3E 00                    ld a,0
```


Abb. 6.4.5 Listing des Monitorprogramms

```
MACRO-80 3.43   27-Jul-81       PAGE    1-35


0BE5'   32 132C                  ld (posx),a
0BE8'   32 132E                  ld (posy),a     ;normieren
0BEB'   C1                       pop bc
0BEC'   C9                       ret
                                 ;
0BED'                            steu3:
0BED'   FE 08                    cp 8            ;CUR LEFT <-
0BEF'   20 26                    jr nz,steu4
0BF1'   C5                        push bc
0BF2'   0E 04                     ld c,0100b
0BF4'   CD 0B31'                  call exec
0BF7'   01 0006                   ld bc,80/12
0BFA'   CD 0B14'                  call wait
0BFD'   3A 132C                   ld a,(posx)
0C00'   FE 00                     cp 0
0C02'   20 0B                     jr nz,s11
0C04'   3A 132E                    ld a,(posy)
0C07'   3D                         dec a
0C08'   E6 0F                      and 00001111b ;pos updaten
0C0A'   32 132E                    ld (posy),a   ;cirkular
0C0D'   3E 00                      ld a,0
0C0F'                             s11:
0C0F'   3D                        dec a
0C10'   E6 3F                     and 00111111b  ;auch x
0C12'   32 132C                   ld (posx),a
0C15'   C1                        pop bc
0C16'   C9                        ret
                                 ;
0C17'                            steu4:
0C17'   FE 09                    cp 9            ;CUR RIGHT  ->
0C19'   20 08                    jr nz,steu5
0C1B'   C5                        push bc
0C1C'   0E 07                     ld c,0111b     ;wie char aus
0C1E'   3E 20                     ld a,' '       ;ohne write
0C20'   C3 0B45'                  jp cbein       ;extra eingang
                                 ;
0C23'                            steu5:
0C23'   FE 0B                    cp 0bh          ;cursor UP
0C25'   20 17                    jr nz,steu6
0C27'   C5                        push bc
0C28'   0E 06                     ld c,0110b     ;up code
0C2A'   CD 0B31'                  call exec
0C2D'   01 0006                   ld bc,80/12
0C30'   CD 0B14'                  call wait
0C33'   3A 132E                   ld a,(posy)    ;neue pos
0C36'   3D                        dec a
0C37'   E6 0F                     and 00001111b
0C39'   32 132E                   ld (posy),a    ;y
0C3C'   C1                        pop bc
0C3D'   C9                        ret
                                 ;
0C3E'                            steu6:
0C3E'   FE 05                    cp 5            ;HOME
0C40'   C0                       ret nz          ;IGNORE REST
0C41'   C5                        push bc
0C42'   0E 00                     ld c,0
0C44'   CD 0B31'                  call exec
0C47'   01 2AF8                   ld bc,11000    ;wie clear
0C4A'   CD 0B14'                  call wait
0C4D'   3E 00                     ld a,0
0C4F'   32 132C                   ld (posx),a
```

Abb. 6.4.5 Listing des Monitorprogramms

```
        MACRO-80 3.43   27-Jul-81        PAGE    1-36


0C52'   32 132E                 ld (posy),a
0C55'   C1                      pop bc
0C56'   C9                      ret
                                ;


                                ; ram bereich und definitionen

                                ;


1300                            regsave equ     endram-0ffh
1300                            rst30jmp equ    regsave ;benutzer spruenge dort
1303                            nmijmp  equ     rst30jmp+3
1306                            hlsto   equ     nmijmp+3
1308                            desto   equ     hlsto+2
130A                            bcsto   equ     desto+2
130C                            afsto   equ     bcsto+2
130E                            ixsto   equ     afsto+2
1310                            iysto   equ     ixsto+2
1312                            rsto    equ     iysto+2
1314                            isto    equ     rsto+2
1316                            hlssto  equ     isto+2
1318                            dessto  equ     hlssto+2
131A                            bcssto  equ     dessto+2
131C                            afssto  equ     bcssto+2
131E                            spsto   equ     afssto+2
1320                            brk1    equ     spsto+2
1322                            brk2    equ     brk1+2  ;breakpoint merker
1324                            ins1    equ     brk2+2  ;merker instrukt
1326                            ins2    equ     ins1+2
1328                            pcsto   equ     ins2+2  ;pc bei brk
132A                            keyboa  equ     pcsto+2 ;status keys MERKER
132C                            posx    equ     keyboa+2
132E                            posy    equ     posx+2  ;merker fuer CRT
                                ;
13FF                            stack   equ     endram  ;stack rueckwaerts


                                end
```

Abb. 6.4.5 Listing des Monitorprogramms

Im Listing auf Seite 1–2 *Abb. 6.4.5* ist die Bedeutung der einzelnen Bit dieses Schalters eingezeichnet. Mit den Positionen 0 bis 3 kann die Baudrate des seriellen Interface programmiert werden, falls es vorhanden ist, Bit 5 muß dazu auf 0 sein, ist es auf 1, wird als Baudrate für das serielle Interface 9600 Baud verwendet. Damit ist ohne die Karte KEY 9600 Baud festgelegt. Ist Bit 7 auf 0, so ist die CRT-Karte vorhanden und die Ausgabe des Rechners geht auf die CRT-Karte. Ist Bit 6 auf 0, so wird die Tastatur der Karte KEY verwendet und alle Eingaben von Befehlen werden von daher genommen. Ist Bit 4 auf 0, so wird die Ausgabe von Abspeicher und Ladebefehlen nicht über die Kassettenschnittstelle,

```
        MACRO-80 3.43   27-Jul-81       PAGE    S

Macros:

Symbols:
131C    AFSSTO          130C    AFSTO           02B4'   AK1
08B7'   ALLAUS          08F6'   AU1             0975'   AU11
0980'   AU12            093D'   AU2             0099'   AUSC1
009F'   AUSCR           091C'   AUSLP2          0959'   AUSLP3
00A8'   AUSUA           08DF'   AUSUB           090F'   AUSUB2
08FA'   AUSWT           094C'   AUSWT2          0239'   B1
0238'   B2              0237'   B3              0236'   B4
0780'   BACK            0753'   BACKGO          131A    BCSSTO
130A    BCSTO           0184'   BELL            09ED'   BI1
0A04'   BI2             09FB'   BIEI1           09EB'   BITAUS
1320    BRK1            1322    BRK2            0B81'   C1
0B78'   C2              0B41'   C93CH           0B8C'   C93CO
0158'   CASINIT         0B45'   CBEIN           05C8'   CHKERR
0101'   CI              0108'   CI1             0111'   CI2
00CA    CMDCAS          011C'   CO              0123'   CO1
012D'   CO2             0069'   CONINIT         0770'   CONTE
02F7'   CRLF            0061    CRTCTR          0060    CRTDAT
00E5'   CSTS            00EC'   CSTS1           00CB    DATCAS
1318    DESSTO          1308    DESTO           07AA'   DI1
07BF'   DI2             081D'   DI3             07AD'   DILP
07C7'   DILP2           07E4'   DILP3           07FB'   DIS1
0800'   DIS2            0793'   DISPLA          025C'   EC1
023C'   ECHO            0B27'   EINTRG          13FF    ENDRAM
1300    ENDUSR          0399'   ERROR           026E'   EX1
027F'   EX2             0B31'   EXEC            025F'   EXPR
0516'   FILL            0A5A'   G1              00C8'   GET
00D4'   GET1            0A5B'   GOEX            0A15'   GOTO
0A1D'   GT1             0A26'   GT3             1316    HLSSTO
1306    HLSTO           1324    INS1            1326    INS2
0B12'   IOCHK           0B11'   IOSET           1314    ISTO
130E    IXSTO           1310    IYSTO           00FD'   KB1
0111'   KBDIN           00F5'   KBDS            132A    KEYBOA
0068    KEYD            0069    KEYS            018A'   LENGTH
013D'   LI              0146'   LO              014E'   LO1
0134'   LOINIT          033D'   LPM             0347'   LPM1
0758'   LPPA            003E'   MAXI1           0041'   MAXI2
0393'   MEMCK           05D1'   MEMERR          003B'   MINI
0000    MONLOC          0548'   MOVE            028C'   NIBBLE
1303    NMIJMP          05F0'   NUL1            05EE'   NULLS
02AA'   OUTH            02DA'   PAK1            1328    PCSTO
02B8'   PEOL            0172'   POO             017A'   POO1
132C    POSX            132E    POSY            02A1'   PRAC
04DF'   PRGERR          04D5'   PRGF1           04A7'   PRGFIN
04B5'   PRGLP4          029C'   PRHL            02ED'   PRINT
0475'   PRLP1           047F'   PRLP2           0490'   PRLP3
0081    PROMA1          0082    PROMA2          0080    PROMO
0421'   PROMER          04E9'   PRRD            0515'   PRRF
04FC'   PRRLP1          0455'   PRS1            02C7'   PUNAC
02C2'   PUNHL           00B1'   PUT             02D0'   PUTH
0696'   QUERRY          06B5'   QUI             0583'   RD1
0580'   RD11            05C4'   RDFIN           05AD'   RDLP
086E'   RE1             0574'   READ            0990'   REG1
09B5'   REG2            09DA'   REG3            0827'   REGIST
1300    REGSAVE         0161'   RI              0169'   RI1
0289'   RIBBLE          11FF    RINGCT          11FE    RINGLS
1200    RINGPO          0283'   RIX             0AE5'   RS1
0AF0'   RS2             1300    RST30JMP        0A98'   RSTART
1312    RSTO            0BD5'   S1              0C0F'   S11
02DE'   SBYTE           0716'   SNL             071B'   SNL1
```

Abb. 6.4.5 Listing des Monitorprogramms

```
          MACRO-80 3.43   27-Jul-81        PAGE    S-1

131E    SPSTO           0BCB'   SRR1            08A9'   SS2
08B5'   SSU1            08B3'   SSU2            13FF    STACK
031B'   START           0B95'   STEU            0BAC'   STEU1
0BCD'   STEU2           0BED'   STEU3           0C17'   STEU4
0C23'   STEU5           0C3E'   STEU6           06CE'   SUBST
088C'   SUCRE            088F'   SULP            06F9'   SULP2
0700'   SUNL1            0735'   SUNZ            06D4'   SUUL
06DF'   SUUL1           0720'   SUUL2           06D7'   SUULA
01EE'   TAB2            01FA'   TAB3            01FD'   TAB4
0200'   TAB5            035F'   TBL             0088'   TERMINAL
075D'   TEXT            0302'   TXT1            0316'   TXT2
0407'   TXTB1           0412'   TXTB2           0419'   TXTB3
0A07'   TXTBIT          0B08'   TXTBRK          05DA'   TXTCHK
05E5'   TXTMEM          0A10'   TXTPCW          00FE    UCMD
00EE    UCMD2           00FF    UCON            00EF    UCON2
00FC    UDAT            00EC    UDAT2           00FD    USTA
00EB    USTA2           0403'   VERFIN          039F'   VERIFY
03B7'   VERLP           03EF'   VERSK           0B14'   WAIT
0B19'   WAIT1           05FA'   WRITE           0639'   WT5
064F'   WTL1            0615'   WTLP

No Fatal error(s)
```

Abb. 6.4.5 Listing des Monitorprogramms

sondern über die Consolschnittstelle gegeben. Liegt das Bit auf 1, so wird das Kassetteninterface zum Abspeichern und Laden von Programmen verwendet.
Zwei Kombinationen sind wichtig:

alle Positionen auf 1: 11111111

Aus- und Eingabe über die Karte SER (Datensichtgerät daran anschließen) und 9600 Baud, Kassetteninterface verwendbar

CRT-Verwendung: 00111111b

Ausgabe mit Hilfe der CRT-Karte und Eingabe über die an der Karte KEY angeschlossene Tastatur.

Das Monitorprogramm ist etwas über 3K Bytes lang und startet auf Adresse 0. Wird eine Version auf einer anderen Adresse gewünscht, so müssen alle mit dem Zeichen „'" gekennzeichneten Adressen geändert werden. Dies ist auch ein Grund dafür, warum der Assembler die Adreßteile nicht im LOW-HIGH-Format ausgibt, sondern in der lesbaren Form. Um die Eingabe des Monitorprogramms zu erleichtern, ist zusätzlich in *Abb. 6.4.6* ein Listing im HEX-Code abgedruckt. Auf der rechten Spalte sind Prüfsummen abgedruckt, die es erlauben, die eingegebenen Werte zu überprüfen. Dazu wird jeweils die Summe über alle Daten einer Zeile gebildet.

Nach dem Start des Monitorprogramms meldet sich dieses auf der Console mit der Überschrift „RDK MONITOR 1.0". Dann wird die Sequenz „–>" ausgegeben und der Monitor wartet auf einen Befehl.

Die Befehle werden in alphabetischer Reihenfolge behandelt: Als Erklärung für die Syntax der Befehle werden folgende Begriffe verwendet. Die Sequenz „CR" steht für die Eingabe der Taste RETURN, die auch als Wagenrücklauf bekannt ist. Mit „adresse" ist die

Eingabe eines 16-Bit-Wertes gemeint und mit „byte" die Eingabe einer 8-Bit-Größe. Dabei kann eine Bezeichnung wie „start" oder „end" vorangestellt sein. Groß geschriebene Buchstaben werden direkt so eingegeben wie dargestellt (außer CR) und Leerzeichen oder Kommas müssen auch so wie angegeben eingegeben werden.

Die Sequenz CTRL- . . . steht für die gleichzeitige Betätigung der Taste CTRL und des Zeichens, was danach angegeben wurde.

1. D-Display: Ausgabe von Speicherbereichen

Syntax: DstartadresseˌendadresseCR
Es wird der Bereich von „startadresse" bis „endadresse" ausgegeben. Dabei werden die Werte sowohl in hexadezimaler (sedezimal) und in der Textdarstellung in Gruppen zu 8 Bytes ausgegeben. Die Ausgabe kann auch jederzeit durch Eingabe des Zeichens CTRL-C abgebrochen werden. Beispiel:
D1000,13FFCR gibt einen 1K Bereich, beginnend bei der Adresse 1000h aus. Anstelle des Kommas kann auch ein Leerzeichen stehen.

2. F-Fill: Füllen eines Speicherbereichs mit einem Wert

Syntax: FstartadresseˌendadresseˌwertbyteCR
Der Bereich von „startadresse" bis „endadresse" wird mit dem Wert von „wertbyte" gefüllt. Es handelt sich um einen Befehl, der ein Speicherfeld mit einem Wert vorbelegen kann. Beispiel:
F1000,1200,0CR. Damit wird der Bereich 1000h bis einschließlich 1200h auf 0 gesetzt.

3. G-Goto: Starten eines Programms

Syntax: GstartadresseCR
Gstartadresse,break1adresseCR
Gstartadresse,break1adresse,break2adresseCR

Dieser Befehl besitzt drei Formen. In der ersten Form wird einfach auf die Adresse „startadresse" gesprungen und das dort stehende Programm ausgeführt. Im zweiten Fall kann eine zusätzliche Adresse angegeben werden, ein sogenannter BREAKPOINT. Wird die Adresse bei der Ausführung des Programms angesprochen, so erfolgt ein Stop des Programms. Bei der dritten Form können zwei solcher Adressen angegeben werden. Um die Ausführung zu stoppen, wird an der Stelle des BREAKPOINTS der Befehl RST 7 (Code 0FFh) geschrieben. Der Befehlscode, der ursprünglich an dieser Stelle stand, wird vom Monitorprogramm gerettet. Wird nun dieser Befehl ausgeführt, so erfolgt ein Unterprogrammsprung auf Adressen 38h. Dort steht ein Sprung auf die Routine RESTART. Es werden alle Register gerettet. Nun kann durch Monitorbefehle der Registerinhalt angesehen werden. Der Monitor schreibt zuvor noch die Originalbefehle wieder an die Breakpoint-Adresse zurück. Beispiel:

Im Speicher steht folgender Code:
1200h: 3E 01 3C 3E 03
Es wird der Befehl G1200,1203CR gegeben. Danach steht im Register A der Wert 02.

Denn das Programm lautete:

```
LD A,1
INC A
LD A,3
```

Auf den Befehl LD A,3 wurde ein Breakpoint gesetzt. Der Breakpoint darf immer nur an den Anfang eines Befehls gesetzt werden, wenn der Befehl aus mehreren Bytes besteht.

4. M-Move: Transport von Speicherbereichen

Syntax: Mstartadresse,endadresse,zieladresseCR
Mit dem Befehl kann ein Speicherbereich kopiert werden. Dabei wird der Bereich von „startadresse" bis „endadresse" nach „zieladresse" transportiert. Es ist darauf zu achten, daß sich die Bereiche nicht überlappen, da sonst unter Umständen der Transport nicht korrekt durchgeführt wird.

5. N-Null: Ausgabe des NUL-Zeichens

Syntax: NCR
Es werden eine Reihe von binären Nullen (ASCII-NUL, Code 00h) auf den Kassettenrecorder gegeben. Dieser Befehl kann für Synchronisationstests verwendet werden.

6. P-Prom: Programmieren eines Eproms und Auslesen

Syntax: PWstartadresse,endadresseCR
PRstartadresse,endadresseCR
Mit dem ersten Befehl wird ein EPROM auf der Promprogrammiererkarte programmiert. Nach Eingabe der Start- und Endadresse, fragt das Programm: „PRG YES=Y". An dieser Stelle wird das EPROM in die Fassung gesteckt und dann Y eingegeben. Der Programmiervorgang erfolgt nun. Nach ein paar Minuten wird bei einer erfolgreichen Programmierung entweder „BURNED" ausgegeben oder „PRG ERR", wenn ein Fehler vorlag. Mit dem Befehl „PR" kann ein EPROM eingelesen werden. Dazu wird Start- und Endadresse angegeben. Das Auslesen und auch Programmieren erfolgt stets von Adresse 0 des EPROMs. Beim Programmieren wie auch Auslesen ist darauf zu achten, daß der DIL-Stecker auf der Karte für den richtigen EPROM-Typ ausgewählt wurde.

7. Q-Querry: IO-Ports bearbeiten

Syntax: QIportbyteCR
QOportbyte,databyteCR
Beim oberen Befehl erfolgt die Ausgabe des am Port anstehenden Wertes in HEX und in BINÄRER Form. Mit dem unteren Befehl ist es möglich, einen Wert auf ein Port auszugeben.
Beispiel: QI10CR gibt den am Port 10h anstehenden Wert aus
QO20,55CR gibt den Wert 55h an den Port 20h aus.

8. R-Read: Lesen von Daten

Syntax: RCR
RbiasadresseCR

Mit diesem Befehl kann ein Speicherdump vom Kassettenrecorder (oder seriellem Port, je nach DIL-Schalter auf Karte KEY) in den Speicher eingelesen werden. Die Aufzeichnung erfolgt dabei im Intel-Hex-Format (siehe Kommentare im Listing des Monitors) und enthält eine Startadresse. Wird der Wert „biasadresse" angegeben, so wird diese Adresse zur eigentlichen Startadresse addiert. Damit können abgespeicherte Programme auch auf andere Adressen geladen werden.

Beispiel: R1000CR

Wurde ein Programm vom Bereich 0 bis 1FFh abgespeichert, so wird es mit diesem Befehl auf den Bereich 1000h bis 11FFh geladen. Da das Intel-Hex-Format eine Prüfsumme enthält, kann bei einem Lesefehler die Meldung „CHECKSUM" erfolgen. Ebenfalls möglich ist die Fehlermeldung „MEMORY", die angibt, daß ein Fehler beim Einschreiben in den Speicher erfolgt ist, vielleicht, weil an der betreffenden Stelle gar kein Speicher war.

9. S-Substitute: Modifizieren von Speicherzellen

Syntax: SstartadresseCR

Nach diesem Befehl wird ein gesonderter Programmteil angesprungen. Dort gibt es eine Reihe von Befehlen:

Zeichen " ", Leerzeichen
Zeichen """", Anführungszeichen
Zeichen CTRL-H, Backstep
Zeichen ".", Punkt
Zeichen CR, Wagenrücklauf

Begonnen wird mit der Ausgabe der eingegebenen „startadresse" und des Inhalts der Speicherzelle an diesem Platz. Dabei wird aber der Inhalt der Speicherzelle als 200-Befehl verstanden und mit der richtigen Befehlslänge dargestellt. Ebenfalls wird eine Textinterpretation durchgeführt und das Zeichen ausgegeben. Beispiel:

```
1000    C3  55  01      :C:
```

Nun kann ein weiterer Befehl eingegeben werden. Wird ein Leerzeichen eingegeben, so wird der nächste Befehl dargestellt, als z. B.:

```
1000    C3  55  01      :C:
1003    3E  3C          :>:
```

Wird anstelle dessen nur ein CR eingegeben, so wird die um eins erhöhte Adresse angezeigt. Beispiel:

```
1000    C3  55  01      :C:
1003    3E  3C          :>:
1004    3C              :<:
```

Mit dem Befehl CTRL-H oder Backstep kann ein Schritt zurück gegangen werden. Die Ausgabe ist dann:

```
1000    C3  55  01      :C:
1003    3E  3C          :>:
1004    3C              :<:
1003    3E  3C          :>:
```

Nun kommt das wichtigste, die Eingabe von Daten. Es werden dazu einfach die Hexwerte angegeben. Soll zum Beispiel die Sequenz 3E 01 D3 55 eingespeichert werden, so kann dies wie folgt geschehen:

```
1000    C3  55  01      :C:3E   01  D3   55
```

Die Sequenz wird mit einem CR abgeschlossen und die nächste freie Speicherzelle wird ausgegeben. Es kann auch nach jedem einzelnen Byte schon ein CR eingegeben werden, dies ist beliebig. Wird eine Hex-Zahl eingegeben, so kann die nächste entweder nach einem Leerzeichen oder nach einem CR eingegeben werden. Nur die Ausgabe auf dem Bildschirm ist unterschiedlich. Eine andere Möglichkeit ist die Eingabe von Texten. Dazu können beliebige ASCII-Zeichen folgen und das ganze wird wieder mit einem Anführungszeichen abgeschlossen. Mit dem Zeichen CTRL-H oder Backstep kann der letzte Eintrag gelöscht werden. Der Cursor des Sichtgerätes fährt dazu ein Zeichen zurück und löscht auch das auf dem Bildschirm stehende Zeichen. Wird bei der Eingabe im HEX-Mode ein falscher Wert gegeben, und wurde noch kein Leerzeichen oder Cr eingegeben, so genügt es, den korrekten Wert dahinterzuschreiben. Beispiel:

2125454EF5A ergibt den Wert 5A, da nur die letzten beiden Zahlen ausgewertet werden.

Mit dem Befehl „.“, also der Eingabe eines Punktes kann der Substitute-Befehl wieder verlassen werden.

Zur Längenbestimmung der Z80-Befehle befindet sich im Monitor ein recht nützliches Unterprogramm mit den Namen LENGTH. Es erhält als Parameter im Registerpaar HL die Adresse, bei der der zu untersuchende Befehl steht. Im B-Register steht dann nach dem Aufruf die Anzahl der Bytes, die der Befehl belegt.

10. T-Terminal: Betrieb als Datensichtgerät

Syntax: TCR

Bei Angabe dieses Befehls wird der Computer als Datensichtgerät verwendet. Dazu muß er die CRT-Karte, die Karte KEY sowie das serielle Interface SER besitzen. Eine Eingabe über SER wird dann auf dem Bildschirm ausgegeben und eine Eingabe auf der Tastatur wird auf die serielle Leitung gegeben. Das Terminal versteht dabei ein paar Sonderzeichen.

CR	Code: 0Dh	Cursor nach links zum Zeilenanfang
LF	Code: 0Ah	Zeilenvorschub
BS	Code: 08h	Ein Zeichen nach links
FF	Code: 0Ch	Löschen des Bildschirms

HT Code: 09h Ein Zeichen nach rechts
VT Code: 0Bh Ein Zeichen nachoben
ENQ Code: 05h Cursor in die obere linke Ecke (HOME)

Der Bereich 20h bis 7Fh wird als ASCII-Zeichen ausgegeben.

11. V-Verify: Vergleichen von Speicherbereichen

Syntax: Vstartadresse, endadresse, mitadresse CR

Der Bereich „startadresse" bis „endadresse" wird mit „mitadresse" verglichen. Stimmt ein Wert nicht überein, so wird die Adresse von „startbereich" angegeben, sowie der Wert dort und der entsprechende Wert im „mitbereich". Ebenfalls erfolgt die Ausgabe der unterschiedlichen Bits in Binär. Dazu werden die beiden Werte mit Exclusiv-Oder verknüpft.

Beispiel: V1000, 1011, 2200 CR
Ausgabe z. B.:

1000 55 50 00000101
1005 10 20 00110000

12. W-Write: Abspeichern von Daten

Syntax: Wsartadresse,endadresse CR
Der Bereich „startadresse" bis „endadresse" wird auf Kassette ausgegeben. Die Ausgabe erfolgt im Intel-Hex-Format dabei wird darin auch eine Prüfsumme abgelegt. Das Format ist aus dem Listing ersichtlich.

13. X-Xamine: Register verändern und ausgeben

Syntax: XCR
Xregister
Mit der ersten Form wird der Inhalt aller Register ausgegeben, beim zweiten Fall kann ein beliebiger Registername angegeben werden, der dann modifiziert werden kann. Als Registernamen sind folgende zugelassen: A_ F_ B_ C_ E_ F_ H_ L_ IX IY SP PC A' F' B' C' D' E' H' L' I_ R_ wobei das Zeichen „_" für ein Leerzeichen steht. Danach wird der alte Wert ausgegben und ein neuer kann mit CR abgeschlossen, eingegeben werden.

Beispiel: XH 50-41 CR

Der alte Wert 50h des Registers H wird in den neuen Wert 41h gewandelt.
Abb. 6.4.6 zeigt den Hex-Dump des Monitors.

```
-------------- Seite  1  ------- File MONITOR -----------
rom  abs                                                  checksum
0000  C3 1B 03 C3 01 01 C3 61 01 C3 1C 01 C3 72 01 C3      += 05A4
0010  46 01 C3 E5 00 C3 12 0B C3 11 0B C3 93 03 C3 98      += 0662
0020  0A C3 3B 00 C3 3E 00 C3 41 00 00 00 00 00 00 00      += 030D
0030  C3 00 13 00 00 00 00 00 C3 98 0A C3 98 0A C3 98      += 04FB
0040  0A C3 98 0A 00 00 83 5F C9 CD 69 24 7D CD 30 0F      += 05FD
0050  E6 02 17 17 17 83 5F C9 CD 69 24 7D CD 4C 0F E6      += 06BD
0060  07 6F 7B CD 4C 0F C3 03 13 3E 1E D3 FF 3E 0B D3      += 063C
0070  FE 0E 0C CD 8C 0B DB 69 00 00 00 32 2A 13 CB 6F      += 0569
0080  C0 E6 0F F6 10 D3 FF C9 CD EC 00 20 12 CD F5 00      += 0903
0090  20 16 CD C8 00 30 02 18 EF 4F CD 2D 01 18 E9 CD      += 061C
00A0  08 01 4F CD B1 00 18 E0 CD 11 01 4F CD 23 01 18      += 0505
00B0  D7 E5 C5 21 00 12 3A FF 11 4F 06 00 09 C1 71 3A      += 05C8
00C0  FF 11 3C 32 FF 11 E1 C9 3A FF 11 47 3A FE 11 B8      += 07CA
00D0  20 02 37 C9 E5 21 00 12 4F 3C 32 FE 11 06 00 09      += 0415
00E0  7E E1 37 3F C9 3A 2A 13 CB 77 28 09 DB FD E6 08      += 074E
00F0  C8 3E FF B7 C9 DB 68 CB 7F 28 02 AF C9 3E FF B7      += 09A8
0100  C9 3A 2A 13 CB 77 28 09 DB FD E6 08 28 FA DB FC      += 0872
0110  C9 DB 68 CB 7F 20 FA F5 DB 69 F1 C9 3A 2A 13 CB      += 09A5
0120  7F 28 0A DB FD E6 10 28 FA 79 D3 FC C9 C5 CD 8C      += 09D0
0130  0B C1 79 C9 3E 1E D3 EF 3E 0B D3 EE C9 DB ED E6      += 09AD
0140  08 28 FA DB EC C9 3A 2A 13 CB 67 CA 23 01 DB ED      += 0819
0150  E6 10 28 FA 79 D3 EC C9 3E 53 D3 CA 3E 50 D3 CA      += 0972
0160  C9 3A 2A 13 CB 67 CA 08 01 DB CA E6 01 28 FA DB      += 07CE
0170  CB C9 3A 2A 13 CB 67 CA 23 01 DB CA E6 02 28 FA      += 07DA
0180  79 D3 CB C9 0E 07 CD 1C 01 C9 06 00 E5 D1 7E E6      += 07C8
0190  DF FE 0D CA 00 02 7E FE CB CA FA 01 FE ED CA EE      += 0B35
01A0  01 7E FE C3 CA 37 02 FE CD CA 37 02 E6 EF FE 22      += 0906
01B0  CA 37 02 FE 2A CA 37 02 E6 CF FE 01 CA 37 02 E6      += 07CB
01C0  C7 FE C2 CA 37 02 FE C4 CA 37 02 7E E6 F7 FE 10      += 09B8
01D0  CA 38 02 FE D3 CA 38 02 E6 E7 FE 20 CA 38 02 E6      += 08AE
01E0  C7 FE 06 CA 38 02 FE C6 CA 38 02 C3 39 02 23 7E      += 0736
01F0  E6 C7 FE 43 CA 36 02 C3 38 02 C3 38 02 C3 36 02      += 06E5
0200  23 7E FE CB CA 36 02 FE 21 CA 36 02 E6 FE FE 34      += 08A3
0210  CA 37 02 E6 F8 FE 70 CA 37 02 7E E6 CF FE 06 CA      += 0953
0220  36 02 E6 C7 FE 02 CA 36 02 7E D6 40 E6 87 FE 06      += 07EC
0230  CA 37 02 C3 38 02 04 04 04 04 EB C9 CD 01 01 E6      += 0579
0240  7F 3C F8 3D C8 FE 4E C8 FE 6E 28 10 FE 0D C8 C5      += 0908
0250  4F CD 1C 01 79 C1 FE 40 D8 FE 7B D0 E6 5F C9 21      += 0901
0260  00 00 CD 3C 02 47 CD 8C 02 78 30 02 37 C9 47 CD      += 056B
0270  8C 02 38 0B 29 29 29 29 B5 6F CD 3C 02 18 EF 78      += 0523
0280  37 3F C9 CD 61 01 E6 7F C9 CD 83 02 D6 30 D8 FE      += 08CA
0290  17 3F D8 FE 0A 3F D0 D6 07 FE 0A C9 7C CD A1 02      += 07DF
02A0  7D F5 1F 1F 1F 1F CD AA 02 F1 E6 0F C6 30 FE 3A      += 077B
02B0  38 02 C6 07 4F C3 1C 01 0E 0D CD 72 01 0E 0A C3      += 046C
02C0  72 01 7C CD C7 02 7D F5 1F 1F 1F 1F CD D0 02 F1      += 0703
02D0  E6 0F C6 30 FE 3A 38 02 C6 07 4F C3 72 01 C5 CD      += 0741
02E0  89 02 07 07 07 07 4F CD 89 02 B1 C1 C9 7E B7 C8      += 0686
02F0  4F CD 1C 01 23 18 F6 0E 0D CD 1C 01 0E 0A CD 1C      += 0470
0300  01 C9 0D 0A 52 44 4B 2D 4D 6F 6E 69 74 6F 72 20      += 04F7
```

Abb. 6.4.6 Hexdump des Monitorprogramms

```
-------------- Seite  2  ------- File MONITOR -----------
rom  abs                                                  checksum
0310  31 2E 30 0D 0A 00 0D 0A 2D 3E 00 31 FF 13 CD 69        += 03A1
0320  00 CD 34 01 CD 58 01 3E 00 32 FE 11 32 FF 11 21        += 050A
0330  02 03 CD ED 02 3E FF 32 24 13 32 26 13 11 3D 03        += 0423
0340  D5 21 16 03 CD ED 02 CD 3C 02 28 FB D6 41 F8 FE        += 0806
0350  1A D0 87 06 00 4F 21 5F 03 09 5E 23 56 EB E9 99        += 0596
0360  03 99 03 99 03 93 07 99 03 16 05 15 0A 99 03 99        += 03E0
0370  03 99 03 99 03 99 03 48 05 EE 05 99 03 21 04 96        += 046E
0380  06 74 05 CE 06 88 00 99 03 9F 03 FA 05 27 08 99        += 04E0
0390  03 99 03 21 00 13 44 7D C9 0E 2A CD 1C 01 C9 CD        += 0515
03A0  5F 02 DA 99 03 E5 DD E1 CD 5F 02 DA 99 03 E5 FD        += 0900
03B0  E1 CD 5F 02 DA 99 03 DD 7E 00 BE 28 32 E5 DD E5        += 089F
03C0  FD E5 CD F7 02 DD E5 D1 EB E5 CD 9C 02 E1 0E 20        += 0A85
03D0  CD 1C 01 7E CD A1 02 0E 20 CD 1C 01 1A CD A1 02        += 057A
03E0  0E 20 CD 1C 01 1A AE CD EB 09 FD E1 DD E1 E1 E5        += 0903
03F0  DD E5 D1 FD E5 E1 AF ED 52 7D B4 E1 28 05 DD 23        += 0A83
0400  23 18 B4 CD F7 02 C9 50 52 47 20 59 45 53 3D 59        += 060E
0410  20 00 42 55 52 4E 45 44 00 50 52 47 20 45 52 52        += 03D2
0420  00 CD 3C 02 FE 52 CA E9 04 FE 57 C2 99 03 CD 5F        += 07F1
0430  02 DA 99 03 E5 CD 5F 02 DD E1 DA 99 03 E5 FD E1        += 0982
0440  D9 21 00 00 D9 FE 0D 28 0C CD 5F 02 DA 99 03 E5        += 069B
0450  D9 E1 E5 D1 D9 3E 40 D3 82 CD F7 02 21 07 04 CD        += 08DB
0460  ED 02 CD 3C 02 FE 59 C2 99 03 CD F7 02 DD E5 E1        += 0918
0470  FD E5 D1 3E 80 7E D3 80 D9 7D D9 D3 81 06 32 D9        += 09D6
0480  7C D9 E6 1F F6 80 D3 82 F6 20 D3 82 E6 DF D3 82        += 0AAA
0490  DB 81 E6 01 20 FA 10 E7 E5 AF ED 52 7D B4 E1 28        += 0961
04A0  06 23 D9 23 D9 18 CE 3E 40 D3 82 DD E5 E1 FD E5        += 093C
04B0  D1 D9 D5 E1 D9 D9 7D D3 81 7C D9 E6 1F F6 40 D3        += 0B46
04C0  82 DB 80 BE 20 19 E5 AF ED 52 7C B5 E1 28 06 23        += 080A
04D0  D9 23 D9 18 E0 21 12 04 CD ED 02 CD 84 01 C9 21        += 06FC
04E0  19 04 CD ED 02 CD 84 01 C9 3E 40 D3 82 CD 5F 02        += 06F5
04F0  DA 99 03 E5 CD 5F 02 D1 DA 99 03 EB 7D D3 81 7C        += 0908
0500  E6 1F F6 40 D3 82 DB 80 77 E5 AF ED 52 7D B4 E1        += 0A47
0510  28 03 23 18 E7 C9 CD 5F 02 DA 99 03 E5 DD E1 CD        += 082A
0520  5F 02 DA 99 03 E5 FD E1 CD 5F 02 DA 99 03 45 FD        += 0880
0530  E5 E1 DD E5 01 AF ED 52 E5 DD E5 E1 E5 D1 13 78        += 0C10
0540  77 C1 78 B1 C8 ED B0 C9 CD 5F 02 DA 99 03 E5 DD        += 09F5
0550  E1 CD 5F 02 DA 99 03 E5 FD E1 CD 5F 02 DA 99 03        += 08EC
0560  E5 FD E5 E1 DD E5 D1 AF ED 52 23 E5 C1 DD E5 E1        += 0C95
0570  01 ED B0 C9 CD 5F 02 DD 21 00 00 38 03 E5 DD E1        += 0841
0580  CD F7 02 CD 83 02 FE 3A 20 F9 CD DE 02 FE 00 CA        += 08DE
0590  C4 05 57 47 CD DE 02 67 CD DE 02 6F 84 82 57 CD        += 07C1
05A0  DE 02 FE 00 C2 99 03 C5 DD E5 C1 09 C1 CD DE 02        += 08FB
05B0  77 BE C2 D1 05 82 57 23 10 F3 CD DE 02 82 C2 C8        += 0885
05C0  05 C3 83 05 CD 84 01 C9 21 DA 05 CD ED 02 C3 99        += 0783
05D0  03 21 E5 05 CD ED 02 C3 99 03 20 43 48 45 43 4B        += 05A7
05E0  53 55 4D 20 00 20 4D 45 4D 4F 52 59 20 00 06 30        += 0364
05F0  C5 0E 00 CD 72 01 C1 10 F7 C9 CD 5F 02 DA 99 03        += 0748
0600  E5 CD 5F 02 DD E1 DA 99 03 E5 DD E5 CD EE 05 DD        += 0A8B
0610  E1 E1 E5 FD E1 CD B8 02 0E 3A CD 72 01 FD E5 E1        += 0A57
```

Abb. 6.4.6 Hexdump des Monitorprogramms

```
-------------- Seite  3  ------- File MONITOR -----------
rom  abs                                                  checksum
0620  DD E5 D1 AF ED 52 7C 06 18 B7 20 0D 11 17 00 AF       += 06D6
0630  ED 52 CB 7C 28 03 19 45 04 50 78 C5 CD C7 02 DD       += 0713
0640  E5 E1 7D 82 84 57 CD C2 02 3E 00 CD C7 02 C1 C5       += 088B
0650  DD 7E 00 82 57 DD 7E 00 CD C7 02 DD 23 C1 10 EF       += 07E5
0660  AF 92 CD C7 02 DD E5 D1 FD E5 E1 1B AF ED 52 7D       += 0AB3
0670  B4 C2 15 06 CD B8 02 0E 3A CD 72 01 AF CD C7 02       += 06E5
0680  21 00 00 CD C2 02 21 00 00 CD C2 02 CD B8 02 CD       += 05B8
0690  EE 05 CD 84 01 C9 CD 3C 02 FE 49 28 18 FE 4F C2       += 07AF
06A0  99 03 CD 5F 02 DA 99 03 4D C5 CD 5F 02 C1 DA 99       += 07B4
06B0  03 7D ED 79 C9 CD 5F 02 0E 20 CD 1C 01 4D ED 78       += 06A7
06C0  F5 CD A1 02 0E 20 CD 1C 01 F1 CD EB 09 C9 CD 5F       += 0824
06D0  02 DA 99 03 CD F7 02 CD 9C 02 CD 8A 01 C5 E5 0E       += 07B9
06E0  20 CD 1C 01 7E CD A1 02 23 10 F4 D1 C1 3E 04 90       += 0683
06F0  47 07 E6 0F 80 47 B7 28 07 0E 20 CD 1C 01 10 F9       += 0511
0700  0E 3A CD 1C 01 1A E6 7F FE 20 38 0A FE 7F 28 06       += 05BC
0710  4F CD 1C 01 18 05 0E 2E CD 1C 01 0E 3A CD 1C 01       += 03AE
0720  E5 CD 5F 02 38 0F C1 47 7D 12 13 78 FE 20 CA 20       += 0684
0730  07 EB C3 D4 06 E1 FE 08 CA 53 07 FE 0D CA 58 07       += 07CE
0740  FE 20 CA D4 06 FE 22 CA 5D 07 FE 2E C2 99 03 CD       += 0867
0750  F7 02 C9 1B EB C3 D4 06 EB 23 C3 D4 06 CD 01 01       += 07DF
0760  FE 08 28 1C 4F CD 1C 01 FE 22 28 04 12 13 18 ED       += 04F9
0770  CD 01 01 4F CD 1C 01 FE 20 CA 20 07 EB C3 D4 06       += 069F
0780  0E 08 CD 1C 01 0E 20 CD 1C 01 0E 08 CD 1C 01 1B       += 0333
0790  C3 5D 07 CD 5F 02 DA 99 03 E5 DD E1 FD 21 00 00       += 078C
07A0  FE 0D 28 06 CD 5F 02 E5 FD E1 CD F7 02 CD E5 00       += 08A2
07B0  FE FF 20 08 CD 01 01 FE 03 20 04 CD F7 02 C9 DD       += 0788
07C0  E5 E1 CD 9C 02 06 08 C5 0E 20 CD 1C 01 7E CD A1       += 0708
07D0  02 23 C1 10 F2 0E 20 CD 1C 01 0E 20 CD 1C 01 DD       += 04F5
07E0  E5 E1 06 08 C5 0E 20 CD 1C 01 7E E6 7F FE 20 38       += 06EA
07F0  0A FE 7F 28 06 4F CD 1C 01 18 05 0E 2E CD 1C 01       += 0431
0800  23 C1 10 E0 FD E5 D1 DD E5 E1 7B E6 F8 5F 7D E6       += 0B45
0810  F8 6F AF ED 52 7D B4 20 04 CD F7 02 C9 11 08 00       += 0752
0820  DD 19 CD F7 02 18 86 CD 3C 02 FE 0D CA B7 08 5F       += 0758
0830  CD 3C 02 57 21 90 09 CD 8C 08 30 32 21 B5 09 CD       += 0588
0840  8C 08 30 2A 21 DA 09 CD 8C 08 DA 99 03 0A 6F 03       += 0545
0850  0A 67 0B C5 0E 20 CD 1C 01 CD 9C 02 0E 20 CD 1C       += 04E8
0860  01 CD 5F 02 C1 DA 99 03 7D 02 03 7C 02 C9 0A C5       += 05FE
0870  F5 0E 20 CD 1C 01 F1 CD A1 02 0E 20 CD 1C 01 CD       += 0660
0880  5F 02 C1 DA 99 03 7D 02 CD F7 02 C9 E5 DD E1 DD       += 0926
0890  7E 00 B7 28 20 DD 7E 00 BB 20 0E DD 7E 01 BA 20       += 05F7
08A0  08 DD 4E 02 DD 46 03 18 0A DD 23 DD 23 DD 23 DD       += 065A
08B0  23 18 DC AF C9 37 C9 CD F7 02 21 90 09 CD 0F 08       += 07C3
08C0  21 90 09 CD FA 08 21 B5 09 CD 0F 08 21 B5 09 CD       += 06C8
08D0  FA 08 21 DA 09 CD 0F 09 21 DA 09 CD 4C 09 C9 7E       += 0658
08E0  B7 28 13 4F CD 1C 01 23 4E CD 1C 01 0E 20 CD 1C       += 049D
08F0  01 23 23 23 18 E9 CD F7 02 C9 7E B7 28 F8 23 23       += 0695
0900  5E 23 56 23 1A CD A1 02 0E 20 CD 1C 01 18 EB E5       += 0584
0910  21 07 0A CD ED 02 0E 20 CD 1C 01 E1 7E B7 28 1D       += 0561
0920  4F CD 1C 01 23 4E CD 1C 01 0E 20 CD 1C 01 0E 20       += 03DA
```

Abb. 6.4.6 Hexdump des Monitorprogramms

```
------------- Seite  4  ------- File MONITOR -----------
rom  abs                                                  checksum
0930  CD 1C 01 0E 20 CD 1C 01 23 23 23 18 DF 0E 20 CD      += 045D
0940  1C 01 21 10 0A CD ED 02 CD F7 02 C9 E5 3A 0C 13      += 05E1
0950  CD EB 09 0E 20 CD 1C 01 E1 7E B7 CA 75 09 23 23      += 067D
0960  4E 23 46 23 0A 5F 03 0A 57 EB CD 9C 02 EB 0E 20      += 0516
0970  CD 1C 01 18 E4 0E 20 CD 1C 01 2A 28 13 CD 8A 01      += 04BB
0980  0E 20 CD 1C 01 7E CD A1 02 23 10 F4 CD F7 02 C9      += 06BC
0990  41 20 0D 13 46 20 0C 13 42 20 0B 13 43 20 0A 13      += 0206
09A0  44 20 09 13 45 20 08 13 48 20 07 13 4C 20 06 13      += 0207
09B0  49 20 14 13 00 41 27 1D 13 46 27 1C 13 42 27 18      += 0248
09C0  13 43 27 1A 13 44 27 19 13 45 27 18 13 48 27 17      += 025E
09D0  13 4C 27 16 13 52 20 12 13 00 53 50 1E 13 49 58      += 02BB
09E0  0E 13 49 59 10 13 50 43 28 13 00 06 08 07 38 0B      += 020C
09F0  F5 C5 0E 30 CD 1C 01 C1 F1 18 09 F5 C5 0E 31 CD      += 077B
0A00  1C 01 C1 F1 10 E7 C9 53 5A 78 48 78 50 4E 43 00      += 0655
0A10  28 50 43 29 00 CD 5F 02 30 03 2A 28 13 FE 0D 28      += 03DD
0A20  05 FE 2C C2 99 03 E5 FE 0D 28 2F 3E C3 32 38 00      += 063F
0A30  21 98 0A 22 39 00 CD 5F 02 38 1F F5 22 20 13 7E      += 046B
0A40  32 24 13 3E FF 77 F1 FE 0D 28 0F CD 5F 02 38 0A      += 05C0
0A50  22 22 13 7E 32 26 13 3E FF 77 E1 3A 14 13 ED 47      += 056A
0A60  3A 12 13 ED 4F D9 08 2A 1C 13 E5 F1 2A 16 13 ED      += 05EB
0A70  5B 18 13 ED 4B 1A 13 D9 08 ED 5B 0C 13 D5 F1 ED      += 06E6
0A80  7B 1E 13 E5 2A 06 13 ED 5B 08 13 ED 4B 0A 13 DD      += 0569
0A90  2A 0E 13 FD 2A 10 13 C9 22 06 13 ED 53 08 13 ED      += 04E1
0AA0  43 0A 13 DD 22 0E 13 FD 22 10 13 E1 2B ED 73 1E      += 054C
0AB0  13 22 28 13 31 FF 13 F5 E1 22 0C 13 D9 08 22 16      += 04E3
0AC0  13 ED 53 18 13 ED 43 1A 13 F5 E1 22 1C 13 D9 08      += 05E3
0AD0  ED 57 32 14 13 ED 5F 32 12 13 3A 24 13 FE FF 28      += 05D6
0AE0  04 2A 20 13 77 3A 26 13 FE FF 28 04 2A 22 13 77      += 044A
0AF0  CD F7 02 21 08 0B CD ED 02 2A 28 13 CD 9C 02 CD      += 0653
0B00  F7 02 CD 84 01 C3 3D 03 2D 42 52 45 41 4B 2D 3E      += 054B
0B10  00 C9 AF C9 CB 38 CB 19 03 DB FD E6 08 C4 27 0B      += 07E7
0B20  0B 78 B1 C2 19 0B C9 C5 CD 08 01 4F CD B1 00 C1      += 070C
0B30  C9 D3 60 79 F6 80 D3 61 E6 7F D3 61 F6 80 D3 61      += 0A62
0B40  C9 79 C5 0E 0F CD 31 0B 01 0A 00 CD 14 0B 3A 2C      += 048A
0B50  13 FE 3F C2 81 0B 3A 2E 13 FE 0F C2 78 0B 01 B3      += 061F
0B60  02 CD 14 0B 3E 20 0E 0D CD 31 0B 01 B3 02 CD 14      += 0407
0B70  0B 3E 00 32 2C 13 C1 C9 3A 2E 13 3C E6 0F 32 2E      += 0450
0B80  13 3A 2C 13 3C E6 3F 32 2C 13 C1 C9 79 FE 20 DA      += 0659
0B90  95 0B C3 41 0B FE 0D 20 13 C5 0E 01 CD 31 0B 01      += 04CB
0BA0  5E 01 CD 14 0B 3E 00 32 2C 13 C1 C9 FE 0A 20 1D      += 04C9
0BB0  C5 3E 20 0E 0A CD 31 0B 01 B3 02 CD 14 0B 3A 2E      += 044E
0BC0  13 FE 0F 28 06 3C E6 0F 32 2E 13 C1 C9 FE 0C 28      += 05AE
0BD0  04 FE 1A 20 18 C5 3E 20 0E 08 CD 31 0B 01 F8 2A      += 04B9
0BE0  CD 14 0B 3E 00 32 2C 13 32 2E 13 C1 C9 FE 08 20      += 04BE
0BF0  26 C5 0E 04 CD 31 0B 01 06 00 CD 14 0B 3A 2C 13      += 0372
0C00  FE 00 20 0B 3A 2E 13 3D E6 0F 32 2E 13 3E 00 3D      += 03C4
0C10  E6 3F 32 2C 13 C1 C9 FE 09 20 08 C5 0E 07 3E 20      += 0587
0C20  C3 45 0B FE 0B 20 17 C5 0E 06 CD 31 0B 01 06 00      += 043C
0C30  CD 14 0B 3A 2E 13 3D E6 0F 32 2E 13 C1 C9 FE 05      += 0599
0C40  C0 C5 0E 00 CD 31 0B 01 F8 2A CD 14 0B 3E 00 32      += 051B
0C50  2C 13 32 2E 13 C1 C9 00 00 00 00 00 00 00 00 00      += 023C
```

Abb. 6.4.6 Hexdump des Monitorprogramms

```
      MACRO-80 3.4    01-Dec-80       PAGE    1


                                      .z80
0000'                                 cseg

                              ;       crt 9364  support unterprogramme
                              ;       820419 rolf-dieter klein
                              ;       m80 ass-code

                                      global c93co    ;gemeinsamer label crt
                                                      ;zeichenausgabe
                                                      ;zeichen in reg C
                                      global c93ch    ;zeichenausgabe ohne
                                                      ;steuerzeichen

0060                          data    equ     60h     ;port fuer daten
0061                          ctrl    equ     61h     ;port fuer steuerung

                              ; belegung ctrl
                              ; 7     6     5     4     3     2     1     0
                              ; -st   x     x     x    en    c2    c1    c0
                              ;
                              ; befehle         max ms
                              ; page erase cu home 132  1     0     0     0
                              ; cursor home        132  0     0     0     0
                              ; erase to eoln cu re4.2  1     0     0     1
                              ; cursor return      4.2  0     0     0     1
                              ; line feed scroll   8.3  1     0     1     0
                              ; line feed nclr     8.3  0     0     1     0
                              ; nop               .08   x     0     1     1
                              ; cursor left       .08   x     1     0     0
                              ; erase culine      8.3   1     1     0     1
                              ; cursor up         .08   x     1     1     0
                              ; normal char       8.3   1     1     1     1
                              ;

                              ; unterprogramm wait  fuer 2MHZ ausgelegt
                              ; bc=1 dann 12ys

0000'                         wait:
0000'   0B                            dec bc
0001'   78                            ld a,b
0002'   B1                            or c
0003'   C2 0000'                      jp nz,wait
0006'   C9                            ret

                              ; exec schreibt ein cmdwort nach crt
                              ; cmdwort in register C
                              ; optional DATA in register a

0007'                         exec:
0007'   D3 60                         out (data),a
0009'   79                            ld a,c
000A'   F6 80                         or 10000000b    ;st=high
000C'   D3 61                         out (ctrl),a
000E'   E6 7F                         and 01111111b   ;st=low
0010'   D3 61                         out (ctrl),a
0012'   F6 80                         or 10000000b    ;st=high
```

Abb. 6.5.1 Listing des CRT-Programms

```
      MACRO-80 3.4    01-Dec-80       PAGE    1-1


0014'   D3 61                          out (ctrl),a
0016'   C9                            ret

                                ; zeichenausgabe des wertes in register C
                                ; ohne Bereichspruefung

0017'                           c93ch:
0017'   79                              ld a,c          ;datenwert holen

0018'   C5                              push bc
0019'   0E 0F                          ld c,0fh        ;daten schreiben
                                ; eingang 2 spez
001B'   CD 0007'                cbein:  call exec       ;ausfuehrung
001E'   01 000A                         ld bc,130/12    ;130ys .. 8.3ms
0021'   CD 0000'                       call wait
0024'   3A 0001"                       ld a,(posx)     ;scroll nur bei y=15
0027'   FE 3F                          cp 63
0029'   C2 0057'                       jp nz,c1        ;
002C'   3A 0000"                       ld a,(posy)     ;test
002F'   FE 0F                          cp 15           ;dann scroll tritt auf
0031'   C2 004E'                       jp nz,c2
0034'   01 02B3                        ld bc,8300/12   ;warten bis ende
0037'   CD 0000'                       call wait       ;dann loeschen zeile
003A'   3E 20                          ld a," "        ;
003C'   0E 0D                          ld c,1101b      ;cursorzeile loeschen
003E'   CD 0007'                       call exec       ;da falscher wert
0041'   01 02B3                        ld bc,8300/12
0044'   CD 0000'                       call wait
0047'   3E 00                          ld a,0
0049'   32 0001"                       ld (posx),a     ;=0 setzen y bleibt
004C'   C1                            pop bc
004D'   C9                            ret
004E'                           c2:                     ;yposition erhoeht
004E'   3A 0000"                       ld a,(posy)
0051'   3C                            inc a
0052'   E6 0F                          and 00001111b
0054'   32 0000"                       ld (posy),a     ;scrollcount
0057'                           c1:
0057'   3A 0001"                       ld a,(posx)     ;
005A'   3C                            inc a
005B'   E6 3F                          and 00111111b
005D'   32 0001"                       ld (posx),a
0060'   C1                            pop bc          ;alte werte beibehalten
0061'   C9                            ret
                                ;



                                ; crt hauptprogramm
                                ; werte die im Bereich 20h bis FFH
                                ; liegen werden direkt ausgegeben
                                ; sonst wird eine steuerfunktion erfuellt
                                ;


0062'                           c93co:
0062'   79                              ld a,c
```

Abb. 6.5.1 Listing des CRT-Programms

```
     MACRO-80 3.4    01-Dec-80       PAGE    1-2


0063'   FE 20                        cp " "          ;vergleich mit 20h
0065'   DA 006B'                     jp c,steu       ;carry dann < 20h
0068'   C3 0017'                     jp c93ch        ;einfach ausgeben
                        ;
006B'                   steu:                       ;Steuerzeichenbehandlung

006B'   FE 0D                        cp 0dh          ;CR
006D'   C2 0083'                     jp nz,steu1
0070'   C5                          push bc
0071'   0E 01                       ld c,1          ;kein write return
0073'   CD 0007'                    call exec
0076'   01 015E                     ld bc,4200/12   ;4.2 ms
0079'   CD 0000'                    call wait
007C'   3E 00                       ld a,0
007E'   32 0001"                    ld (posx),a     ;0=posy
0081'   C1                          pop bc
0082'   C9                          ret

0083'   FE 0A                  steu1:  cp 0ah          ;LF
0085'   C2 00A4'                     jp nz,steu2
0088'   C5                          push bc
0089'   3E 20                       ld a," "        ;loeschen bei scroll
008B'   0E 0A                       ld c,1010b      ;lf mit scroll loeschen
008D'   CD 0007'                    call exec
0090'   01 02B3                     ld bc,8300/12
0093'   CD 0000'                    call wait
0096'   3A 0000"                    ld a,(posy)
0099'   FE 0F                       cp 15           ;=15 dann bleibt so
009B'   CA 00A2'                     jp z,s1
009E'   3C                          inc a
009F'   32 0000"                    ld (posy),a
00A2'   C1                      s1:     pop bc
00A3'   C9                          ret

00A4'   FE 0C                  steu2:  cp 0ch          ;CLEAR
00A6'   C2 00C1'                     jp nz,steu3
00A9'   C5                          push bc
00AA'   3E 20                       ld a," "        ;Leerzeichen loeschen
00AC'   0E 08                       ld c,1000b
00AE'   CD 0007'                    call exec
00B1'   01 2AF8                     ld bc,11000     ;132ms/12
00B4'   CD 0000'                    call wait
00B7'   3E 00                       ld a,0
00B9'   32 0001"                    ld (posx),a
00BC'   32 0000"                    ld (posy),a
00BF'   C1                          pop bc
00C0'   C9                          ret

00C1'   FE 08                  steu3:  cp 8            ;CURSOR LEFT
00C3'   C2 00ED'                     jp nz,steu4
00C6'   C5                          push bc
00C7'   0E 04                       ld c,0100b
00C9'   CD 0007'                    call exec
00CC'   01 0006                     ld bc,80/12
00CF'   CD 0000'                    call wait
00D2'   3A 0001"                    ld a,(posx)
00D5'   FE 00                       cp 0
```

Abb. 6.5.1 Listing des CRT-Programms

```
      MACRO-80 3.4    01-Dec-80       PAGE    1-3


00D7'   C2 00E5'                      jp nz,s11
00DA'   3A 0000"                      ld a,(posy)
00DD'   3D                            dec a
00DE'   E6 0F                         and 00001111b
00E0'   32 0000"                      ld (posy),a
00E3'   3E 00                         ld a,0
00E5'   3D                    s11:    dec a
00E6'   E6 3F                         and 00111111b
00E8'   32 0001"                      ld (posx),a
00EB'   C1                            pop bc
00EC'   C9                            ret

00ED'   FE 09                 steu4:  cp 9              ;CURSOR RIGHT
00EF'   C2 00FA'                      jp nz,steu5
00F2'   C5                            push bc
00F3'   0E 07                         ld c,0111b        ;wie zeichenaus
00F5'   3E 20                         ld a," "
00F7'   C3 001B'                      jp cbein

00FA'   FE 0B                 steu5:  cp 0bh            ;CURSOR UP
00FC'   C2 0116'                      jp nz,steu6
00FF'   C5                            push bc
0100'   0E 06                         ld c,0110b        ;up
0102'   CD 0007'                      call exec
0105'   01 0006                       ld bc,80/12
0108'   CD 0000'                      call wait
010B'   3A 0000"                      ld a,(posy)
010E'   3D                            dec a
010F'   E6 0F                         and 00001111b
0111'   32 0000"                      ld (posy),a
0114'   C1                            pop bc
0115'   C9                            ret

0116'   FE 05                 steu6:  cp 5              ;HOME
0118'   C2 0131'                      jp nz,steu7
011B'   C5                            push bc
011C'   0E 00                         ld c,0
011E'   CD 0007'                      call exec
0121'   01 2AF8                       ld bc,11000       ;wie clear
0124'   CD 0000'                      call wait
0127'   3E 00                         ld a,0
0129'   32 0001"                      ld (posx),a
012C'   32 0000"                      ld (posy),a
012F'   C1                            pop bc
0130'   C9                            ret

0131'                         steu7:
0131'   C9                            ret               ;ignorieren rest

0132'                                 dseg
0000"   00                    posy:   defb    0         ;y-richtung oben=0
0001"   00                    posx:   defb    0         ;x-richtung links=0
                              ;
                                      end
```

Abb. 6.5.1 Listing des CRT-Programms

```
       MACRO-80 3.4    01-Dec-80        PAGE    1


                                        .Z80
0000'                                   cseg

                                ;* testprogramm fuer crt9364
                                ;* zeichen werden von der console geholt
                                ;* und ausgegeben

                                        external c93ch,c93co

0000'                           start:
0000'   0E 0C                          ld c,0ch        ;clear bildschirm
0002'   CD 0000*                       call c93co

0005'   0E 00                          ld c,0          ;schleife alle zeichen
0007'   C5                      loop:   push bc         ;ausgeben
0008'   CD 0000*                       call c93ch
000B'   C1                              pop bc
000C'   0D                              dec c
000D'   C2 0007'                        jp nz,loop      ;ok
0010'   0E 0A                          ld c,0ah        ;LF CR ausfuehren
0012'   CD 0000*                       call c93co
0015'   0E 0D                          ld c,0dh
0017'   CD 0000*                       call c93co

001A'   CD F003                 loop1:  call 0f003h     ;ZEICHEN HOLEN
001D'   B7                              or a
001E'   CA F01E                         jp z,0f01eh     ;STOP wenn = 0
0021'   4F                              ld c,a
0022'   CD 0000*                       call c93co
0025'   C3 001A'                        jp loop1        ;

                                        end


            MACRO-80 3.4    01-Dec-80        PAGE    S


       Macros:

       Symbols:
       C93CH   0009*   C93CO   0023*   LOOP    0007'   LOOP1   001A'
       START   0000'
```

Abb. 6.5.2 Hauptprogramm für das CRT-Programm

6.5 CRT-Controller-Software

Die Routinen, die auch schon im Monitorprogramm für den CRT-Controller verwendet wurden, sind hier nochmals dargestellt. In *Abb. 6.5.1* ist das Listing der einzelnen Routinen dargestellt. Hier wird eine andere Möglichkeit gezeigt, verschiedene Programmteile zu verbinden. Der Befehl GLOBAL gibt dann, daß die dort genannten Programme weiter verwendet werden können. *Abb. 6.5.2* zeigt das Hauptprogramm. Dort werden die Routinen mit dem Befehl EXTERNAL deklariert. Das bedeutet, daß sie sich in einem ande-

```
0100 00 00 00 0E 0C CD 8F 01 0E 00 C5 CD 44 01 C1 0D ............D...
0110 C2 0A 01 0E 0A CD 8F 01 0E 0D CD 8F 01 CD 03 F0 ................
0120 B7 CA 1E F0 4F CD 8F 01 C3 1D 01 00 00 0B 78 B1 ....O.........x.
0130 C2 2D 01 C9 D3 60 79 F6 80 D3 61 E6 7F D3 61 F6 .-...`y...a...a.
0140 80 D3 61 C9 79 C5 0E 0F CD 34 01 01 0A 00 CD 2D ..a.y....4.....-
0150 01 3A 2C 01 FE 3F C2 84 01 3A 2B 01 FE 0F C2 7B .:,..?...:+....{
0160 01 01 B3 02 CD 2D 01 3E 20 0E 0D CD 34 01 01 B3 .....-.> ...4...
0170 02 CD 2D 01 3E 00 32 2C 01 C1 C9 3A 2B 01 3C E6 ..-.>.2,...:+.<.
0180 0F 32 2B 01 3A 2C 01 3C E6 3F 32 2C 01 C1 C9 79 .2+.:,.<.?2,...y
0190 FE 20 DA 98 01 C3 44 01 FE 0D C2 B0 01 C5 0E 01 . ....D.........
01A0 CD 34 01 01 5E 01 CD 2D 01 3E 00 32 2C 01 C1 C9 .4..^..-.>.2,...
01B0 FE 0A C2 D1 01 C5 3E 20 0E 0A CD 34 01 01 B3 02 ......> ...4....
01C0 CD 2D 01 3A 2B 01 FE 0F CA CF 01 3C 32 2B 01 C1 .-.:+......<2+..
01D0 C9 FE 0C C2 EE 01 C5 3E 20 0E 08 CD 34 01 01 F8 .......> ...4...
01E0 2A CD 2D 01 3E 00 32 2C 01 32 2B 01 C1 C9 FE 08 *.-.>.2,.2+.....
01F0 C2 1A 02 C5 0E 04 CD 34 01 01 06 00 CD 2D 01 3A .......4.....-.:
0200 2C 01 FE 00 C2 12 02 3A 2B 01 3D E6 0F 32 2B 01 ,.......:+.=..2+.
0210 3E 00 3D E6 3F 32 2C 01 C1 C9 FE 09 C2 27 02 C5 >.=.?2,......'..
0220 0E 07 3E 20 C3 48 01 FE 0B C2 43 02 C5 0E 06 CD ..> .H....C.....
0230 34 01 01 06 00 CD 2D 01 3A 2B 01 3D E6 0F 32 2B 4.....-.:+.=..2+
0240 01 C1 C9 FE 05 C2 5E 02 C5 0E 00 CD 34 01 01 F8 ......^.....4...
0250 2A CD 2D 01 3E 00 32 2C 01 32 2B 01 C1 C9 C9     *.-.>.2,.2+....
```

Abb. 6.5.3 Hex-Dump des gebundenen CRT-Programms

ren Programmodul befinden. Mit Hilfe eines Programms das Linker genannt wird, können sie getrennt übersetzten Programme zusammengebunden werden. Die beiden Programme werden dazu vom Assembler in ein Zwischenformat übersetzt, daß alle Informationen über Adressen und Namen, die noch undefiniert sind, enthält. Nach dem Binden entsteht das Programm in *Abb. 6.5.3*. Es wurde in diesem Beispiel auf die Adresse 103h gebunden. Die Befehle des CRT-Controller-ICs sind im Listing auf Abb. 6.5.1 dargestellt.

notenskala fuer ay-3-9812 bei 2000000 HZ

1	1	32.703	32.705	-0.007 %	3822	0EEE
2	1	34.648	34.645	0.008 %	3608	0E18
3	1	36.708	36.711	-0.007 %	3405	0D4D
4	1	38.891	38.892	-0.003 %	3214	0C8E
5	1	41.203	41.200	0.008 %	3034	0BDA
6	1	43.654	43.660	-0.015 %	2863	0B2F
7	1	46.249	46.245	0.009 %	2703	0A8F
8	1	48.999	49.000	-0.003 %	2551	09F7
9	1	51.913	51.910	0.005 %	2408	0968
10	1	55.000	54.993	0.012 %	2273	08E1
11	1	58.270	58.275	-0.009 %	2145	0861
12	1	61.735	61.728	0.011 %	2025	07E9
1	2	65.406	65.411	-0.007 %	1911	0777
2	2	69.296	69.290	0.008 %	1804	070C
3	2	73.416	73.400	0.022 %	1703	06A7
4	2	77.782	77.785	-0.003 %	1607	0647
5	2	82.406	82.399	0.008 %	1517	05ED
6	2	87.308	87.291	0.020 %	1432	0598
7	2	92.498	92.524	-0.028 %	1351	0547
8	2	97.998	97.962	0.036 %	1276	04FC
9	2	103.826	103.821	0.005 %	1204	04B4
10	2	110.000	110.035	-0.032 %	1136	0470
11	2	116.540	116.496	0.038 %	1073	0431
12	2	123.470	123.518	-0.039 %	1012	03F4
1	3	130.812	130.753	0.045 %	956	03BC
2	3	138.592	138.581	0.008 %	902	0386
3	3	146.832	146.886	-0.037 %	851	0353
4	3	155.564	155.473	0.059 %	804	0324
5	3	164.812	164.908	-0.058 %	758	02F6
6	3	174.616	174.581	0.020 %	716	02CC
7	3	184.996	184.911	0.046 %	676	02A4
8	3	195.996	195.925	0.036 %	638	027E
9	3	207.652	207.641	0.005 %	602	025A
10	3	220.000	220.070	-0.032 %	568	0238
11	3	233.080	233.209	-0.055 %	536	0218
12	3	246.940	247.036	-0.039 %	506	01FA
1	4	261.624	261.506	0.045 %	478	01DE
2	4	277.184	277.162	0.008 %	451	01C3
3	4	293.664	293.427	0.081 %	426	01AA
4	4	311.128	310.945	0.059 %	402	0192
5	4	329.624	329.815	-0.058 %	379	017B
6	4	349.232	349.162	0.020 %	358	0166
7	4	369.992	369.822	0.046 %	338	0152
8	4	391.992	391.850	0.036 %	319	013F
9	4	415.304	415.282	0.005 %	301	012D
10	4	440.000	440.141	-0.032 %	284	011C
11	4	466.160	466.418	-0.055 %	268	010C
12	4	493.880	494.071	-0.039 %	253	00FD
1	5	523.248	523.013	0.045 %	239	00EF
2	5	554.368	555.556	-0.214 %	225	00E1
3	5	587.328	586.854	0.081 %	213	00D5
4	5	622.256	621.891	0.059 %	201	00C9
5	5	659.248	657.895	0.205 %	190	00BE
6	5	698.464	698.324	0.020 %	179	00B3
7	5	739.984	739.645	0.046 %	169	00A9
8	5	783.984	786.164	-0.278 %	159	009F
9	5	830.608	833.333	-0.328 %	150	0096
10	5	880.000	880.282	-0.032 %	142	008E
11	5	932.320	932.836	-0.055 %	134	0086
12	5	987.760	984.252	0.355 %	127	007F

Abb. 6.6.1 Tabelle der Konstanten einer Tonleiter

1	6	1046.496	1050.420	-0.375 %	119	0077
2	6	1108.736	1106.195	0.229 %	113	0071
3	6	1174.656	1179.245	-0.391 %	106	006A
4	6	1244.512	1250.000	-0.441 %	100	0064
5	6	1318.496	1315.789	0.205 %	95	005F
6	6	1396.928	1404.494	-0.542 %	89	0059
7	6	1479.968	1488.095	-0.549 %	84	0054
8	6	1567.968	1562.500	0.349 %	80	0050
9	6	1661.216	1666.667	-0.328 %	75	004B
10	6	1760.000	1760.563	-0.032 %	71	0047
11	6	1864.640	1865.672	-0.055 %	67	0043
12	6	1975.520	1984.127	-0.436 %	63	003F
1	7	2092.992	2083.333	0.461 %	60	003C
2	7	2217.472	2232.143	-0.662 %	56	0038
3	7	2349.312	2358.491	-0.391 %	53	0035
4	7	2489.024	2500.000	-0.441 %	50	0032
5	7	2636.992	2659.574	-0.856 %	47	002F
6	7	2793.856	2777.778	0.575 %	45	002D
7	7	2959.936	2976.190	-0.549 %	42	002A
8	7	3135.936	3125.000	0.349 %	40	0028
9	7	3322.432	3289.474	0.992 %	38	0026
10	7	3520.000	3472.222	1.357 %	36	0024
11	7	3729.280	3676.471	1.416 %	34	0022
12	7	3951.040	3906.250	1.134 %	32	0020
1	8	4185.984	4166.667	0.461 %	30	001E
2	8	4434.944	4464.286	-0.662 %	28	001C
3	8	4698.624	4629.630	1.468 %	27	001B
4	8	4978.048	5000.000	-0.441 %	25	0019
5	8	5273.984	5208.333	1.245 %	24	0018
6	8	5587.712	5681.818	-1.684 %	22	0016
7	8	5919.872	5952.381	-0.549 %	21	0015
8	8	6271.872	6250.000	0.349 %	20	0014
9	8	6644.864	6578.947	0.992 %	19	0013
10	8	7040.000	6944.444	1.357 %	18	0012
11	8	7458.560	7352.941	1.416 %	17	0011
12	8	7902.080	7812.500	1.134 %	16	0010

Abb. 6.6.1 Tabelle der Konstanten einer Tonleiter

6.6 Der Soundgenerator als Musikinstrument

Ein hübsches Programmbeispiel ergibt sich mit dem Sound-Generator. In *Abb. 6.6.1 a+b* ist eine Tabelle von Konstanten gezeigt, um eine Tonleiter zu erzeugen. Mit Hilfe dieser Tabelle läßt sich ein Musikinstrument programmieren. *Abb. 6.6.2* zeigt ein BASIC-Programm zur Erzeugung der Tabelle für diejenigen, die BASIC verstehen. In *Abb. 6.6.3* ist eine Klaviatur gezeigt, wie sie sich auf einer Tastatur abbilden läßt. Das Klavierprogramm zeigt *Abb. 6.6.4*. In diesem Programm findet auch die Makroverarbeitung eine Anwendung. *Abb. 6.6.5* zeigt einen Hex-Dump des fertigen Programms. Es kann direkt in den Computer eingegeben werden.

```
10 REM ay-3-9812
12 H=1
15 DIM N(12):FORI=1TO12:READN(I):NEXTI
20 J=1 ' in zweierpotenzen
30 PRINT#D,"notenskala"
31 INPUT"ausgabe 0,2";D
32 INPUT"takt in HZ";TK
34 PRINT#D,"notenskala fuer ay-3-9812 bei";TK;" HZ"
40 FOR I=1 TO 12
50 K=TK/(N(I)*J*16)
55 F=TK/(INT(K+.5)*16)
60 PRINT#D,USING110;I,H,N(I)*J,F,(N(I)*J-F)/(N(I)*J)*100,INT(K+.5),HEX$(
INT(K+.5))
70 NEXTI
80 J=J*2
85 H=H+1
90 IF J<256 THEN 40
100 END
110 !  ##  ##       ####.###    ####.###    ##.### %   ####  'LLL
1000 DATA 32.703,34.648,36.708,38.891,41.203,43.654
1010 DATA 46.249,48.999,51.913,55.000,58.270,61.735
```

Abb. 6.6.2 BASIC-Programm zur Erzeugung der Konstanten

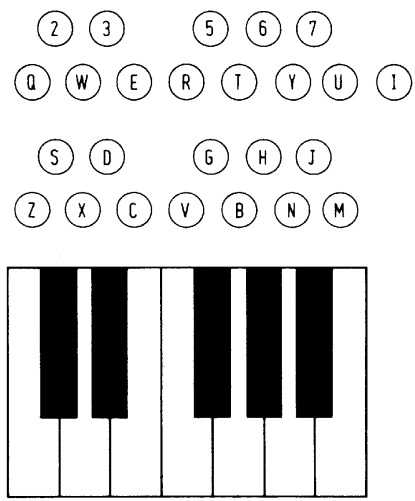


Abb. 6.6.3 Anordnung der Tasten der Klaviatur

```
      MACRO-80 3.4    01-Dec-80       PAGE    1


                              .z80
0000'                         aseg
                              org 1000h       ;start im ram

                              ;****************************************
                              ;* Sound Generator AY-3-98-12           *
                              ;* Rolf-Dieter Klein 820227      1.0    *
                              ;****************************************


1000    C3 10DC               jp start                ;hauptprogramm


00E0                          cmd     equ     0e0h    ;cmd aus und daten ein
00E1                          dat     equ     0e1h    ;daten aus


                              ; arbeiten mit makros erleichtert spaeter die
                              ; lesbarkeit
                              ;
                              ; sound adr,data  gibt den wert data an den
                              ;                 soundgenerator aus
                              ;
                              sound   macro   adr,data        ;makrodefinitio

                              ld a,adr                ;parameter laden
                              out (cmd),a             ;und befehl aus
                              ld a,data               ;wert auch
                              out (dat),a             ;ausgeben
                              endm                    ;ende makrodef
                              ;
                              ;
                              ; soundi  adr gibt den akkumulator an
                              ; den soundgenerator aus
                              ;
                              soundi  macro   adr             ;
                              push psw                ;akku retten
                              ld a,adr                ;adresse aus
                              out (cmd),a
                              pop psw                 ;daten wert
                              out (dat),a
                              endm                    ;ende makrodef
                              ;
                              ;

                              ; sndtab adr     gibt eine ganze tabelle
                              ;                an den soundgenerator
                              ;                und belegt damit alle
                              ;                register
                              ;
                              sndtab  macro   adr
                              ld hl,adr       ;start der tabelle
                              call tabaus     ;uprg aufrufen
                              endm
                              ;
                              ;
                              ; delay ms       wartet ms milisekunden
```

Abb. 6.6.4 Listing des Musikprogramms

```
      MACRO-80 3.4    01-Dec-80       PAGE    1-1


                              ;
                              delay   macro   ms
                              ld bc,ms
                              call mswait
                              endm            ;ende
                              ;
                              ;

                              ; unterprogramm tabellenausgabe fuer fertig
                              ; aufbereitete tabellen

1003                          tabaus:
1003    06 10                 ld b,16         ;16 eintraege
1005    0E E0                 ld c,cmd        ;adresse port
1007    16 00                 ld d,0          ;start adr
1009                          tblp1:
1009    ED 51                  out (c),d      ;adresse aus
100B    7E                     ld a,(hl)
100C    D3 E1                  out (dat),a    ;und data wert
100E    23                     inc hl         ;next adr
100F    14                     inc d          ;next adresse
1010    10 F7                 djnz tblp1
1012    C9                    ret

                              ;
                              ; in bc anzahl milisekunden
                              ;
1013                          mswait:
1013    C5                     push bc        ;retten zeit
1014    01 0053                ld bc,1000/12  ;1ms
1017                           mslp1:
1017    0B                      dec bc
1018    78                      ld a,b
1019    B1                      or c
101A    C2 1017                jp nz,mslp1
101D    C1                     pop bc
101E    0B                     dec bc
101F    78                     ld a,b
1020    B1                     or c
1021    C2 1013               jp nz,mswait
1024    C9                    ret
                              ;

                              ; tonhoehen
                              ;
1025                          noten:
1025    01DE                  defw    478     ;1 4
1027    01C3                  defw    451     ;2 4
1029    01AA                  defw    426     ;3 4
102B    0192                  defw    402     ;4 4
102D    017B                  defw    379     ;5 4
102F    0166                  defw    358     ;6 4
1031    0152                  defw    338     ;7 4
1033    013F                  defw    319     ;8 4
1035    012D                  defw    301     ;9 4
1037    011C                  defw    284     ;10 4
```

Abb. 6.6.4 Listing des Musikprogramms

```
      MACRO-80 3.4    01-Dec-80       PAGE    1-2


1039    010C                    defw    268     ;11 4
103B    00FD                    defw    253     ;12 4
103D    00EF                    defw    239     ;1 5
103F    00E1                    defw    225     ;2 5
1041    00D5                    defw    213     ;3 5
1043    00C9                    defw    201     ;4 5
1045    00BE                    defw    190     ;5 5
1047    00B3                    defw    179     ;6 5
1049    00A9                    defw    169     ;7 5
104B    009F                    defw    159     ;8 5
104D    0096                    defw    150     ;9 5
104F    008E                    defw    142     ;10 5
1051    0086                    defw    134     ;11 5
1053    007F                    defw    127     ;12 5
1055    0077                    defw    119     ;1 6
                                ;

                                ; tastatur zuordnungstabelle
                                ; um mit ascii-tastatur melodien
                                ; geben zu koennen

1057                            tabkey:
1057    5A 53 58 44             defm'ZSXDCVGBHNJM'
105B    43 56 47 42
105F    48 4E 4A 4D
1063    51 32 57 33             defm'Q2W3ER5T6Y7UI'
1067    45 52 35 54
106B    36 59 37 55
106F    49
                                ;
1070                            tabk2:          ;alternative
1070    7A 73 78 64             defm 'zsxdcvgbhnjm'
1074    63 76 67 62
1078    68 6E 6A 6D
107C    71 32 77 33             defm 'q2w3er5t6y7ui'
1080    65 72 35 74
1084    36 79 37 75
1088    69


                                ;
                                ; ergebnis in de und hl notenwert
                                ;
1089                            suchind:        ;suche index zeichen in akku
1089    21 1057                 ld hl,tabkey    ;erste tabelle
108C    06 19                   ld b,25         ;zwei tonleitern
108E    0E 00                   ld c,0          ;index
1090                            sulp1:
1090    BE                       cp (hl)
1091    28 15                    jr z,sufo      ;found
1093    23                       inc hl
1094    0C                       inc c
1095    10 F9                   djnz sulp1
1097    21 1070                 ld hl,tabk2     ;vielleicht 2.tabelle
109A    06 19                   ld b,25
109C    0E 00                   ld c,0
109E                            sulp2:
```

Abb. 6.6.4 Listing des Musikprogramms

```
        MACRO-80 3.4    01-Dec-80       PAGE    1-3


109E    BE                      cp (hl)
109F    28 07                   jr z,sufo
10A1    23                      inc hl
10A2    0C                      inc c
10A3    10 F9                   djnz sulp2      ;
10A5    AF                      xor a
10A6    37                      scf
10A7    C9                      ret             ;carry = not found
10A8                            sufo:           ;ok
10A8    21 1025                 ld hl,noten     ;index berechnen
10AB    06 00                   ld b,0          ;2*bc + hl -> hl
10AD    09                      add hl,bc
10AE    09                      add hl,bc       ;da wortabstand
10AF    5E                      ld e,(hl)       ;lsb
10B0    23                      inc hl
10B1    56                      ld d,(hl)       ;DE ist ergebnis
10B2    AF                      xor a
10B3    D5                      push de
10B4    E1                      pop hl          ;und HL auch
10B5    C9                      ret             ;no carry ok

                                ;
10B6                            piano:          ;tabelle IM RAM
10B6    0000                    defw    0       ;note a
10B8    0000                    defw    0       ;note b
10BA    0000                    defw    0       ;note c
10BC    00                      defb    0       ;noise
10BD    38                      defb    38h     ;enable
10BE    10                      defb    10h     ;envelop ampl a
10BF    10                      defb    10h     ;ampl b
10C0    10                      defb    10h     ;ampl c
10C1    00                      defb    0       ;lsb envelop
10C2    0A                      defb    0ah     ;msb envelop
10C3    00                      defb    0       ;shape
10C4    00 00                   defb    0,0     ;io
                                ;
10C6                            tonaus:         ;fuer piano
10C6    22 10B6                 ld (piano),hl
10C9    2B                      dec hl          ;interferenze
10CA    22 10B8                 ld (piano+2),hl
10CD    23                      inc hl
10CE    CB 0C                   rrc h
10D0    CB 1D                   rr l            ;/2 oktave tiefer
10D2    22 10BA                 ld (piano+4),hl
                                sndtab piano    ;makro verwenden
10D5    21 10B6         +       ld hl,piano     ;start der tabelle
10D8    CD 1003         +       call tabaus     ;uprg aufrufen
10DB    C9                      ret             ;fuer ausgabe
                                ;


                                ; HAUPTPROGRAMM
                                ;
10DC                            start:
10DC    CD 0003                 call 03h        ;CI Consol zeichen
10DF    CD 1089                 call suchind
```

Abb. 6.6.4 Listing des Musikprogramms

```
       MACRO-80 3.4    01-Dec-80        PAGE    1-4

10E2    38 F8                   jr c,start      ;bis gueltig
10E4    CD 10C6                 call tonaus     ;und ausgeben dann
10E7    18 F3                   jr start


                                end

           MACRO-80 3.4    01-Dec-80        PAGE    S

    Macros:
    DELAY   SNDTAB  SOUND   SOUNDI

    Symbols:
    CMD     00E0    DAT     00E1    MSLP1   1017    MSWAIT  1013
    NOTEN   1025    PIANO   10B6    START   10DC    SUCHIN  1089
    SUFO    10A8    SULP1   1090    SULP2   109E    TABAUS  1003
    TABK2   1070    TABKEY  1057    TBLP1   1009    TONAUS  10C6


    No  Fatal error(s)
```

Abb. 6.6.4 Listing des Musikprogramms

```
-
-d1000 10e8
1000 C3 DC 10 06 10 0E E0 16 00 ED 51 7E D3 E1 23 14 ..........Q~..#.
1010 10 F7 C9 C5 01 53 00 0B 78 B1 C2 17 10 C1 0B 78 .....S..x......x
1020 B1 C2 13 10 C9 DE 01 C3 01 AA 01 92 01 7B 01 66 .............{.f
1030 01 52 01 3F 01 2D 01 1C 01 0C 01 FD 00 EF 00 E1 .R.?.-..........
1040 00 D5 00 C9 00 BE 00 B3 00 A9 00 9F 00 96 00 8E ................
1050 00 86 00 7F 00 77 00 5A 53 58 44 43 56 47 42 48 .....w.ZSXDCVGBH
1060 4E 4A 4D 51 32 57 33 45 52 35 54 36 59 37 55 49 NJMQ2W3ER5T6Y7UI
1070 7A 73 78 64 63 76 67 62 68 6E 6A 6D 71 32 77 33 zsxdcvgbhnjmq2w3
1080 65 72 35 74 36 79 37 75 69 21 57 10 06 19 0E 00 er5t6y7ui!W.....
1090 BE 28 15 23 0C 10 F9 21 70 10 06 19 0E 00 BE 28 .(.#...!p......(
10A0 07 23 0C 10 F9 AF 37 C9 21 25 10 06 00 09 09 5E .#....7.!%.....^
10B0 23 56 AF D5 E1 C9 00 00 00 00 00 00 00 38 10 10 #V...........8..
10C0 10 00 0A 00 00 00 22 B6 10 2B 22 B8 10 23 CB 0C ......"..+"..#..
10D0 CB 1D 22 BA 10 21 B6 10 CD 03 10 C9 CD 03 00 CD ..".."..!.......
10E0 89 10 38 F8 CD C6 10 18 F3 ..8.....
-
```

Abb. 6.6.5 Hexdump des Musikprogramms

6.7 Der Soundgenerator als Geräuschgenerator

Mit dem Sound-Generator können aber auch Geräusche erzeugt werden. In *Abb. 6.7.1* ist ein Beispielprogramm gezeigt. Bei der Eingabe des Programms ist wie immer darauf zu achten, daß der Assembler den Adreßteil in der Form HIGH LOW ausgibt und die Bytes im Speicher in umgekehrter Reihenfolge stehen müssen.

```
      MACRO-80 3.4    01-Dec-80       PAGE    1


                              .z80
0000'                         aseg
                              org 1000h       ;start im ram

                              ;****************************************
                              ;* Sound Generator AY-3-98-12           *
                              ;* Rolf-Dieter Klein 820227      1.0    *
                              ;* Geraeusche Demo                      *
                              ;****************************************

1000    C3 1065               jp start                ;hauptprogramm


00E0                          cmd     equ     0e0h    ;cmd aus und daten ein
00E1                          dat     equ     0e1h    ;daten aus


                              ; arbeiten mit makros erleichtert spaeter die
                              ; lesbarkeit
                              ;
                              ; sound adr,data  gibt den wert data an den
                              ;                 soundgenerator aus
                              ;
                              sound   macro   adr,data        ;makrodefinition

                              ld a,adr                ;parameter laden
                              out (cmd),a             ;und befehl aus
                              ld a,data               ;wert auch
                              out (dat),a             ;ausgeben
                              endm                    ;ende makrodef
                              ;
                              ;
                              ; soundi  adr gibt den akkumulator an
                              ; den soundgenerator aus
                              ;
                              soundi  macro   adr             ;
                              push psw                ;akku retten
                              ld a,adr                ;adresse aus
                              out (cmd),a
                              pop psw                 ;daten wert
                              out (dat),a
                              endm                    ;ende makrodef
                              ;
                              ;

                              ; sndtab adr     gibt eine ganze tabelle
                              ;                an den soundgenerator
                              ;                und belegt damit alle
                              ;                register
                              ;
                              sndtab  macro   adr
                              ld hl,adr       ;start der tabelle
                              call tabaus     ;uprg aufrufen
                              endm
                              ;
                              ;
```

Abb. 6.7.1 Listing des Geräuschedemos

```
MACRO-80 3.4    01-Dec-80       PAGE    1-1


                                ; delay ms      wartet ms milisekunden
                                ;
                                delay   macro   ms
                                ld bc,ms
                                call mswait
                                endm            ;ende
                                ;
                                ;

                                ; unterprogramm tabellenausgabe fuer fertig
                                ; aufbereitete tabellen

1003                            tabaus:
1003    06 10                   ld b,16         ;16 eintraege
1005    0E E0                   ld c,cmd        ;adresse port
1007    16 00                   ld d,0          ;start adr
1009                            tblp1:
1009    ED 51                    out (c),d      ;adresse aus
100B    7E                       ld a,(hl)
100C    D3 E1                    out (dat),a    ;und data wert
100E    23                       inc hl         ;next adr
100F    14                       inc d          ;next adresse
1010    10 F7                   djnz tblp1
1012    C9                      ret

                                ;
                                ; in bc anzahl milisekunden
                                ;
1013                            mswait:
1013    C5                       push bc        ;retten zeit
1014    01 0053                  ld bc,1000/12  ;1ms
1017                             mslp1:
1017    0B                        dec bc
1018    78                        ld a,b
1019    B1                        or c
101A    C2 1017                  jp nz,mslp1
101D    C1                       pop bc
101E    0B                       dec bc
101F    78                       ld a,b
1020    B1                       or c
1021    C2 1013                 jp nz,mswait
1024    C9                      ret
                                ;

                                ; verschiedene geraeuschtabellen
                                ; fuer 2MHZ Takt

1025                            loko:
1025    00 00 00 00             defb 0,0,0,0,0,0
1029    00 00
102B    0F C7 10 10             defb 15,199,16,16,16
102F    10
1030    B4 03 0C                defb 180,3,12
1033    00 00                   defb 0,0
                                ;
1035                            meer:
```

Abb. 6.7.1 Listing des Geräuschedemos

```
      MACRO-80 3.4    01-Dec-80       PAGE    1-2


1035    00 00 00 00           defb 0,0,0,0,0,0
1039    00 00
103B    1F C7 10 10           defb 31,199,16,16,16
103F    10
1040    FF 3C 0E 00           defb 255,60,14,0,0
1044    00
                              ;
1045                          laser:
1045    00 00 00 00           defb 0,0,0,0,0,0
1049    00 00
104B    00 3E 0F 00           defb 0,3eh,15,0,0
104F    00
1050    00 00 00 00           defb 0,0,0,0,0
1054    00
                              ;
1055                          stop:
1055    00 00 00 00           defb 0,0,0,0,0,0
1059    00 00
105B    00 00 00 00           defb 0,0,0,0,0,0
105F    00 00
1061    00 00 00 00           defb 0,0,0,0
                              ;

1065                          start:
                              sndtab stop     ;generator ausschalten
1065    21 1055         +     ld hl,stop      ;start der tabelle
1068    CD 1003         +     call tabaus     ;uprg aufrufen
                              delay 2000
106B    01 07D0         +     ld bc,2000
106E    CD 1013         +     call mswait
1071    0E 04                 ld c,4          ;4 mal laser
1073                          laslp:
1073    C5                     push bc
1074    06 28                  ld b,40         ;start
1076                           stlp:
1076    C5                      push bc
1077    78                      ld a,b
1078    32 1045                 ld (laser+0),a        ;muss ram sein
                                delay 1
107B    01 0001         +     ld bc,1
107E    CD 1013         +     call mswait
                                sndtab laser    ;und ausgabe
1081    21 1045         +     ld hl,laser     ;start der tabelle
1084    CD 1003         +     call tabaus     ;uprg aufrufen
1087    C1                      pop bc
1088    04                      inc b
1089    78                      ld a,b
108A    FE B4                   cp 180          ;obere grenze
108C    20 E8                  jr nz,stlp
                               sndtab stop     ;und ausschalten
108E    21 1055         +     ld hl,stop      ;start der tabelle
1091    CD 1003         +     call tabaus     ;uprg aufrufen
                               delay 20
1094    01 0014         +     ld bc,20
1097    CD 1013         +     call mswait
109A    C1                     pop bc
```

Abb. 6.7.1 Listing des Geräuschedemos

```
      MACRO-80 3.4    01-Dec-80       PAGE    1-3


109B    0D                      dec c
109C    20 D5                   jr nz,laslp
                                delay 2000      ;
109E    01 07D0         +       ld bc,2000
10A1    CD 1013         +       call mswait
                                sndtab loko     ;nun lokomotive
10A4    21 1025         +       ld hl,loko      ;start der tabelle
10A7    CD 1003         +       call tabaus     ;uprg aufrufen
                                delay 5000
10AA    01 1388         +       ld bc,5000
10AD    CD 1013         +       call mswait
                                sndtab stop
10B0    21 1055         +       ld hl,stop      ;start der tabelle
10B3    CD 1003         +       call tabaus     ;uprg aufrufen
                                delay 2000
10B6    01 07D0         +       ld bc,2000
10B9    CD 1013         +       call mswait
                                sndtab meer     ;meeresrauschen
10BC    21 1035         +       ld hl,meer      ;start der tabelle
10BF    CD 1003         +       call tabaus     ;uprg aufrufen
                                delay 10000
10C2    01 2710         +       ld bc,10000
10C5    CD 1013         +       call mswait
                                sndtab stop
10C8    21 1055         +       ld hl,stop      ;start der tabelle
10CB    CD 1003         +       call tabaus     ;uprg aufrufen
                                ;
10CE    CD 0000                 call 0          ;ende BREAK
                                ;




                        end


      MACRO-80 3.4    01-Dec-80       PAGE    S

Macros:
DELAY   SNDTAB  SOUND   SOUNDI

Symbols:
CMD     00E0    DAT     00E1    LASER   1045    LASLP   1073
LOKO    1025    MEER    1035    MSLP1   1017    MSWAIT  1013
START   1065    STLP    1076    STOP    1055    TABAUS  1003
TBLP1   1009


No  Fatal error(s)
```

Abb. 6.7.1 Listing des Geräuschedemos

6.8 Fragen zum Software-Teil

1. Was bewirkt der Befehl LD A, (1000h)?
2. Der Befehl JP 4567h ist zu codieren.
3. Gegeben ein Programm wie folgt:

```
LOOP: INC A
      CALL LOOP
```

was passiert?

4. Was bewirkt der Befehl LD HL, 1000h?
5. Was passiert beim Befehl LD HL, (1000h)?
6. Welcher Wert steht nach dem folgenden Programm im Akku:

```
ALPHA EQU 3
BETHA EQU ALPHA+3
      LD A,BETHA
```

7. Warum wird in Abb. 6.3.3 der Befehl JR Z,ELSEPART und nicht JR NZ,ELSEPART angegeben?
8. Warum genügt es nicht in Abb. 6.3.13 einfach JR NZ,DOLP ohne die beiden Befehle LD A,E; OR D zu schreiben?
9. Welche Baudrate ist bei der Einstellung von

11010010 an den Schaltern von der Karte KEY

zu erwarten?

10. Wozu wird die Warteschleife WAIT in Abb. 6.5.1 gebraucht?

7 Ausblick

In dem Buch konnten nur die wichtigsten Grundlagen behandelt werden. Wer Geschmack an der Programmierung oder beim Aufbau von Mikrorechnerschaltungen gefunden hat, der findet eine Fortsetzung in dem Buch „Mikrocomputer entwickeln mit Moduln" in dem weitere Platinen sowie höhere Softwarekonzepte behandelt werden.

8 Lösung zu den im Text gestellten Fragen

zu 1.6

1. Die Siebelkos sollen die gleichgerichtete Wechselspannung glätten.
2. Blockkondensatoren dienen hier der Unterdrückung von Schwingneigung.

zu 2.5

1. *Abb. 8.1* zeigt zwei mögliche Lösungen. Die benötigten Nicht-Gatter können ebenfalls aus dem Nand-Gatter erzeugt werden. Dabei ist der zweite Weg zu bevorzugen, da die Eingangslast nicht 1 TTL-Last beträgt. Werden zwei Eingänge zusammengeschaltet, so muß ein größerer Eingangsstrom fließen, um die Gatter zu schalten. Ein davor liegender Ausgang wird dann stärker belastet.

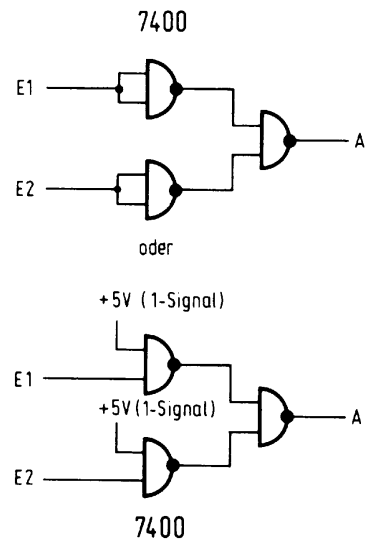


Abb. 8.1 Lösung
Oder-Gatter

2. *Abb. 8.2* zeigt eine Lösung. Mit den Setz-Rücksetzeingängen kann ein RS-Flip-Flop simuliert werden. Den Takt und D-Eingang benötigt man dazu nicht.

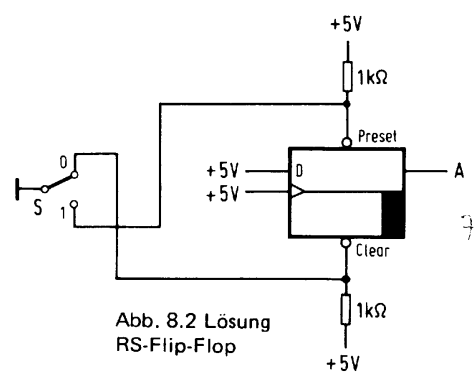


Abb. 8.2 Lösung
RS-Flip-Flop

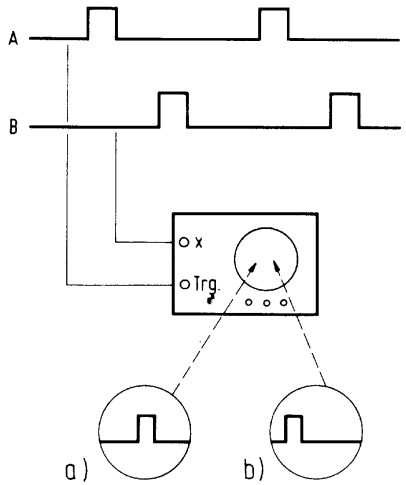


Abb. 8.3 Lösung des Meßproblems

zu 3.4

1. *Abb. 8.3* zeigt die Lösung. Das eine der beiden Signale wird an den Trigger-Eingang geschaltet, das andere auf den X-Eingang. Erscheinen die beiden Pulse nicht gleichzeitig, so ergibt sich ein Bild ähnlich a), treten sie gleichzeitig auf, so sieht das Scop-Bild eher wie b) aus.

2. Nein. Mit dem Prüfstift können solche Pulsfolgen nicht von einem statischen Signal unterschieden werden. Abhilfe schafft ein Monoflop, z. B. 74121 das von jedem Puls getriggert wird und eine Led am Ausgang besitzt. Die Zeitkonstante kann ca. 1/2 sek. gewählt werden.

zu 4.9

1. Da sich im allgemeinen gleiche Ausgangsfunktionen wiederholen und daher eine Lösung z. B. nach Abb. 4.1.4 platzsparender ist.

2. Da der Dekoder mit einem tiefaktiven Signal arbeitet und somit eine negative Logik besitzt, sind die Ausgänge normalerweise auf einem High-Pegel. Nur bei einem Low-Signal soll die Diode durchschalten. Daher wird die Kathode benötigt.

3. Der Ladeeingang ist nötig, um die Sprungadresse in den Programmzähler zu übertragen.

4. Das Befehlsformat kann in beliebiger Reihenfolge aus dem Elementen (Sprungadr, Sprung, A, B, C, D, E) zusammengesetzt werden. Es empfihelt sich nur das Befehlsformat möglichst übersichtlich darzustellen.

5. Bei 10 Adreßeingängen gibt es 2 HOCH 10 also 1024 Möglichkeiten, somit müssen in dieser Schaltung 1024 Ausgänge zur Adressierung der einzelnen Zeilen vorhanden sein. Um diese große Zahl zu verringern, wird in der Praxis der Dekoder in zwei Teile gespalten, die jeweils 5 Adressen (in unserem Beispiel) erhalten würden. Die Dekoder brauchen dann nur 2 HOCH 5 also 32 Ausgänge zu besitzen und die Ausgänge werden Matrixförmig an die Speicherzellen geführt. Dort muß dann allerdings noch eine UND-Verknüpfung durchgeführt werden, die jedoch nicht so aufwendig ist.

6. Das System würde dann freilaufen. Sobald ein Zustand am Ausgang angelengt ist, wird sofort der nächste wieder eingenommen. Dies führt aber noch zu anderen Problemen. Die Laufzeit für die einzelnen Signale ist nicht gleich groß und Fehlzeiten würden zu fehlerhaften Informationen zwischendurch führen, das ganze würde also überhaupt nicht richtig arbeiten. Eine Möglichkeit gibt es aber dennoch die Anordnung verwenden zu können, es muß dafür gesorgt werden, daß sich immer nur ein Ausgangssignal gleichzeitig ändern kann, dann wäre der Folgezustand eindeutig. Dennoch muß in der Praxis dringend vor solchen Lösungen gewanrt werden, da das Zeitverhalten der gesamten Anordnung völlig unbestimmt bleibt.

7. Die Lösung ist verblüffend einfach: weil sie nicht benötigt werden.

8. Wenn der Programmierpuls bei unseren Promtypen länger als 50ms ist, so werden sie im allgemeinen zerstört.

9. Es ist die Zeit, die von Beginn das Selekt-Signals bis zum Erhalt der Information auf der Datenleitung verstreicht.

10. Im Prinzip ja, jedoch fehlt hier jegliche Peripherie, um mit der Außenwelt in Verbindung zu treten.

11. Bei 16 Adreßleitungen kann er 2 HOCH 16 Adressen, also 65536 Adreßplätze.

12. Um getrennte Adressräume zu erhalten und Peripherie vom Speicher trennen zu können. Damit sind insgesamt 65536 + 256 Adressen anwählbar. Bei anderen Prozessoren z. B. 6502 ist das nicht so, dort müssen Peripherieplätze von dem Speicherraum abgegeben werden. Die Auswahl erfolgt beim Z80 durch die Signale -MREQ (Speicher) und -IORQ (Peripherie).

13. Der Z80 kann 256 Peripherie-Geräte adressieren. Werden aber mehr benötigt, so kann auf einen Teil des Speicheradreßraums verzichtet und damit die Zahl an Peripherieplätzen noch weiter erhöht werden.

14. Mit dem -WAIT-Eingng des Z80s kann die CPU bei Zugriffen auf Speicher oder Peripherie angehalten werden, solange bis die Daten gültig sind. Damit ist es möglich, auch langsame Speicher oder Peripherie-Einheiten anzukoppeln.

15. Da dann auch bei einem Peripheriezugriff ein Selekt erfolgen würde.

16. Der Widerstand R3 ist ein sogenannter Pull-up-Widerstnd und hat die Aufgabe, den Pegel bei einem 1-Signal auf nahe 5V anzuheben, damit die CPU einwandfrei arbeiten kann.

17. Da -MREQ schon über die Selekt-Eingänge eingebracht wurde.

18. Mit -M1 alleine würde nur bei einem OP-Code-Fetch eine Busumschaltung erfolgen und wird nur -RD verwendet, ist der Betrieb mit Interrupt-Verkehr nicht möglich, da dieser auch von der CPU eingelesen werden muß, aber kein -RD sondern nur -M1 und -IORQ anliegt.

19. Damit ist es möglich, mehrere Karten auf den gleichen Adreßbereich zu legen und über eine externe Dekodierung deren Ausgang an BANKEN zuführen und eine von diesen Karten auszuwählen.

20. Wenn der Gesamtraum an Adreßplätzen nicht ausreicht. Oder wenn es nötig ist, mehrere Speicherbereiche auf der gleichen Adresse anzusprechen. Soll zum Beispiel beim Stromeinschalten auf dem Adreßberich 0 bis 2FFFh ein EPROM Bereich liegen und spä-

ter dort ein RAM-Bereich sein, so werden RAM und EPROM mit gleichen Adressen angesprochen, jedoch wird mit einer Zusatzlogik eine dieser beiden BANKS angewählt und nur eine zur gleichen Zeit angesprochen.

zu 5.8

1. Peripheriegeräte sind: Drucker, Datensichtgeräte.
2. Mit der parallelen Schnittstelle lassen sich Daten schneller übertragen, da mehrere Bits gleichzeitig übertragen werden können.
3. Es gilt Anzahl der Zeichen pro Sekunde, ist

   300 / (7+1+1) Zeichen/Sekunde
   = 33.333 Zeichen/Sekunde

4. Bitmuster + even Parität: 10010110 0. Bei den seriellen Bausteinen wird aber die Parität im allgemeinen als MSB-Bit übertragen.
5. Sie müssen die Spannungspegel anpassen. Einmal von +/-12V auf den ITL-Pegel und umgekehrt.
6. Das Zeichen ‚o' besitzt den ASCII-Code 6Fh.
7. Sonst wäre der Ausgang blockiert. CTS muß auf einem +12V-Pegel liegen, damit an -CTS 0V liegt. Der Ausgang RTS führt dieses Signal und kann daher als Spannungsquelle verwendet werden.
8. Bei CTRL-C wird der Code 03 (ETX) ausgegeben. Dazu wird einfach der ASCII-Code des Buchstabens ‚C' genommen und 40h subtrahiert.
9. Der Zeichengenerator beinhaltet sämtliche darstellbare Zeichen.
10. Der Bildwiederholspeicher wird durch die beiden ICs 4045 SP1 und SP2 realisiert.
11. Die Codierung des Zeichens ist wie folgt:

Zeichen	Binär	HEX
....	00000000	00
.******.	01111110	7E
..*..*..	00100100	24
...**...	00011000	18
....	00000000	00
....	00000000	00
....	00000000	00
....	00000000	00

Der Code wird dann nacheinander abgelegt, 0, 7Eh, 24h, 18h, 0, 0, 0, 0.

12. Das Zeichen soll bei dem Code 7Fh erscheinen. Dazu errechnet sich die EPROM Adresse: 7Fh * 8 oder dezimal 127 * 8 = 1016 oder in HEX = 3F8h. Beginnend bei Adresse 3F8h bis Adresse 3FFh wird der obige Code in das EPROM eingegeben.

13. I3 in Abb. 5.4.1 hat die Aufgaben, den Pegel am Programmiereingang im Falle des Ruhezustands, oder beim Lesen des EPROM-Inhalts auf 0 V zu legen.

14. Wenn I1 an Masse läge, würde auch bei einem Speicherzugriff ein Signal am Ausgang des Vergleichers zu einer Selektion der Karte führen. Da die Karte nur bei IO-Operationen ansprechen soll, muß die Verknüpfung mit -IORQ durchgeführt werden.

15. Die Polarität der Daten kann mit dem Schalter S1 umgedreht werden. Das ist nötig, weil einige Kassettenrecorder ein gegenüber der Aufnahme inverses Signal liefern.

16. Die Amplitude eines Kanals oder mehrere Kanäle kann mit dem Hüllkurvengenerator nach der in Abb. 5.7.3 abgebildeten Form moduliert werden.

zu 6.8

1. Durch den Befehl LD A, (1000h) wird der auf der Adresse 1000h stehende Wert ins Register A geladen. A ist der Akkumulator.

2. Es ergibt sich für den Befehl JP 4567h

   C3 67 45

3. Das Programm hört nie auf, jedoch wird der Stack immer tiefer und wenn das Programm im RAM steht, wird er es zerstören.

4. Mit LD HL, 1000h wird der Wert 10h nach H und 00h nach L geladen.

5. Beim Befehl LD HL, (1000h) wird der Inhalt der Zeile 1000h in das Register L und der Inhalt der Zeile 1001h in das Register H geladen.

6. Es ist der Wert 6, da in BETHA der Wert 6 steht.

7. Der JA-Teil soll genau dann ausgeführt werden, wenn die Bedingung NZ erfüllt ist. Damit muß der Ja-Teil genau dann übersprungen werden, wenn Z vorliegt.

8. Der Befehl DEC DE setzt das Zero-Flag nicht.

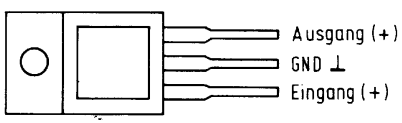
9. Es sind 75 BAUD.

10. Der CRT-Controller benötigt eine gewisse Zeit, um die Befehle verarbeiten zu können, daher muß im Programm solange gewartet werden, wie die Ausführung dauert.

9 Pinbelegungen und Timings

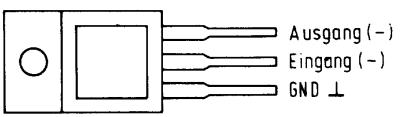
In *Abb. 9.1 a...u* sind verschiedene Pin-Belegungen der Bauteile, die in diesem Buch gebraucht wurden, dargestellt. In *Abb. 9.9. a...j* sind die Timing-Diagramme des Z80 vollständig abgedruckt.

78xx

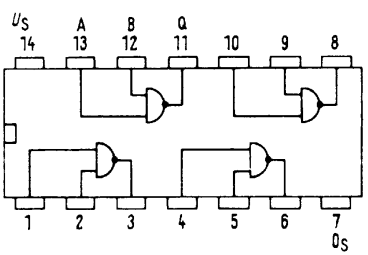


Ansicht von oben

79xx

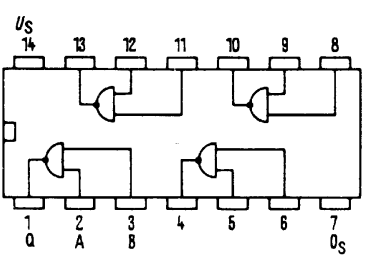


Ansicht von oben



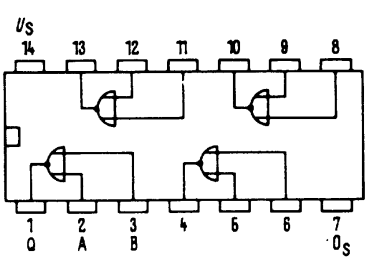
Vier NAND-Glieder mit je 2 Eingängen

7400



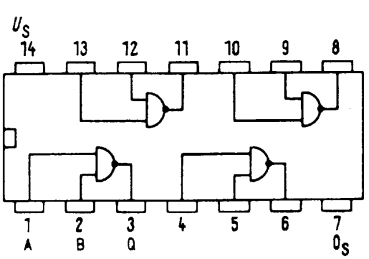
Vier NAND-Glieder mit je 2 Eingängen und offenem Kollektorausgang

7401



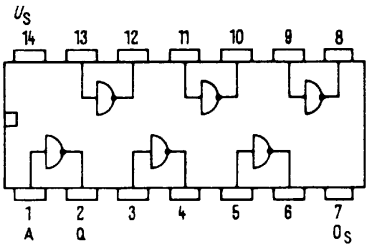
Vier NOR-Glieder mit je 2 Eingängen

7402



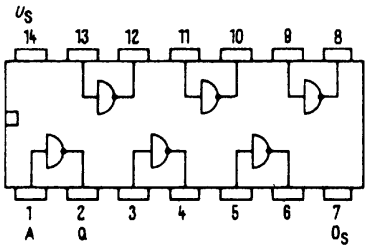
Vier NAND-Glieder mit je 2 Eingängen und offenem Kollektorausgang

7403



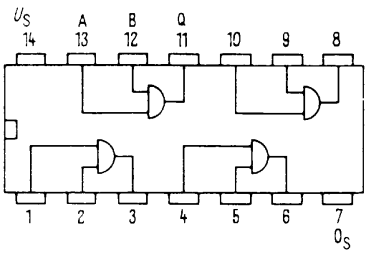
Sechs Inverter

7404



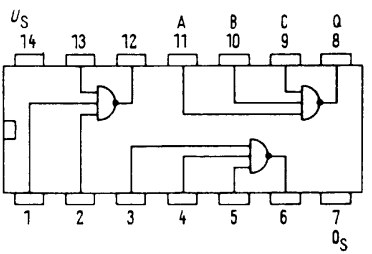
Sechs invertierende Treiberstufen
mit offenem Kollektorausgang für 40 mA

7406



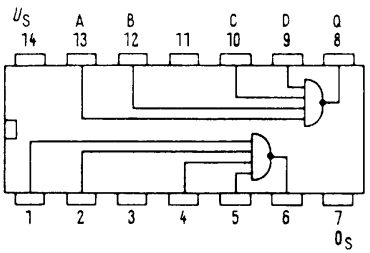
Vier UND-Glieder mit je zwei Eingängen

7408



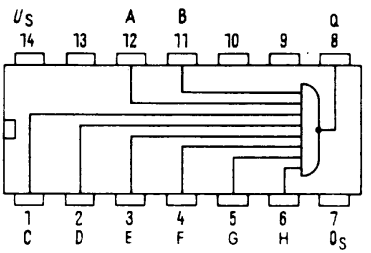
Drei NAND-Glieder mit je 3 Eingängen

7410



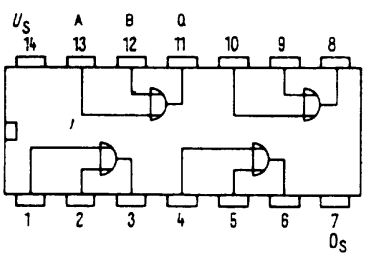
Zwei NAND-Glieder mit je 4 Eingängen

7420



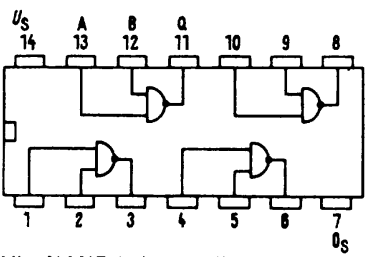
NAND-Glied mit 8 Eingängen

7430



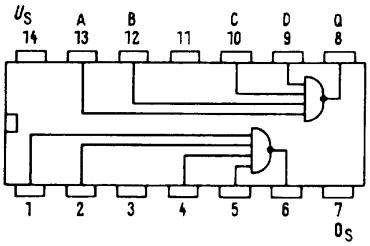
Vier ODER-Glieder mit je 2 Eingängen

7432



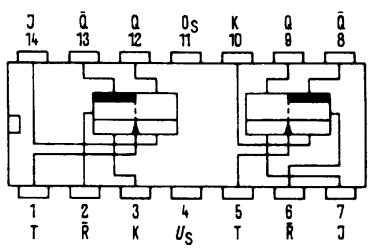
Vier NAND-Leistungsglieder
mit je 2 Eingängen und offenem
Kollektorausgang

7438



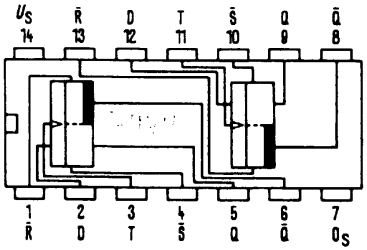
Zwei NAND-Leistungsglieder
mit je 4 Eingängen

7440



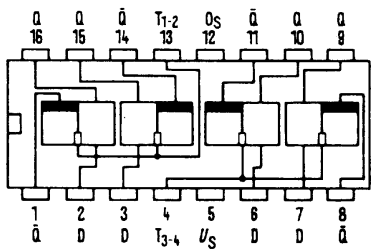
Zwei JK-Master-Slave-Flipflop
mit Rückstelleingang

7473



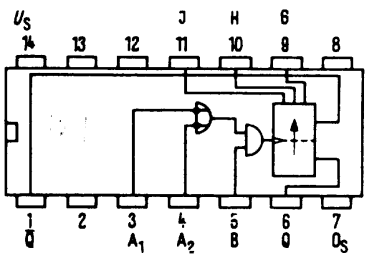
Zwei D-Flipflop

7474



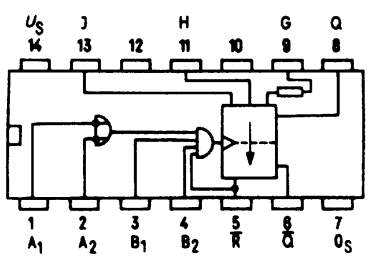
Vier D-Flipflop

7475



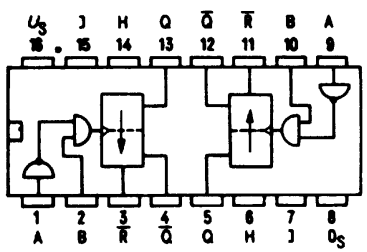
Monostabile Kippstufe

74121



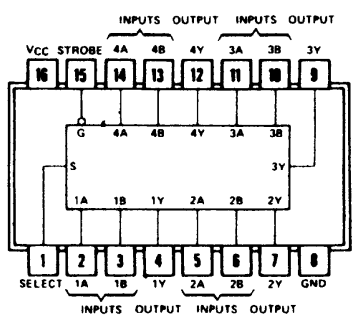
Nachtriggerbare monostabile Kippstufe
mit Rückstelleingang

74122



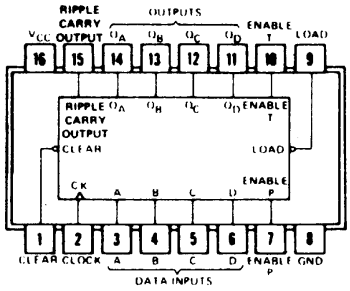
Zwei nachtriggerbare monostabile Kippstufen mit Rückstelleingang

74123

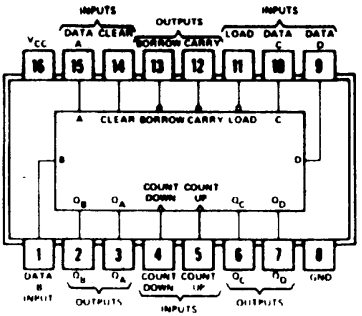


Vier 2 zu 1 Multiplexer

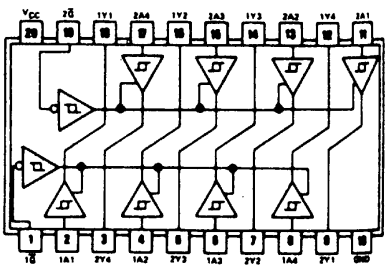
74157



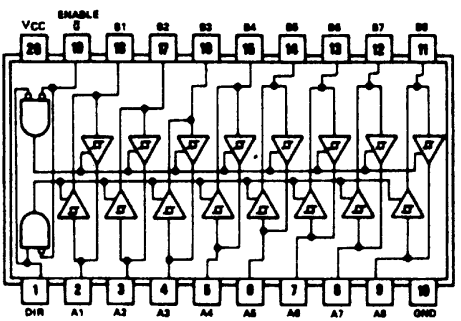
Synchroner 4-Bit Binärzähler
74161



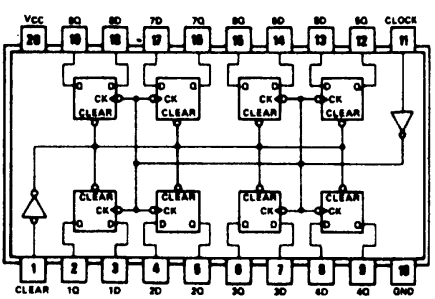
Synchroner Vor-/Rückwärtszähler
74193



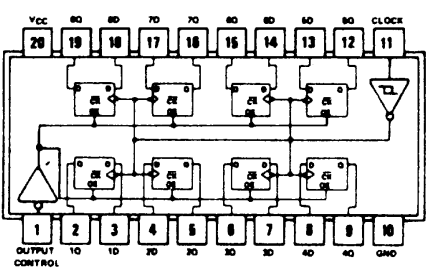
Acht Bus-Treiber
74244



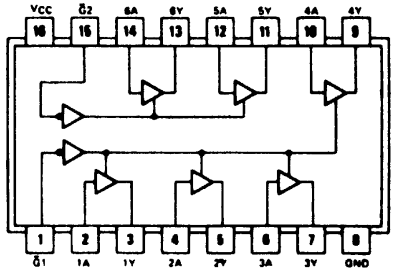
Acht Bi-direktionale Bus-Treiber
74245



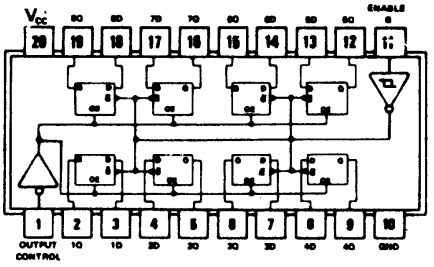
8-Bit D-Register
74273



8-Bit D-Latch
74373

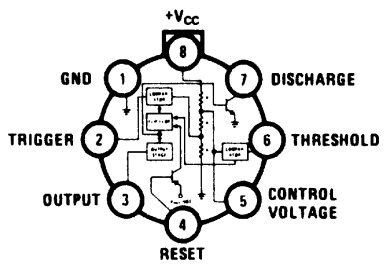


Sechs Bus-Treiber
74 367



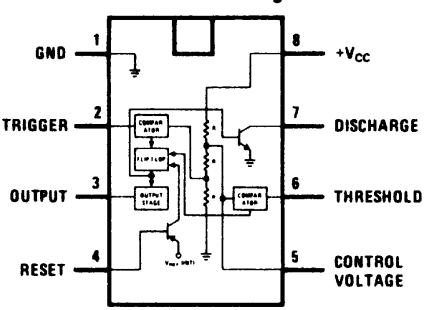
8-Bit D-Register Tristate-Ausgänge
74 374

Metal Can Package

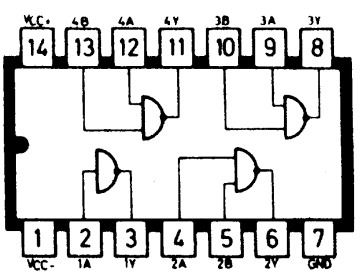


TOP VIEW
Order Number LM555H or LM555CH

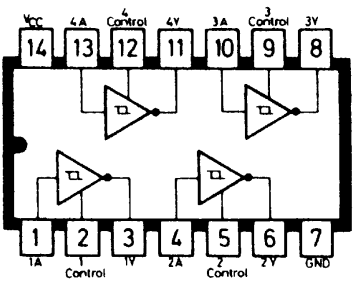
Dual-In-Line Package



TOP VIEW
Order Number LM555CN

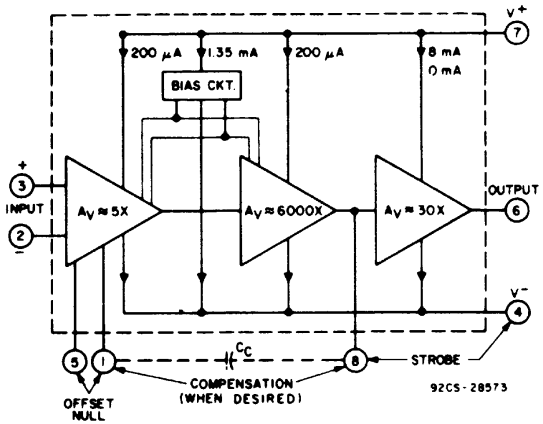


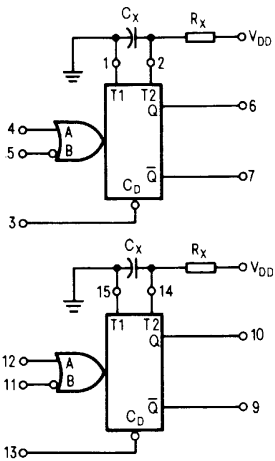
75188



75189

Abb. 9.1 Verschiedene Pin-Belegungen

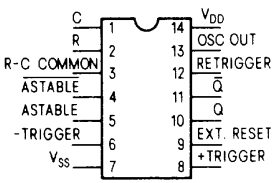




R_X and C_X are external components

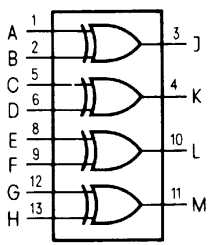
V_{DD} = Pin 16

V_{SS} = Pin 8



TOP VIEW

Terminal Diagram



$J = A \oplus B$ $M = G \oplus H$

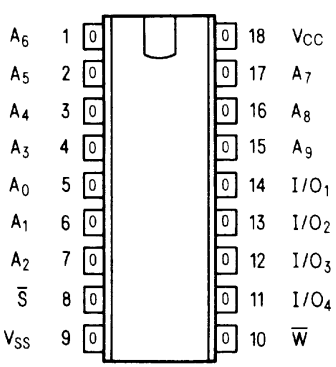
$K = C \oplus D$ $L = E \oplus F$

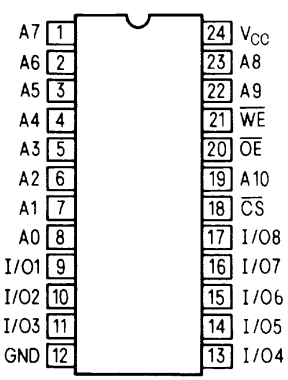
$V_{SS} = 7$

$V_{DD} = 14$

CD 4070 B
FUNCTIONAL DIAGRAM

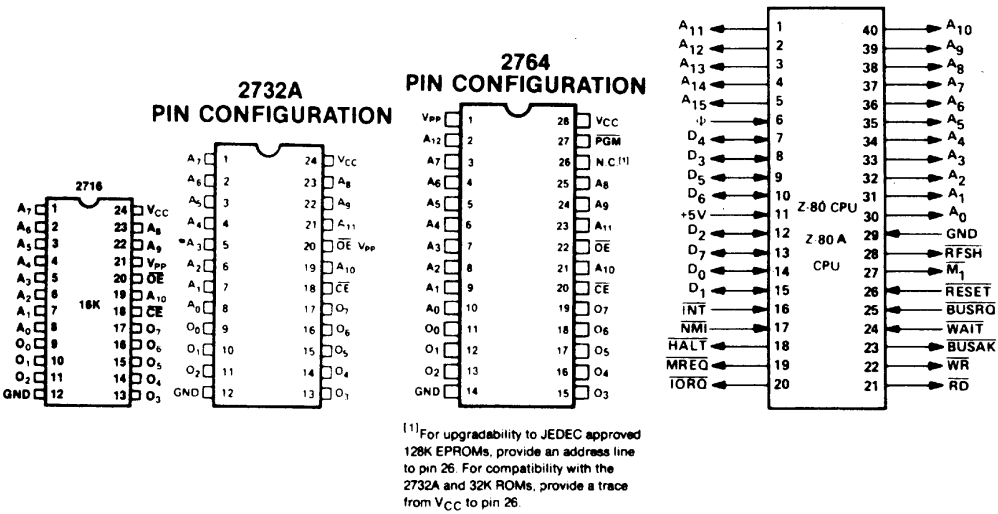
TMS 4045
18-PIN CERAMIC AND PLASTIC
DUAL-IN-LINE PACKAGES
(TOP VIEW)





(Top View)

6116



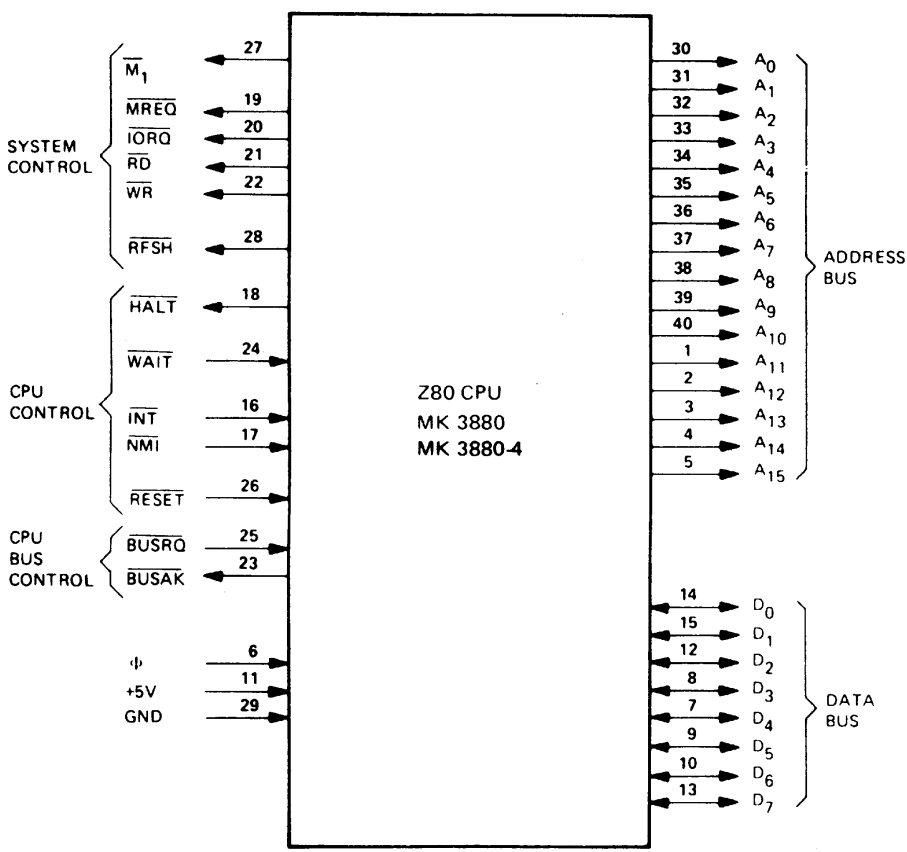
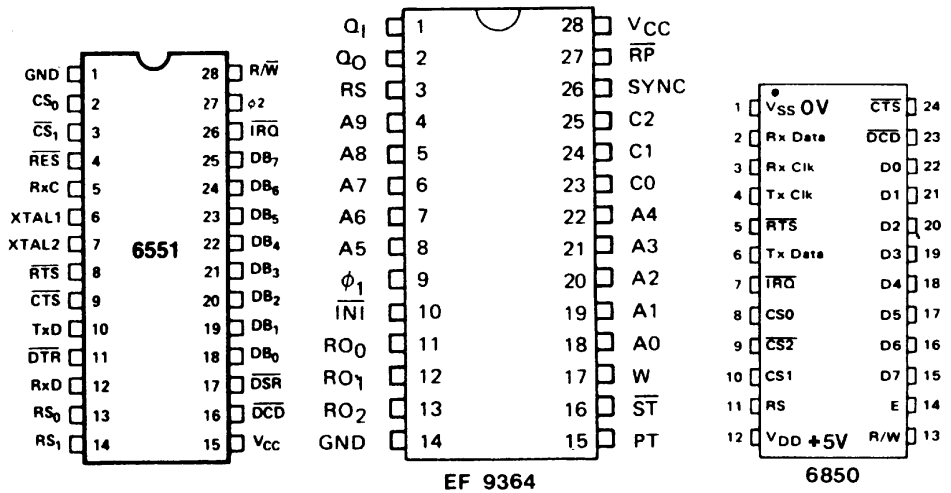
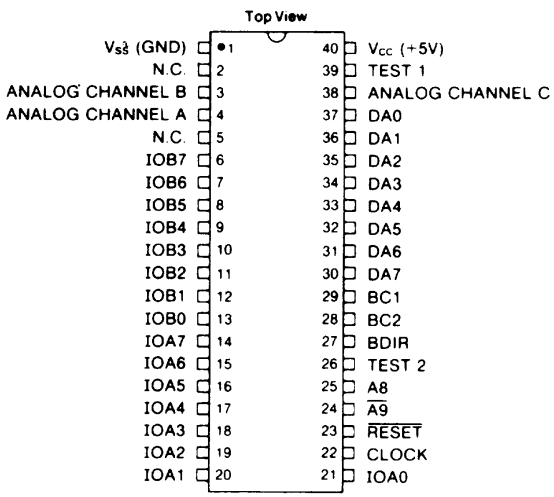


Abb. 9.1 Verschiedene Pin-Belegungen



PIN CONFIGURATIONS

40 LEAD DUAL IN LINE
AY-3-8910



28 LEAD DUAL IN LINE
AY-3-8912

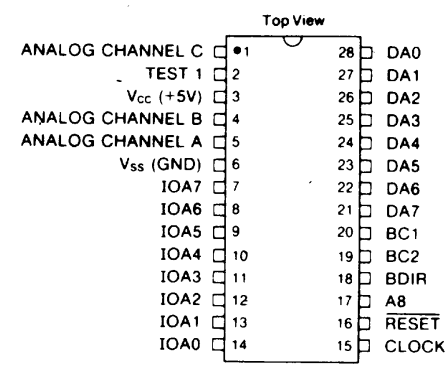


Abb. 9.1 Verschiedene Pin-Belegungen

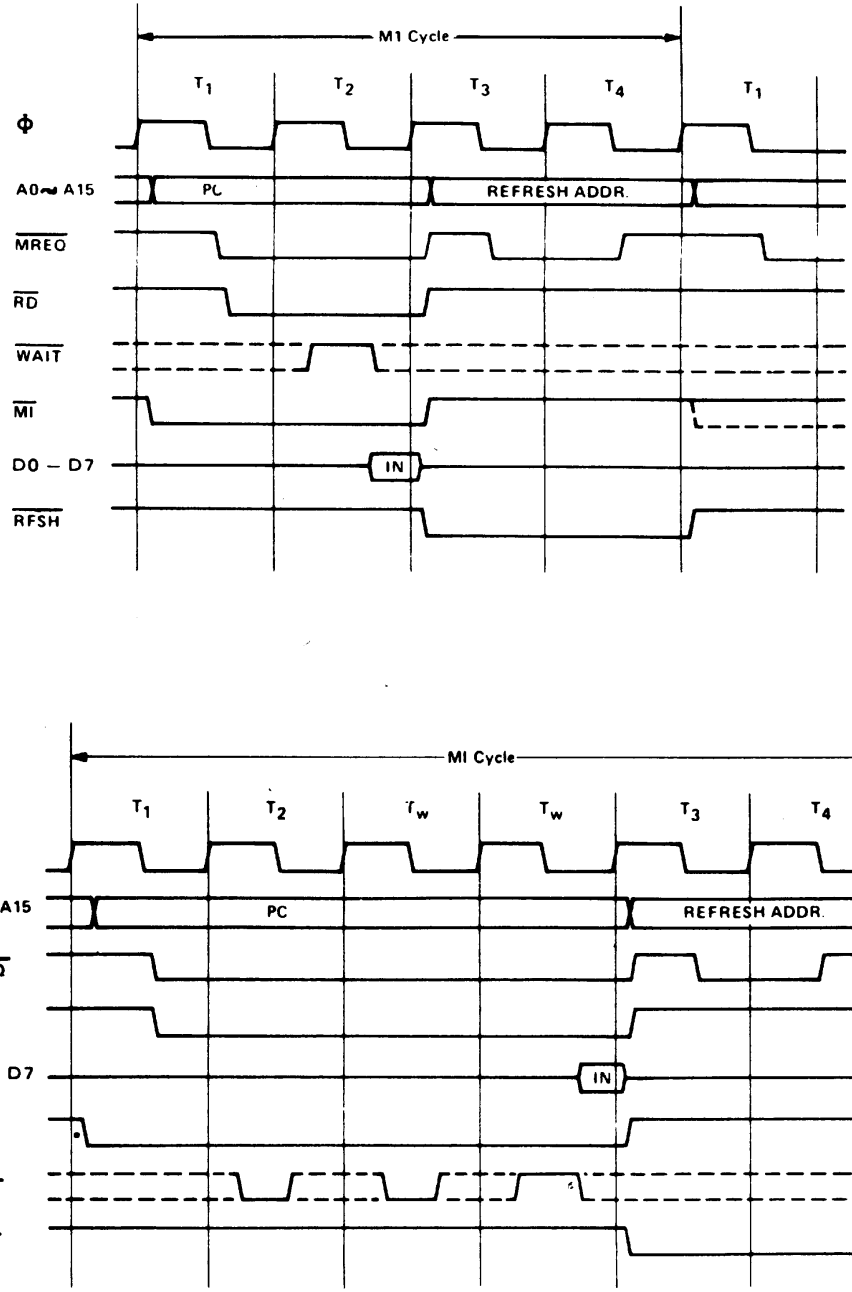


Abb. 9.2 Timingdiagramme zum Z80

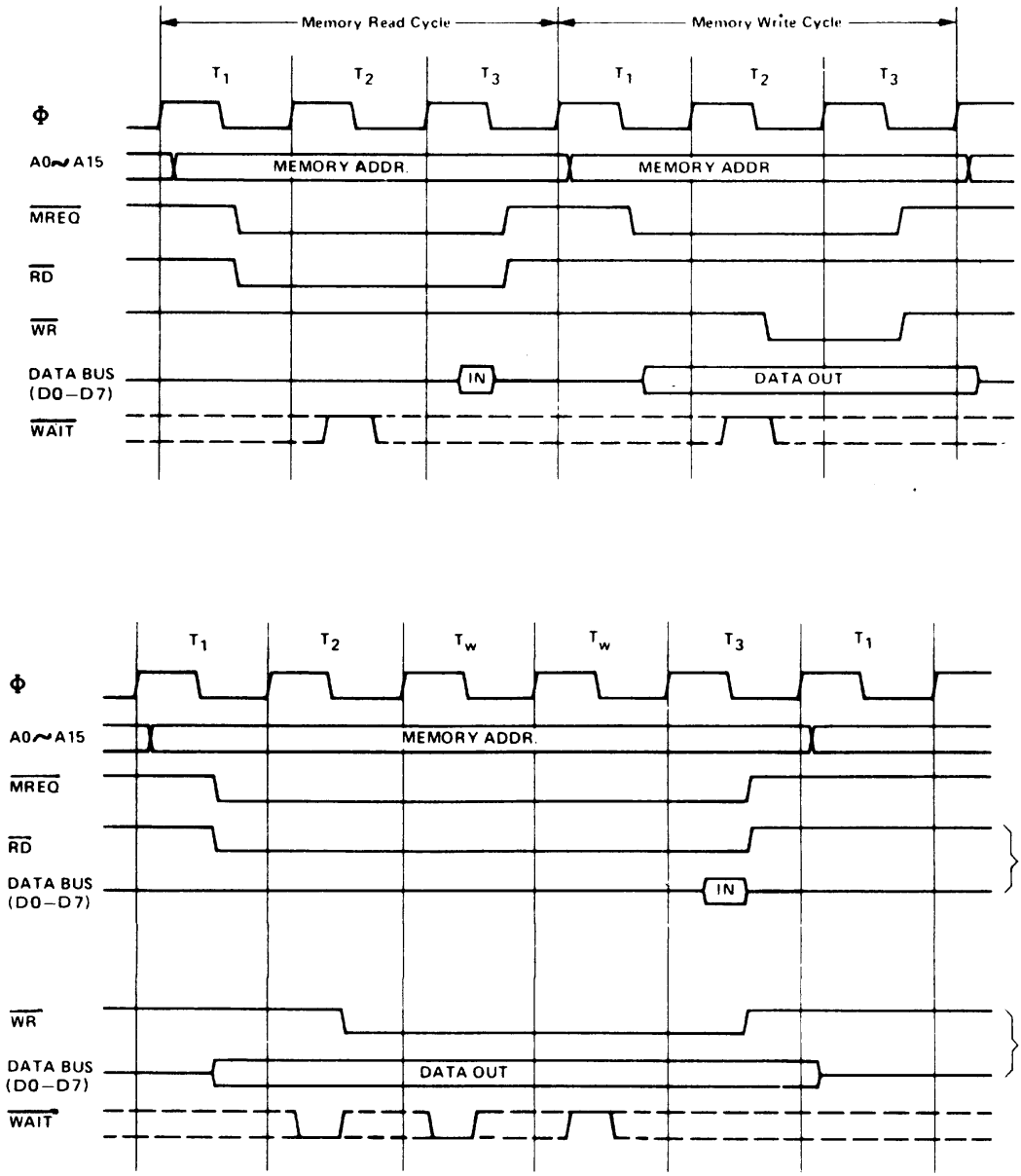
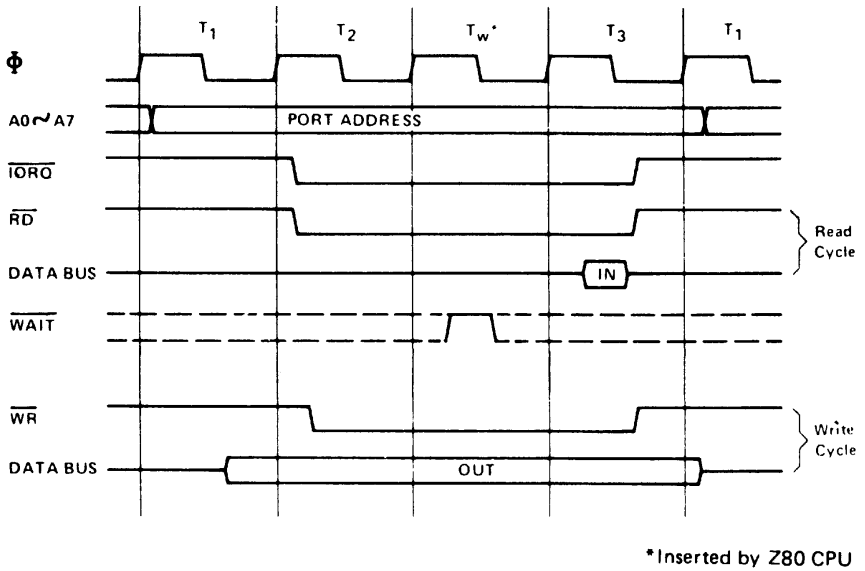


Abb. 9.2 Timingdiagramme zum Z80



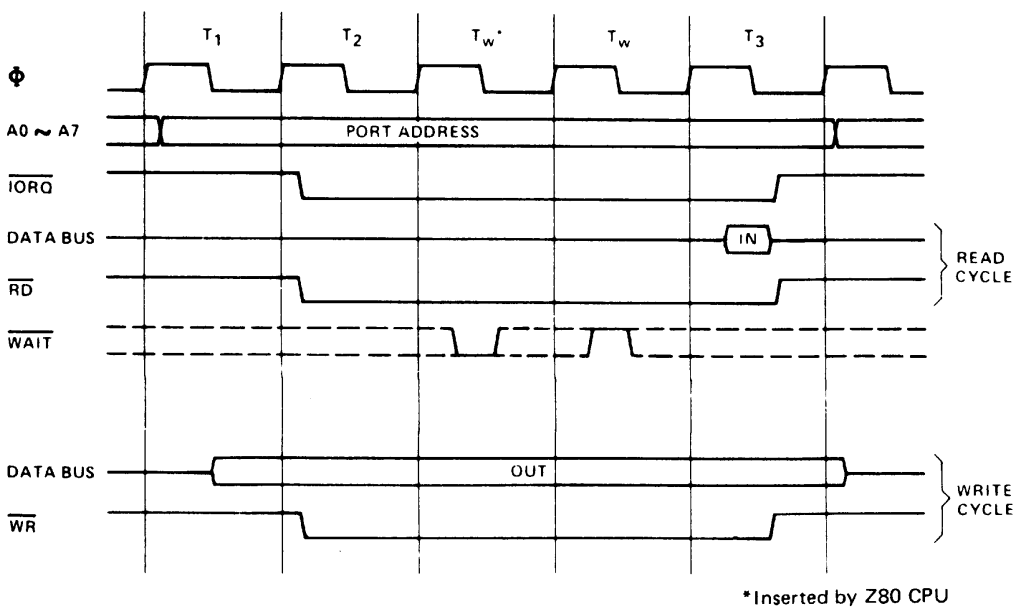
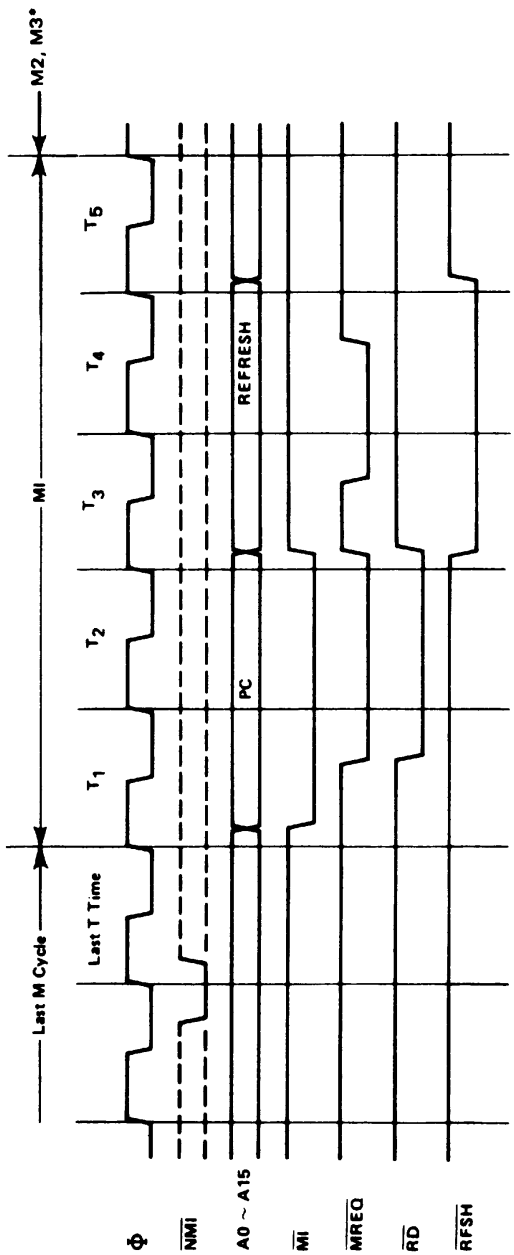


Abb. 9.2 Timingdiagramme zum Z80



*M2 and M3 are stack write operations

Abb. 9.2 Timingdiagramme zum Z80

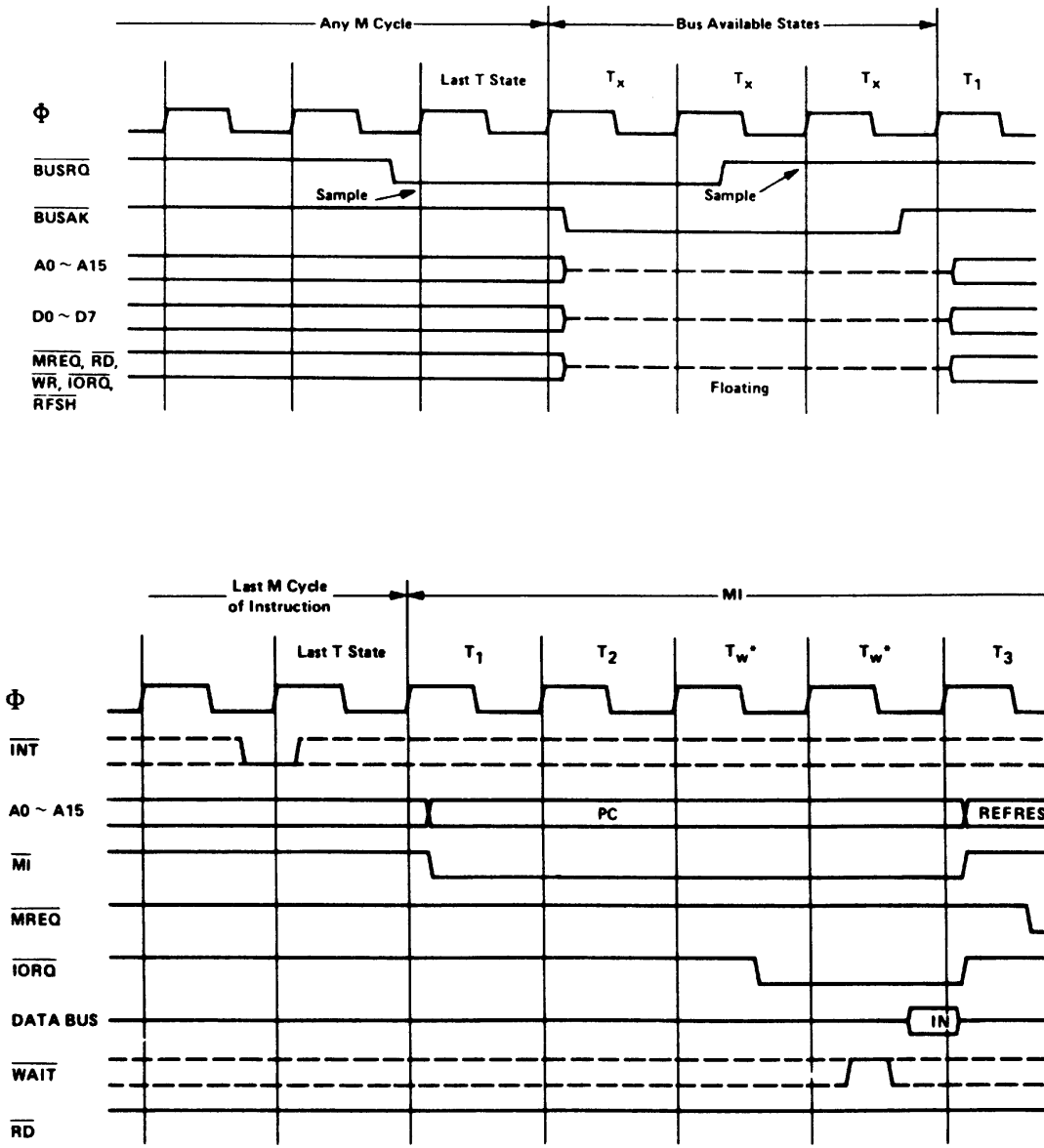


Abb. 9.2 Timingdiagramme zum Z80

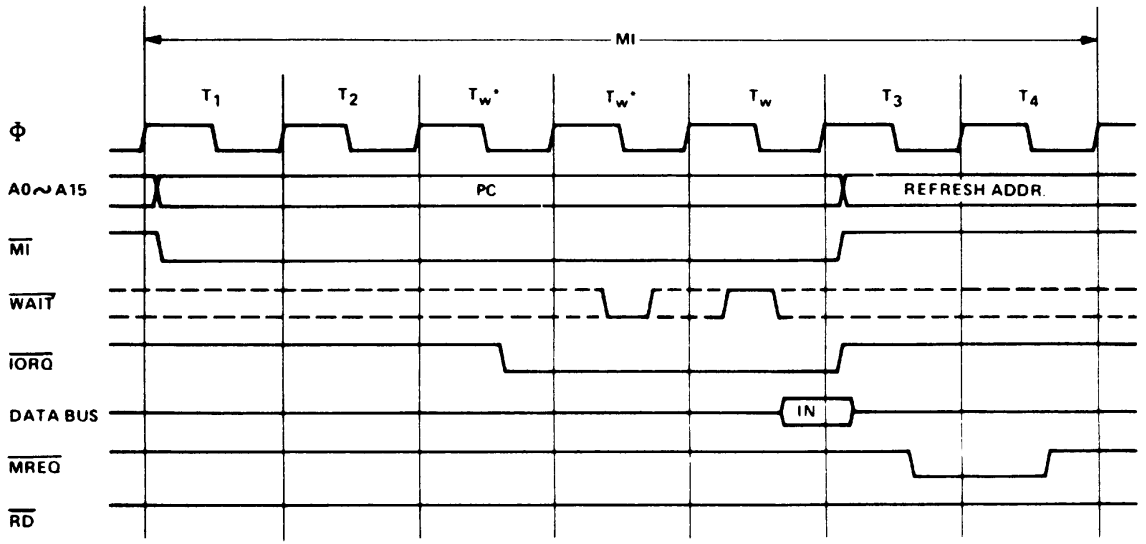


Abb. 9.2 Timingdiagramme zum Z80

10 Literaturverzeichnis

Bücher

[1] ZILOG Z80-Datenbuch. Vertrieb durch Kontron München.
[2] Z80-Assembler Handbuch. Vertrieb curch Kontron München.
[3] *Klein, Rolf-Dieter.* Mikrocomputer entwickeln mit Moduln. Franzis-Verlag, München.
[4] *Klein, Rolf-Dieter.* Mikrocomputersysteme, Franzis-Verlag.
[5] *Klein, Rolf-Dieter.* Mikrocomputer Hard- und Softwarepraxis. Franzis-Verlag. München.
[6] *Klein, Rolf-Dieter.* Basic-Interpreter. Franzis-Verlag.
[7] *Klein, Rolf-Dieter.* Was ist Pascal. Franzis-Verlag.
[8] *Klein, Rolf-Dieter.* Von der Siebensegmentanzeige zum Datensichtgerät. Franzis-Verlag, München.
[9] *Klein, Michael.* Z80-Applikationsbuch. Franzis-Verlag, München.
[10] *Leventhal, Lance A.* Z80 Assembly Language Programming. Osborne/McGRAW-Hill. Berkeley, California.
[11] *Blomeyer-Bartenstein H.-P.* Mikrocomputertechnik. Ing. W. Hofacker GmbH Verlag. München.
[12] *Wirth, N.* Systematisches Programmieren. Teubner Studienbücher. Stuttgart.
[13] *Wirth, N.* Algorithmen und Datenstrukturen. Teubner Studienbücher, Stuttgart.
[14] *Feichtinger, Herwig.* Mikrocomputer von A bis Z. Franzis-Verlag, München.
[15] *Feichtinger, Herwig.* Basic für Microcomputer. Franzis-Verlag. München.

Zeitschriften

[1] MC. Franzis-Verlag. München.
[2] ELEKTRONIK. Franzis-Verlag, München.
[3] BYTE. Byte Publications Inc. Peterborough.
[4] Dr. Dobbs Journal. Menlo Park.
[5] *Klein, Rolf-Dieter.* Längenbestimmung von Z80-Befehlen. ELEKTRONIK, Heft 23, S85..87.1980.
[6] *Klein, Rolf-Dieter.* Basic für 8080-Systeme. Hobbycomputer Sonderheft 1, Franzis-Verlag, München.

11 Bezugsquellenverzeichnis

Platinen und Bauteile der hier vorgestellten Schaltungen:

Graf Elektronik Systeme GmbH
Magnusstraße 13
Postfach 1610
8968 Kempten
Tel. 0831/61930

Z80-Bausteine

Zilog
Vertrieb z. B. Kontron München

Mostek
Vertrieb z. B. DEMA München

Soundgenerator von *General Instrument*
München

Thomson-Bausteine
Vertrieb z. B. Metronik München

TTL-Bausteine
Texas Instruments
Vertrieb über viele Fachgeschäfte

12 Terminologieverzeichnis

A

Abblockkondensator
Dient zur Unterdrückung von Störungen auf der Versorgungsspannung

Access
Zugriff

Adresse
Eine Bezeichnung für einen bestimmten Speicherplatz oder eines Speicherbereiches

Adreßbus
Ein Bus auf dem die Adressen eines Mikroprozessors geführt werden

Akkumulator
Ein Register über das arithmetische und meist auch logische Befehle ausgeführt werden können. Der Akkumulator hat direkte Verbindung mit dem Rechenwerk

ALGOL
Algorithmic Language. Es handelt sich um eine Programmiersprache für technisch wissenschaftliche Probleme

ALU
Arithmetic Logic Unit. Rechenwerk. In diesem Teil des Rechners werden die arithmetischen und logischen Verknüpfungen ausgeführt.

APL
A Programming Language. Eine Programmiersprache für den technisch wissenschaftlichen Bereich

APU
Arithmetic Processur Unit

ASCII-Satz
American Standard Code for Information Interchange. Wird auch mit ISO-7-Bit Code bezeichnet (DIN 66003)

Assembler
Ein Übersetzungsprogramm, das eine maschinennahe Programmiersprache in den Maschinencode übersetzt

B

Bankselekt
Ein solchen Siglan dient der Erweiterung des Adreßbereichs eines Prozessors

BAS-Mischer
Aus den Synchronsignalen und dem Videosignal wird ein genormtes Signal zur Ansteuerung eines Videomonitors erzeugt

BASIC
Beginners All Purpose Symbolic Instruction Code. Eine einfache Dialogorientierte Programmiersprache

Baudrate
Messung des Datenflusses wobei die Zeit zur Übertragung des kürzesten Elementes als Maß genommen wird

Baudrategenerator
Ein Baustein zur Erzeugung des Grundtaktes für die serielle Übertragung – ist beim 6551 mit eingebaut

Befehlsformat
Anordnung der Befehlsteile in einem Befehlswort

Betriebssystem
Eine Reihe von Programmen, die es dem Computer ermöglichen, selbstständig Programme zu bearbeiten, CP/M ist z. B. ein einfaches Betriebssystem für Mikrorechner

Bi-Direktionale-Bustreiber
Ein Schaltkreis der logische Informationen auf einer Leitung in beiden Richtungen übertragen kann

Bildwiederholtspeicher
Bei einem Datensichtgerät sind darin die darzustellenden Zeichen abgelegt

Bit
Binary Digit. Kleinste Informationseinheit

Boot
Das Neustarten eines Systems bei dem Daten geholt werden müssen, wird auch als Boot bezeichnet

Branch
Verzweigung

Brumm
Bezeichnung für eine meist vom Netz verursachte Störung (50 Hz)

Buffer
Puffer. Speicher in dem Daten kurzzeitig festgehalten werden oder auch Treiber zum Schalten von größeren Lasten

Bus
Sammelleitung, an die mehrere Bausteine angeschlossen werden können

Bus
Eine Gruppe von Signalen ist zusammengefaßt und kann von mehreren Einheiten angesprochen werden

Bus-Schaltkreise
Dienen zur Ansteuerung von Bus-Systemen

BUSAK
Busacknowledge. Rückmeldung von dem Z80 das er die BUSRQ Anforderung angenommen hat

BUSRQ
Busrequest. Bus-Anforderung

Busy
Besetzt. Belegt. Beschäftigt.

Byte
8 Bits

C

Eine Programmiersprache die zur Systemprogrammierung eingesetzt wird. Sie ähnelt PASCAL

Clock
Takt

CMOS-RAM, 52
Ein RAM-Baustein mit sehr niedrigem Stromverbrauch

COBOL
Common Business Oriented Language. Eine Programmiersprache für vorwiegend kaufmännische Probleme

Compiler
Ein Übersetzungsprogramm, das eine höhere Programmiersprache in den Maschinencode übersetzt

Conditional
Bedingt

Controller
Steuereinheit

CP/M
Disk Operating System für 8080, Z80, 8086, 68000 von DIGITAL RESEARCH

Cross-Assembler
Ein Assembler, der nicht auf der Maschine läuft, für die er den Maschinencode erzeugt

Cross-Compiler
Ein Compiler, der nicht auf der Maschine läuft, für die er den Maschinencode erzeugt

CRT
Cathode Ray Tube. Datensichtgerät oder Datensichtgeräte-Bausteine

Cursor
Sichtmarke zur Kennzeichnung der aktuellen Schreibposition z. B. in Datensichtgeräten

D

Datenbus
Ein Bus auf dem die Daten eines Mikroprozessors geführt werden

Debugging
Wörtlich „entwanzen". Fehlersuche und Beseitigung

Decrement
erniedrigen

Digit
Ziffer, Stelle

DIL
Dual In Line – Gehäuseform

Dil-Schalter
Miniaturschlater in der Baugröße eines ICs (DIL-Format)

Dioden-Matrix, 32
eine kreuzförmige Anordnung zweier Schienen, die durch Dioden verbunden werden

Direktory
Inhaltsverzeichnis: z. B. von einer Floppy Disk

DMA
Direct Memory Access. Direkter Zugriff auf den Speicher eines Rechners, wobei die Zugriffssteuerung von einer Peripherieeinheit vorgenommen wird

DOS
Disc Operating System. Ein Programm, das es ermöglicht, mit einer Floppy oder einem Plattenspeicher zu arbeiten (z. B. CP/M).

dynamische Speicher
ein Speicher, bei dem ein ständiger refresh ausgeführt werden muß, damit keine Information verloren geht

E

EDITOR
Ein Programm, das die Eingabe von Text erlaubt

EEPROM
electrical Erasable Programmable Read Only Memory

Emulation
Softwaremäßige Nachbildung eines Computers, so daß der Befehlssatz des einen Computers auf einem anderen verfügbar wird

Enable
Freigabe

enter
eingeben

EPROM
Erasable Programmable Read Only Memory. Ein löschbarer Nur-Lese-Speicher

erase
Löschen

Error
Fehler. Irrtum

Europakarte
Leiterplatte mit genormtem Format: 100 mm x 160 mm

Even
bedeutet gerade

Expression
Ausdruck

Fan-in
Eingangslastfaktor

Fan-out
Ausgangslastfaktor. Er gibt an, wieviele Bausteine der gleichen Logikserie an diesem Ausgang angeschlossen werden dürfen

Festwertspeicher, 37
Ein Speicher, dessen Inhalt nicht (oder nur mit Mühe) geändert werden kann

Fifo
First In First Out. Zuerst eingehende Daten werden auch zuerst wieder ausgegeben

File
Datei. Daten. Eine Ansammlung von Datengruppen, die in einer Datei angelegt werden

Firmware
Eine Software, die fest zur Funktionsfähigkeit eines Systems nötig ist und z. B. in einem ROM abgelegt wird

Fixed-Point
Festkomma

Flag
Eine Marke oder ein Flip-Flop zum Festhalten eines Zustands. Merker

Flip-Flop-Schaltungen, 21
ein Schaltkreis der zwei Zustände haben kann

Floating-Point
Gleitkomma

FORTRAN
Formula Translation. Eine problemorientierte Programmiersprache für den technisch wissenschaftlichen Bereich

Frontpanel
Bedienungsfeld

FSK
Frequency shift. Verfahren bei der Aufzeichnung auf Datenträger.

G

Gate
Verknüfpungsschaltung

GND
Masseanschluß

H

HALT
Halt. Halt-Zustand

Handshake
Quittungsbetrieb. Methode um Geräte mit verschiedenen Arbeitsgeschwindigkeiten durch den Austausch von Steuersignalen zu synchronisieren

Hardcopy
Kopie. Z. B. direkter Ausdruck eines Bildschirminhalts

Hardware
Damit sind alle Bauteile, Geräte eines Systems gemeint

Hexadezimal
siehe Sedezimal

High order
höherwertige Stelle

I

ICE
In-Circuit Emulator. Gerät für den Test von Mikrorechnerentwicklungen

Increment
erhöhen

Initialisierung
Die Anfangsschritte in einem Programm um definierte Startwerte zu erhalten

Input
Eingabe

Instruktionszyklus
Ablauf eines Befehlsausführungsvorgangs

INT
Interrupt. Unterbrechungsanforderung

Interpreter
z. B. ein Programm, das Befehle einer höheren Sprache direkt ausführt und sie nicht vorher in Maschinencode übersetzt

Interrupt
Unterbrechung

IORQ
IO Request. Peripherieanforderung

J

Job
Auftrag

Joystick
Ein Kreuzknüppelpotentiometer, das zur Eingabe von geometrischen Daten verwendet werden kann

Jump
Sprung

K

Keyboard
Tastatur

Kit
Bausatz

Kompatibel
Austauschbar, aneinander angepaßt

L

Label
Marke. In Programmiersprachen ist damit meist eine symbolische Adresse gemeint

Leuchtdiode, 15
eine Diode, die bei Beschaltung in Flußrichtung leuchtet

Lichtgriffel
Ein Stift mit einem optischen Aufnehmer, der zur Eingabe von geometrischen Daten dient

Lifo
Last in First Out. Zuletzt gespeicherte Daten werden zuerst ausgegeben. Stack

Linker
Ein Programm, das mehrere Teilprogramme, die schon assembliert wurden und spezielle Information enthalten, zu einem binden kann.

LISP
List Processing. Eine Programmiersprache für die Verarbeitung von Listenstrukturen und rekursiver Technik für Probleme der künstlichen Intelligenz

Listing
Ausdruck. Auflistung

Loader
Ein Ladeprogramm

Logic analyzer
Ein Gerät zum Testen von Digitalschaltungen und Mikrorechnern

Loop
Schleife. Durch einen Sprung zurück kann eine Schleife entstehen

Low order
niederwertige Stelle

M

M1
Maschine Cycle One. Dort wird beim Z80 der Befehlscode vom Speicher geholt

Mark
Bei der seriellen Übertragung ist damit der Logisch Wert 1 gemeint

Maschinenbefehl
Ein Befehl, den ein Computer direkt verstehen kann

maskieren
damit kann die Ausführung von Interrupts z. B. gestoppt oder freigegeben werden

Memory
Speicher

Mikrocomputer
Besteht aus einem Mikroprocessor. Speichern und Peripherie.

Mirkoprogrammierbar
Der Befehlssatz eines Prozessors kann mit Hilfe von Mikrobefehlen definiert werden. Nicht mit Maschinensprache verwechseln.

Mikroprogrammsteuerung
Sie legt den internen Ablauf fest

Mikroprozessor
Ein integrierter Baustein, als Teil eines Mikrocomputers, der ein Leit- und Rechenwerk besitzt

Mikrorechner
Ein Computer, der mit einem Mikroprozessor aufgebaut ist

mnemotechnische Darstellung
leicht zu merkende Abkürzung für längere Begriffe

Modem
Modulator und Demodulator. Eine Schaltung die Daten für eine Fernübertragung aufbereitet

Monoflop
Ein bistabiles Speicherelement das in einen Zustand nach einer bestimmten Zeit zurückkippt

MREQ
Memory Request. Speicheranforderung

Multimeter
Vielfachmeßgerät

Multiplex
Übertragung von mehreren verschiedenen Informationen, die dazu zeitlich hintereinander übertragen werden

Multiprozessing
Ein aus mehreren CPUs oder Teilcomputern zusammengesetzter Rechner

N

Nesting
Verschachtelung. Z. B. verschachteln von Interrupts

NMI
Von Maskable Interrupt. Eine nicht maskierbare Unterbrechung

O

Odd
bedeutet ungerade

Off-line
Die Kommunikation mit dem Computer wird über einen Datenträger abgewickelt

offener Kollektor
Schaltung, dessen Endtransistor einen herausgeführten unbeschalteten Kollektor hat

Oktal
Zahlendarstellung zur Basis 8

On-line
Das Terminal des Benutzers ist über eine Datenleitung direkt mit dem Computer verbunden

Oszilloscop
Meßgerät zur Wiedergabe von dynamischen Vorgängen

Output
Ausgabe

P

Packen
Dabei werden z. B. zwei Dezimalzahlen in einem Byte untergebracht

Parity
Parität. Gleichheit

PASCAL
Eine höhere Programmiersprache, die für Lehrzwecke entworfen wurde und zunehmende Verbreitung findet

PASS
Lauf oder auch Durchgang, z. B. bei einem Übersetzungsvorgang

PE-Verfahren
Phase encoding. Verfahren bei der Aufzeichnung auf Datenträger.

Pegel
Spannungsbereich

Peripherie-Geräte
Einheiten zum die mit der Außenwelt eines Computers in Verbindung treten können

PHI
Takteingang des Z80

Pipelining
Fließbandverarbeitung. An mehreren Stellen wird in kleinen Schritten gleichzeitig eine Verarbeitung vorgenommen

PL/1
Programming Language 1. Eine höhere Programmiersprache

PL/M
Ähnlich der Programmiersprache PL/1 jedoch eine Teilmenge daraus

Plotter
Ein Gerät ähnlich zu einem SY-Schreiber für die Ausgabe von graphischen Darstellungen

Pointer
Zeiger. Ein Speicherplatz, der eine Adresse enthält

Polling
Aufrufbetrieb. Aufruftechnik. Z. B. in dem Monitorprogramm wir die serielle Schnittstelle gepollt.

Port
Tor

prellen
mechanische Schalter berühren die Kontaktflächen beim Schließen oder Öffnen mehrere Male

Programm
Ist eine Folge von Anweisungen (Befehlen), die zur Lösung eines Problems dienen sollen

Programmiersprache
Eine Sprache zur Formulierung von Programmen, die automatisch (von einem Übersetzungsprogramm) in Maschinensprache übersetzt werden können

Programmspeicher
Dort ist das auszuführende Programm gespeichert

Programmzähler
er legt die Adresse der Speicherzelle des nächsten Befehls

PROM
Programmable Read Only Memory. Der Speicher kann beschrieben werden

Pseudobefehl
Eine Instruktion die nach Herstellerdefinition eigentlich nicht existiert

Pull-Up-Widerstand
ein Widerstand, z. B. nach +5V um eine offene Kollektorschaltung zu beschalten

punch
Stanzen. Lochen

Q

Quarzgenerator
Ein Oszillator bei dem zur Erzeugung der Schwingung ein Quarz verwendet wird

Queue
Warteschlange. Daten werden in einer Warteschlange angesammelt, wenn sie noch nicht verarbeitet sind

R

RAM
Random Access Memory

RD
Read Access. Lesezugriff

Reader
(Lochstreifen oder Lochkarten) Leser

Real time clock
Echtzeituhr

Real-Time
Echtzeit

Redundanz
Teil einer Nachricht, die zur eigentlichen Information nichts mehr beiträgt. Sie kann z. B. zur Fehlererkennung oder Korrektur verwendet werden

refresh
wiederauffrischen

Relokalisierbar
Ein Programm, das auf verschiedenen Speicheradressen direkt lauffähig ist, heißt relokalisierbar

RESET
Rücksetzen

Reset
Rücksetzen

RESET-Taste
Dient zum Rücksetzen und Start der CPU

RFSH
Refreh. Signal das eine gültige Refreshadresse angibt

ROM
Read Only Memory. Nur-Lese-Speicher

Run
Durchlauf

S

SBC
Abkürzung für Single Board Computer

scan
abtasten

scrollen
der Bildschirminhalt wird nach oben oder unten verschoben

sedezimal
Zahlendarstellung zur Basis 16. (0 1 2 3 4 5 6 7 8 9 A B C D E F). Häufig auch (fälschlicherweise) mit Hexadezimalsystem bezeichnet

select
auswählen

sense
abtasten

Simulator
Ein Programm, das einen Befehlssatz simuliert

Software
Hierunter versteht man alle Arten von Programmen, wie auch Texte und Informationen

Software
Hierunter versteht man alle Arten von Programmen – wie auch Texte und Information

Source
Quelle

Space
Bei der seriellen Übertragung ist damit der logische Wert 0 gemeint

Speicherbaustein
Ein Element das Informationen behalten kann

Stack
Stapelspeicher, Kellerspeicher. siehe Lifo

State
Zustand

Statement
Anweisung. Befehl

statische RAMs
Speicher, die mit zwei Transistorzellen aufgebaut sind und wie ein bistabiles Flip-Flop arbeiten

Steuerwerk
Dieser Teil des Computers kontrolliert die Ausführung sämtlicher Befehle. Er wird auch mit Leitwerk bezeichnet

Strukturierte Programmierung
Verfahrensweise um einfach zu testende und verständliche Programme zu erzeugen

Subroutine
Unterprogramm

Supervisor
Ein Organisationsprogramm

T

Tantal-Kodensator
wird in yP-Schaltungen gerne zur Unterdrückung von Störspitzen auf der Versorgung verwendet

Terminal
Datenendstation

Text-Editor
siehe Editor

Time sharing
Zeitscheibenverfahren. Dabei können mehrere Benutzer auf ein und denselben Computer zugreifen

Timing-Diagramm
Zeitlicher Ablauf bildlich dargestellt

Trace
Ablaufverfolgung. Methode zur Fehlersuche in Programmen

transfer
Übertragen

Tristate-Treiber
ein Schaltkreis der drei Zustände besitzt, Pegel auf 0, Pegel auf 1 oder offen (Pegel undefiniert)

TTL-Signal
Der Spannungsbereich bei TTL-Signalen liegt zwischen 0V und +5V

U

UART
Universal Asynchronous Receiver/Transmitter
Diese Schaltung übernimmt die Parallel/Serienwandlung für eine asynchrone Datenübertragung

Unit
Einheit, Gerät

Unterprogramm
Gleiche Befehlsfolgen, die in einem Programm mehrmals vorkommen werden im allgemeinen als Unterprogramm realisiert

UV-Lampe
dient zum Löschen von EPROMs

V

V24
Schnitstellen-Norm für serielle Signale

Valid
Gültig

Vektor Interrupt
Das auslösende Gerät liefert eine Information mit, die der CPU sagt, welche Unterbrechungsroutine ausgeführt werden soll

W

WAIT
Ein Signal, das an den Z80 gegeben werden kann, um ihr kurzzeitig anzuhalten

Wired-Or
Eine logische Verknüpfung die nur durch Leitungsführung zustande kommt

Worst case
Ungünstigster Fall

Wort
Zusammenfassung mehrerer Bits zu einer logischen Einheit, z. B. 8, 16, 32 Bits

WR
Write Access. Schreibzugriff

X

XY-recorder
Ein XY-Schreiber

Z

Z80-CPU
Ein Mikroprozessor-Baustein

Zeichengenerator
Meist ein Festwertspeicher, in dem ein Zeichensatz binär, z. B. in Matrixform abgelegt ist

Zugriff
Zugang z. B. zu einer bestimmten Speicherzelle

Zyklus
Eine Anzahl von Schritten, die wiederholt werden und im Ablauf gewisse Ähnlichkeiten aufweisen

Sachverzeichnis

Weitere Franzis-Computer-Fachbücher

Z-80-Applikationen

Eine Sammlung von gut dokumentierten Programmen, die universell anwendbar sind.
Von Michael **Klein.**
144 Seiten, 89 Abbildungen.
Lwstr-geb. DM 36.–
ISBN 3-7723-6672-4

Das Applikationsbuch ist ein Schritt in die Richtung „Lösen von Standardproblemen".
Dem Benutzer werden einige fertige Programme angeboten und erklärt, anhand derer er entweder seine eigenen Probleme verstehen und bewältigen kann, oder die er mindestens teilweise übernehmen kann.
Bewußt legt der Autor den Schwerpunkt auf die sofortige Anwendbarkeit der dargestellten Methoden und Programme. Er bietet Standardlösungen z. B. für die Ein-/Ausgabe über eine serielle Schnittstelle oder Interruptschaltung, führt Programme zur Meßwertverarbeitung an und zeigt, wie man sich nützliche Arbeitshilfen schaffen kann.

Mikrocomputersysteme

Selbstbau – Programmierung – Anwendung.
Von Rolf-Dieter **Klein.**
3., verbesserte Auflage, 159 Seiten, 134 Abbildungen und 12 Tabellen.
Lwstr-geb. DM 34.–
ISBN 3-7723-6383-0

Kaum zu glauben, daß ein Mikrocomputer im Selbstbau hergestellt werden kann! Daß dieses Vorhaben glückte, hat der Autor bewiesen. Wie ein hinreichend ausgebildeter Elektroniker das nachvollziehen kann, wird in dem Buch hier dargestellt.
Zunächst wird die Hardware geschaffen, deren Elemente zu einer funktionierenden Einheit zusammengeschlossen werden. Danach die Software. Ausführliche Programme werden vorgestellt, die Spiele und mathematische Aufgaben lösen können.

IEC-Bus

Die Funktionsweise des IEC-Bus und seine Anwendung in Geräten und Systemen.
Von Dr. Anton **Piotrowski.**
300 Seiten, 125 Abbildungen und 95 Tabellen.
Lwstr-geb. DM 48.–
ISBN 3-7723-6951-0

Hier liegt eine fachgerechte Darstellung der Funktionsweise des IEC-Bus und dessen Anwendung in den Geräten und den Systemen vor. Diese gliedert sich in drei Teile: 1. Theoretische Grundlagen der Funktion des IEC-Bus mit ausführlicher Beschreibung der Nachrichten- und der Zustandsdiagramme. 2. Beschreibung der IEC-Interface-Bausteine und deren Anwendungen bei der Realisierung von IEC-Schnittstellen mit Mikroprozessoren. 3. Gerätetechnik, Aufbau und Programmierung von IEC-Bus-Systemen, inklusiv der Fehleranalyse. Die Darstellung zeichnet sich durch eine bewundernswerte Systematik und eine klare Sprache aus. Damit ist gewährleistet, daß der in der Praxis erfahrene Techniker und Informatiker den IEC-Bus in allen seinen Varianten versteht und aufnimmt. Ausgewählte und in sich abgeschlossene praktische Anwendungsbeispiele in Hard- und Software erleichtern ihm die tägliche Arbeit.

Was ist Pascal?

Eine einfache und kompakte Darstellung der Programmiersprache mit vielen Beispielen.
Von Rolf-Dieter **Klein.**
124 Seiten, 72 Abbildungen.
Lwstr-geb. DM 28.–
ISBN 3-7723-7001-2

Dies ist eine praxisnahe Arbeitsanleitung von Anfang an mit Pascal zu programmieren. Gerade eine schrittweise Einführung beantwortet am besten und schnellsten die Frage: Was ist Pascal?
Von anderen Pascal-Büchern unterscheidet sich dieses dadurch, daß es auf die Erfordernisse der Mikrocomputer und Heimcomputer besonders eingeht.

Mikrocomputer Hard- und Softwarepraxis

Anhand ausführlicher Beispiele und größerer Programme wird das Programmieren immer perfekter.
Von Rolf Dieter **Klein.**
220 Seiten, 125 Abbildungen, 6 Tabellen.
Lwstr-geb. DM 38,–
ISBN 3-7723-6812-3

Hier geht es um den Z80, das Arbeitspferd unter den Mikrocomputern. Ihn hardwaremäßig gut zu verstehen, ihn softwaremäßig voll auszufahren, das bewirkt dieses Buch.
Der Band besteht aus drei Teilen, die deutlich miteinander verknüpft sind.
Erstens, dem Hardware-Teil. Darin bespricht der Autor die zahlreichen Peripheriegeräte, die einen Z80 erst zum vollwertigen Mikrocomputer machen. Als da sind: Serienwandler. CRT-Controler. Druckeransteuerung. A/D- und D/A-Umsetzer.
Zweitens, dem Softwareteil. Hier wird die wichtige System-Software so intensiv besprochen, daß sie dem Anwender möglichst schnell geläufig wird. Monitorprogramme. Editor. BASIC mit Abwandlungen und Vereinfachungen. Assembler. Softwaretracer in Maschinensprache. Ausgedehnte Befehlslisten, praxisorientiert, sind für diesen Buchteil charakteristisch.
Drittens, dem Programmteil. Der lehrt das freie Programmieren, wiederum mit ausführlichen Programmlisten. Diese vermitteln dem Benutzer einen großen Erfahrungsschatz, den andere eingesammelt haben.

Preisänderungen vorbehalten!

Franzis-Verlag, München

Weitere Franzis-Computer-Fachbücher

Basic für Einsteiger

Der leichte Weg zum selbständigen Programmieren.
Von Rudolf **Busch.**
239 Seiten, 32 Abbildungen.
Lwstr-geb. DM 38.–
ISBN 3-7723-7081-0

Von Anfang an wird die Programmiersprache BASIC dem Anfänger dargestellt. Anhand von zahlreichen, anregenden Beispielen werden die Sprachelemente erläutert und ihre richtige Anwendung geübt. Dem Benutzer des Buches wird vom Autor beigebracht, wie eine Problemstellung zu analysieren ist und wie sie dann Schritt für Schritt in lauffähige Basic-Programme umgesetzt werden kann. Die Beispiele, mit denen der Autor sein Ziel erreicht, sind aus dem täglichen Leben gegriffen. Hier eine knappe Themenauswahl: Der Computer als... Kaufmannsgehilfe, als Managementberater, als Textautomat, als Lagerverwalter, als Vermögensberater, als Sortiermaschine... Was will man mehr!

Mikrocomputer von A bis Z

Bits und Bytes und andere EDV-Begriffe verständlich gemacht.
Von **Herwig Feichtinger.**
176 Seiten, 34 Abbildungen.
Lwstr-geb. DM 24.–
ISBN 3-7723-7061-6

Dieses Begriffswörterbuch dient der täglichen Praxis. Es macht das unumgängliche „Fach-Chinesisch" der Computer-Branche verständlich.
Dieses Begriffswörterbuch führt zu objektiven und sachgerechten Beurteilungen der Herstellerangaben. Das spart unter Umständen die Ausgabe von einigen tausend Mark, weil es zu der richtigen Auswahl des richtigen Computers führt.
Dieses Begriffswörterbuch ist auch ein einfaches, elementares Lehrbuch der Mikrocomputertechnik. Wer von ihr angehaucht worden ist, liest die Definitionen hintereinander wie ein gut gemachtes Fachbuch und hat erheblichen Nutzen davon.

Basic-Interpreter

Funktionsweise und Implementierung in 8080/Z-80-Computern.
Von Rolf-Dieter **Klein.**
2., verbesserte Auflage. 178 Seiten, 45 Abbildungen.
Lwstr-geb. DM 36.–
ISBN 3-7723-6942-1

Wie man 8080- oder Z-80-Systeme nachträglich mit einem Basic-Interpreter ausrüsten kann, beschreibt dieses Buch. Dabei werden mehrere Ausführungen erörtert und beschrieben. Die beiden interessantesten sind: Ein Tiny-Basic-Interpreter und ein komfortabler 12-KByte-Basic-Interpreter. Die Krönung bildet die ausführliche Beschreibung eines Basic-Interpreters für den 16-Bit-Prozessor Z8000.

Pascal: Einführung – Programmentwicklung – Strukturen

Ein Arbeitsbuch mit zahlreichen Programmen, Übungen und Aufgaben.
Von **Jürgen Plate** und **Paul Wittstock.**
387 Seiten mit 178 Abbildungen.
Lwstr-geb. DM 48.–
ISBN 3-7723-6901-4

Das Buch könnte auch die Pascal-Fibel genannt werden. Schritt für Schritt führt es den Leser in das Programmieren mit Pascal ein. Die Autoren haben sich echt in die Ahnungslosigkeit des Anfängers hineinversetzt. Sie bringen ihm das besondere Denken des routinierten Programmierers bei. Das Verblüffende dabei ist, sie kommen mit einer einfachen klaren Sprache aus, verabscheuen das EDV-Chinesisch, setzen nichts voraus, können wunderbar erklären. Wer sich an dieses Buch heranmacht, meint, es gäbe nichts Einfacheres als Pascal.

Erfolgreicher mit CBM arbeiten

Für alle CBM-Anwender eine verständliche Einführung in die Maschinensprache.
Von Dipl.-Ing. Franz **Wunderlich.**
148 Seiten, 8 Abbildungen.
Lwstr-geb. DM 34.–
ISBN 3-7723-7051-9

CBM-Anwender mit Basic-Erfahrung holen mit diesem Buch mehr aus ihrem Computer heraus. Sie können nämlich mit dem Programmieren in der Maschinensprache beginnen. Zunächst wird ihnen beigebracht, wie ein 6502 programmiert wird. Zahlensysteme, Speicherkonzepte, Adressierung und Befehlssätze werden behandelt und gewinnen Klarheit. Im Hauptteil wird speziell die geräteabhängige Software der Commodore-Serie CBM abgehandelt. So wird beschrieben, wie Interpreter, Betriebssysteme, Monitor und Peripherie arbeiten. Natürlich bringt der Autor auch fertige Programme und viele CBM-spezielle Anwenderbeispiele im Anhang.

Basic für Mikrocomputer

Geräte – Begriffe – Befehle – Programme.
Von Herwig **Feichtinger.**
2., neu bearbeitete Auflage. 264 Seiten, 42 Abbildungen.
Lwstr-kart. DM 28.–
ISBN 3-7723-6822-0

Dieses praxisorientierte Buch ist Einführung und Nachschlagewerk zugleich. Begriffe aus der Computer-Fachsprache wie ASCII, RS-232-Schnittstelle oder IEC-Bus werden ebenso ausführlich erläutert wie alle derzeit üblichen Befehlsworte. Marktübliche Basic-Rechner werden einander gegenübergestellt, um vor dem Kauf die Wahl zu erleichtern und um das Anpassen von Programmen an den eigenen Rechner zu ermöglichen. Schließlich findet der Leser handfeste Tips für das Erstellen eigener Programme und Beispiele fertiger Problemlösungen für typische Anwendungsfälle.

Preisänderungen vorbehalten!

Franzis-Verlag, München

Klein
Mikrocomputer selbstgebaut und programmiert

Rolf-Dieter Klein

Vier gewichtige Punkte unterscheiden dieses Buch von den herkömmlichen Titeln über die Mikrocomputerei:

1. Die Hardware wird durch ein Experimentalsystem, bestehend aus kleinen, selbst zu bauenden Modulen im wahrsten Sinne des Wortes begriffen.
2. Alle Schaltungen sind mit einer ausführlichen Aufbau- und Testanleitung, einschließlich der Platinenvorlage, ausgestattet.
3. Die Software-Unterrichtung erfolgt mit genau auf den Z-80 abgestimmten Programmierschritten. Kein Befehl zuviel, keiner zu wenig.
4. Die konsequente Systematik und die pädagogisch sinnvollen Kontrollfragen machen den Band zu einem vorzüglichen Lehr- und Arbeitsbuch.

Wer das Buch durchgearbeitet hat, der hat die Computer-Grundlagen in sich aufgenommen und parallel dazu einen leistungs- und ausbaufähigen Mikrocomputer selbst gebaut.

ISBN 3-7723-7161-2

Franzis'