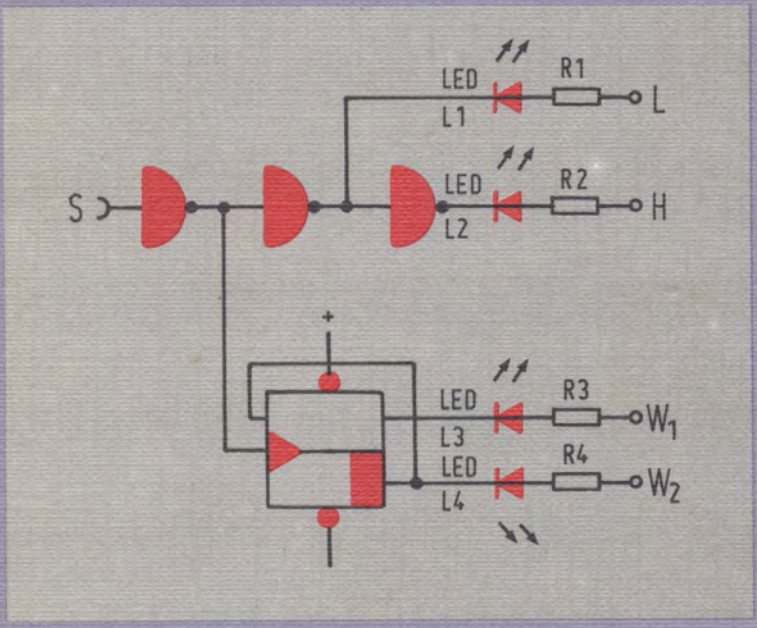
Franzis
Computer-Praxis

Klein
Mit HEXMON Programme entwickeln

Schaltungsaufbau, Befehle, Unterprogramme und Listings



Inbetriebnahme von HEXMON
Die SBC II-Karte
HEXMON-Befehle
6 spe – auf Kassette speichern
A prop – Eprom programmieren
start – Programme starten
E pul – Pulsbreite messen
F umw – Zahlensysteme umwandeln
BEF – Befehle eingeben
Unterprogramme für den Anwender
HOLETASTE
TONUM
TOSEG
PRINT
PRTHL
PRTBIN
GETDEZ

Franzis'

Klein
Mit HEXMON Programme entwickeln

In der Reihe
Franzis Computer-Praxis
sind erschienen:

Andersen/Zirpel, Die Programmierpraxis der technischen-naturwissenschaftlichen Taschenrechner
Busch, Basic für Aufsteiger
Busch, Basic für Einsteiger
Esders, Das zum Buch zum Apple II
Feichtinger, Mit Computern steuern
Haugg, Software-Engineering
Klein M./Klein R.D., Z-80 Applikationsbuch
Klein R.D., Basic-Interpreter
Klein R.D., Mikrocomputer Hard- und Softwarepraxis
Klein R.D., Mikrocomputer selbstgebaut und programmiert
Klein R.D., Mikrocomputersysteme
Klein R.D., Was ist Pascal?
Link, Messen, Steuern und Regeln mit Basic
Piotrowski, IEC-Bus
Plate, Betriebssystem CP/M
Plate/Wittstock, Pascal: Einführung – Programmentwicklung – Strukturen
Troitzsch, Mikrocomputer-Schaltungstechnik
Wunderlich, Erfolgreicher mit CBM arbeiten

Franzis Computer-Praxis

Rolf-Dieter Klein

Mit HEXMON Programme entwickeln

Schaltungsaufbau, Befehle, Unterprogramme und Listings

Mit 47 Abbildungen

Franzis'

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Klein, Rolf-Dieter
Mit HEXMON Programme entwickeln: Schaltungsaufbau, Befehle, Unterprogramme u. Listings / Rolf-Dieter Klein. – München: Franzis, 1985.
(Franzis Computer-Praxis)
ISBN 3-7723-7831-5

Satz: Grafikteam W. Meyer, 8000 München
Druck: Hablitzel & Sohn GmbH, 8060 Dachau

Printed in Germany · Imprimé en Allemagne

ISBN 3-7723-7831-5

Vorwort

Um einen Mikrocomputer zu programmieren, braucht es nicht immer große Systeme mit Bildschirmen und Text-Eingabetastaturen, sondern man kann mit einer kleinen Anzeige und einem nur wenige Tasten umfassenden Eingabefeld auskommen. Damit läßt sich dann sehr preiswert in die Thematik „Mikrocomputer" einsteigen, aber es ist auch möglich, damit kleinere Entwicklungen durchzuführen.

Dazu wurde das Programm HEXMON und die Baugruppe HEXIO geschaffen. Damit kann der NDR-KLEIN-COMPUTER auch schon mit wenigen Elementen aufgebaut werden.

HEXMON ist ein Programm, das den Computer so steuert, daß er Befehle annimmt.

Die Befehle werden über eine Tastatur eingegeben, und die Ergebnisse werden auf einer 8-stelligen Anzeige ausgegeben. Die Tastatur und die Ausgabe sind zusammen auf einer Leiterplatte untergebracht, HEXIO genannt. Mit nur fünf Karten (SBCII + IOE + HEXIO + BUS + POW5V) kann ein vollständiger Mikrocomputer aufgebaut werden, der auch frei programmierbar ist, also für den man selbst Programme schreiben kann. Alle zum Aufbau des Computers nötigen Baugruppen werden hier mit allen Schaltungen komplett beschrieben.

HEXMON ist so ausgelegt, daß man natürlich auch einen Kassettenrekorder anschließen kann (mit Baugruppe CAS), um seine Programme speichern zu können, oder daß man einen Eprom-Programmierer verwenden kann (mit Baugruppe PROMMER + POW22/26), um entworfene Programme dauerhaft festzuhalten.

So kann man mit HEXMON Programme zur Steuerung von Anlagen entwickeln (Alarmanlage, Eisenbahn, Roboter, Musik) und damit eine Vielzahl von Aufgaben erledigen.

München, Juli 1984 Rolf-Dieter Klein

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 Inbetriebnahme von HEXMON

Folgende Elemente werden benötigt:

1x SBCII-Bausatz
1x IOE-Bausatz
1x HEXIO-Bausatz
1x HEXMON-EPROM
1x BUS-Bausatz
1x POWSV-Bausatz

Tab. 1-1: HEXMON-Bausatz

Als Ausbau wird empfohlen:

1x CAS-Bausatz

Tab. 1-2: Kassettenschnittstelle

Und wer Eproms selber programmieren will:

1x PROMMER-Bausatz + POW22/26

Tab. 1-3: Eprom-Programmierung

Alle anderen Baugruppen des NDR-Klein-Computers können natürlich ebenfalls eingesetzt werden.

Die Bauanleitungen sowie alle Schaltpläne für die einzelnen Baugruppen finden sich im Anhang dieses Buches. Nachfolgend wird die Inbetriebnahme beschrieben, wenn alle Karten fertig bestückt vorliegen.

1.1 Die SBCII-Karte

Das Programm HEXMON ist in einem EPROM untergebracht. Dieses EPROM wird in die SBCII-Karte, Sockel 0 gesteckt. Als RAM genügt zum Betrieb ein einziger Baustein

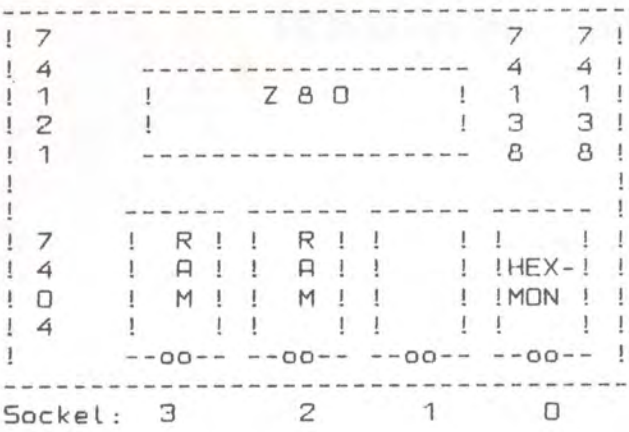


Abb. 1-1:
SBCII-Karte mit HEXMON und RAMs

in Sockel 2, jedoch ist es besser, wenn man mehr Platz zum Programmieren besitzt und die SBCII-Karte gleich mit zwei RAM-Bausteinen (6116 oder ähnliche) versieht.

Beim Einsetzen des EPROMs mit dem Programm HEXMON muß unbedingt auf die Orientierung geachtet werden, wie bei allen anderen ICs auch.

1.2 Die IOE-Karte

Die IOE-Karte wird voll bestückt verwendet. Auf der IOE-Karte befinden sich vier Brücken zum Einstellen einer Adresse. Alle vier Brücken werden eingelötet, da dies zum Betrieb von HEXMON nötig ist. Die Karte besitzt dann einen Adreßbereich von 0 bis F (sedezimal), d.h. sie kann mit dem Z80 durch geeignete Befehle auf diesen Adressen angesprochen werden.

Abb. 1-2:
Einlöten der Brücken auf der IOE-Karte

Da die IOE-Karte mit der HEXIO-Karte verbunden werden muß, muß Stiftleisten in die Rasterlöcher eingelötet werden. Dazu werden zwei 50-polige Stiftleisten verwendet. Diese Stiftleisten werden direkt in das Anschlußfeld gesteckt, an das die Treiberbausteine (74LS245, 74LS374) angeschlossen sind. Dabei werden auch einige leere (kein IC ist dort angeschlossen) Rasterplätze belegt.

Es bleibt aber kein Loch unter oder über der Stiftleiste frei, da die beiden nebeneinanderliegenden Lochreihen exakt je 50 Löcher besitzen.

Man muß noch eine Leitung zusätzlich anlöten. Diese wird auf der Lötseite der Leiterplatte, wie nachfolgende Abb. zeigt, angelötet: Einmal wie eingezeichnet an die zwei Stifte der 50-poligen Stiftleiste und dann noch an die +5V-Versorgung. Die Abb.

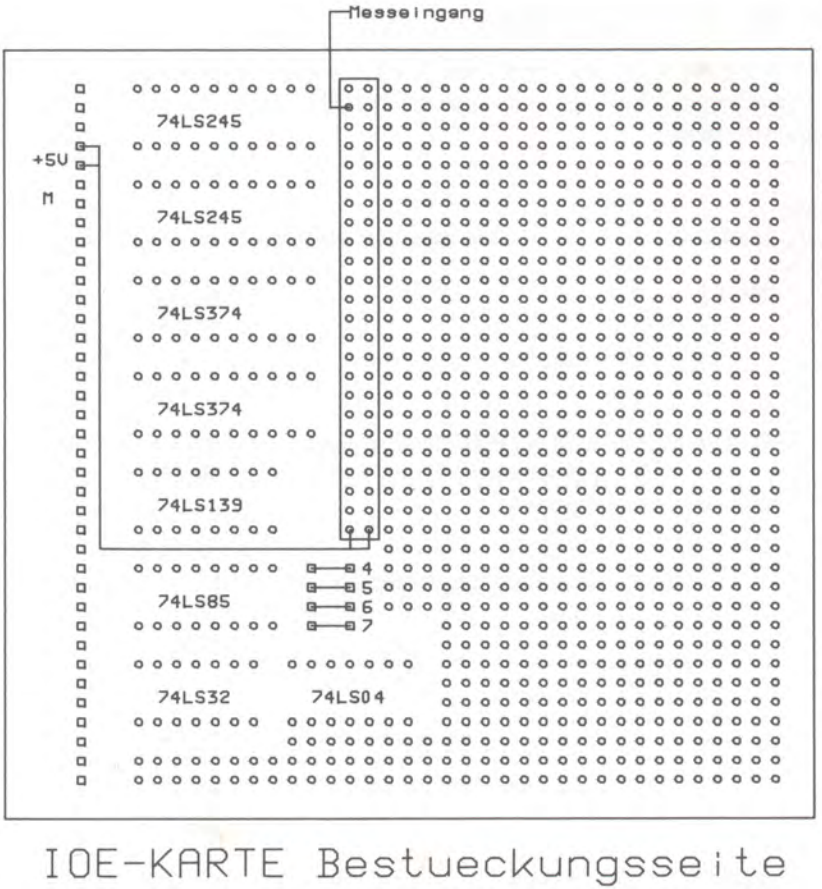


Abb. 1-3: IOE-Karte mit Zusatzleitungen und Brücken

zeigt die IOE-Karte von der Bestückungsseite aus, jedoch wird die Leitung natürlich auf der Lötseite angelötet.

In der Abb. ist ferner noch der Anschluß einer Meßleitung sichtbar, die wir später brauchen.

1.3 HEXIO

Die Baugruppe HEXIO wird über eine Flachbandleitung mit der IOE-Baugruppe verbunden. Und dies ist auch der Grund, warum man die 5V-Leitung noch auf die Stiftleiste löten mußte, denn die HEXIO-Karte benötigt diese Spannung zum Betrieb.

1.4 Verbinden der Baugruppen

Die einzelnen Baugruppen werden durch eine Bus-Leiterplatte miteinander verbunden. Zur Spannungsversorgung benötigt man noch die POW5V-Baugruppe oder eine andere 5V-Spannungsversorgung.

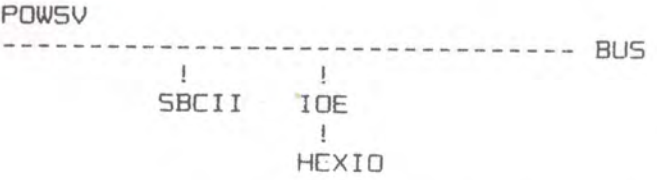


Abb. 1-4:
Konfiguration für HEXMON

Die HEXIO-Baugruppe wird über die Flachbandleitung mit der IOE-Karte verbunden. Die Baugruppen sollen so zu liegen kommen, wie im Bild sichtbar. Die Flachbandleitung wird auf der Bestückungsseite aufgesteckt.

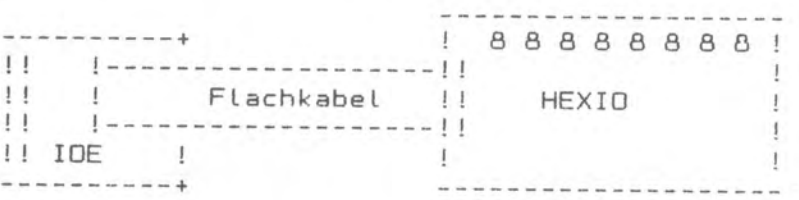


Abb. 1-5: Verbinden von IOE und HEXIO

Auf das Tastenfeld der HEXIO-Karte kann man eine Schablone legen, in die alle Befehle eingetragen sind. Dazu findet sich im Anhang eine Ausschneidetafel.

Dann kann's losgehen. Nach dem Einschalten der Spannung muß, nachdem der RESET (Rücksetzen) ausgelöst ist, auf der Anzeige nach kurzer Zeit die Meldung HALLO-1.1 erscheinen.

2 Einführung in die Bedienung von HEXMON

Nach dem Starten von HEXMON erscheint nach kurzer Zeit folgende Meldung auf der Anzeige.

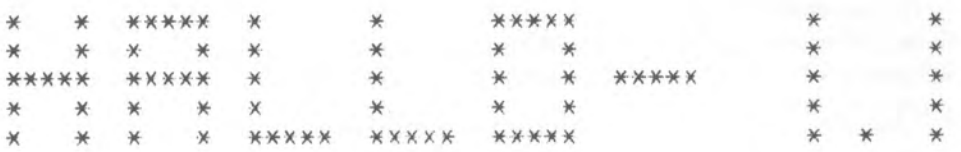


Abb. 2-1: Einschaltmeldung

Damit zeigt HEXMON, daß Befehle eingegeben werden können.

Die Tastatur hat folgende Belegung:

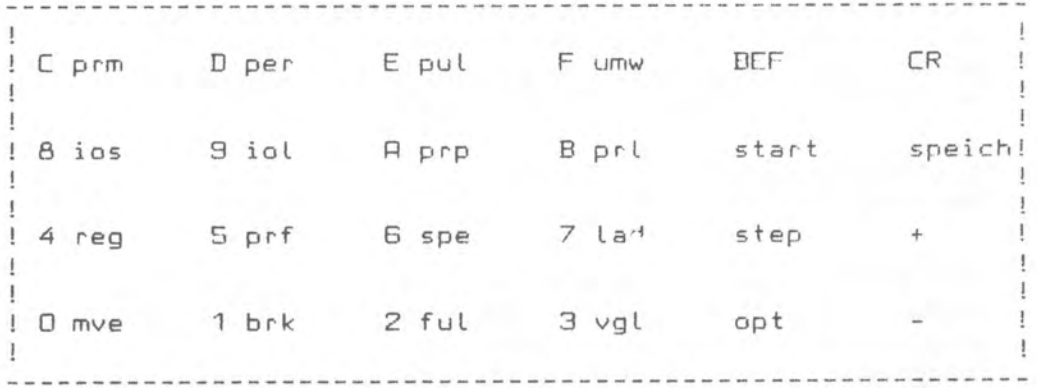


Abb. 2-2: Tastaturbelegung

Über diese Tastatur werden die Befehle an HEXMON gegeben. Dabei sind auf der linken Hälfte der Tastatur neben den Abkürzungen der Befehle auch noch Buchstaben und Zahlen angegeben. Diese Tasten sind, wie man sagt, doppelt belegt, da sie zwei Funktionen erfüllen.

Die Bedeutung der Tasten werden wir nach und nach kennenlernen.

2.1 Zahlensysteme

Warum heißt HEXMON eigentlich HEXMON. Dies ist eine Zusammenfassung von zwei Begriffen, einmal HEXadezimalsystem und zum anderen MONitor. HEX ist das Zahlensystem, und mit MONitor wird ein Programm bezeichnet, das einen Computer in die Lage versetzt, Befehle von einem Benutzer anzunehmen und auszuführen. Denn ohne Programm geht nichts bei Computern. Selbst der größte Supercomputer kann nichts ohne Programme tun. Es ist dann nicht mal möglich, ihm Programme einzugeben, denn auch dazu braucht der Computer schon ein Programm, das Monitorprogramm.

Da das Monitorprogramm eigentlich auch einmal eingegeben werden muß, ist die Sache nur deshalb lösbar, weil wir bei unserem Computer ein EPROM verwenden, in welchem das Programm fest „eingebrannt" ist. Es muß also nicht über eine Tastatur eingegeben werden.

Der Begriff HEX ist eigentlich falsch, wenn man das 16-er Zahlensystem damit meint, denn korrekterweise heißt es sedezimales Zahlensystem und nicht hexadezimales Zahlensystem. Dennoch wird in Anlehnung an den amerikanischen Sprachgebrauch oft von HEXZAHLEN oder dem HEX-Rechner gesprochen.

Was hat es nun mit dem sedezimalen Zahlensystem auf sich. Dazu eine kleine Vorgeschichte.

Computer bevorzugen eigentlich das duale Zahlensystem. In einem Computer, wie wir ihn verwenden, arbeitet man mit zwei Zuständen, Spannung da, Spannung nicht da, oder Strom da, Strom nicht da. Daher kann man die Zahlen 1 und 0 im Rechner einfach als Spannung liegt an, und Spannung ist „aus" definieren.

Spannung liegt an	=	1
Spannung ist "aus"	=	0

Tab. 2-1: Definition von 1 und 0

Wie gelingt es einem nun mit nur zwei Ziffern, nämlich 0 und 1, beliebige Zahlen darzustellen, z.B. die Zahl 23. Zunächst kann man die Ziffern 0 und 1 aneinander reihen, also 101001010110. Wie kann man aus solchen Ziffernketten wieder auf eine dezimale Zahl schließen. Dazu folgende Überlegung. Wie macht man es im dezimalen Zahlensystem, wenn man mehr als eine Ziffer braucht:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

Man nimmt eine neue Stelle hinzu und fängt bei der niedrigeren Stelle wieder bei 0 an zu zählen. Genauso im dualen Zahlensystem. Also 0, 1 dann 10, dann 11, dann 100, dann 101, dann 110, dann 111, dann 1000, dann 1001 usw. Um eine Zuordnung zwischen dem dualen und dem dezimalen Zahlensystem zu erreichen, kann man beide Zahlensysteme einmal nebeneinander schreiben, also:

Dual-Zahl	Dezimal-Zahl	Dual-Zahl	Dezimal-Zahl
0	0	1001	9
1	1	1010	10
10	2	1011	11
11	3	1100	12
100	4	1101	13
101	5	1110	14
110	6	1111	15
111	7	10000	16
1000	8	10001	17

Tab. 2-2: Dual-Zahlen

Die Tabelle kann beliebig fortgesetzt werden. Nun führt diese duale Schreibweise zu sehr langen Zahlenketten, und damit man nicht so lange Ziffernketten schreiben muß, hat man sich eine abkürzende Schreibweise ausgedacht, das sedezimale Zahlensystem (auch oft HEX-Zahlensystem genannt). Dabei zählt man bei 9 einfach mit A, B, C, D, E, F weiter.

Die Umrechnung von Dual auf Dezimal kann man sich so vorstellen, daß man die Dualzahl in Vierergruppen aufteilt und die entsprechende sedezimale Ziffer dort einträgt. Beispiel: 011110010101101 soll in eine sedezimale Zahl gewandelt werden. Also zuerst die Aufteilung in Vierergruppen: 011 1100 1010 1101.

Dual-Zahl	Dezimal-Zahl	Sedezimal-Zahl
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10
10001	17	11

Tab. 2-3: Sedezimal-Zahlen

Dann wird jede dieser Vierergruppen getrennt umgerechnet und die sedezimale Ziffer unter den Viererblock geschrieben:

```
011 1100 1010 1101
 3   C    A    D
```

Die Umrechnung ins dezimale Zahlensystem ist nicht so einfach, doch dazu haben wir ein Programm im HEXMON, mit dem wir das tun wollen.

2.2 Versuch: Umwandlung von Zahlen

Wir drücken, nachdem sich HEXMON mit HALLO 1.1 gemeldet hat, die Taste mit der Beschriftung F umw. Es erscheint folgende Kombination auf der Anzeige: 0.0.0.0.

Nun können wir die sedezimale Zahl eingeben, die wir umrechnen wollen. Dazu dient die linke Hälfte der Tastatur, die auch die Beschriftungen 0, 1, 2, 3, 4, 5, 6, 7, 8, A, B, C, D, E, F neben den Befehlsabkürzungen trägt. Wir wollen einmal unsere sedezimale Zahl eingeben. Dazu drücken wir zuerst die Taste mit der Beschriftung 3 vgl, es erscheint eine 3 rechts auf der Anzeige: 0.0.0.3.

Die Punkte geben übrigens immer an, daß der Rechner auf eine Zahleneingabe wartet. Nun geben wir die nächste Stelle ein und drücken die Taste C prm. Die 3 ist nun eine Stelle nach links gerückt, und die Anzeige sieht so aus: 0.0.3.C. Dann geben wir die restlichen zwei Stellen ein, also die Taste mit A und die Taste mit D.

Auf der Anzeige erscheint dann: 3.C.A.D. Nun müssen wir dem Rechner sagen, daß das die Zahl ist, die er umrechnen soll. Denn wir könnten auch weitere Ziffern eintasten, z.B. nach Eingabe von 2 wäre auf der Anzeige C.A.D.2. zu sehen und die Ziffer 3 ginge verloren. Mit diesem Verfahren kann man auch Eingabefehler korrigieren, indem einfach die richtige Zahl nochmals eingegeben wird. Wenn weniger als vier Stellen eingegeben werden sollen, so muß man auch die Nullen mit eingeben, also will man nicht 3.C.A.D. umrechnen sondern eine Zahl 123, dann muß man 0 und 1, und 2 und 3 eingeben.

Doch wir wollen ja 3.C A.D. umrechnen, und dazu muß man jetzt die mit CR beschriftete Taste drücken. Auf der Anzeige erscheint dann der Wert: d 15533.

Das „d" an der ersten Stelle steht für dezimal, denn eine dezimale Zahl kann man nicht von einer gleichen sedezimalen unterscheiden, z.B. 1234 kann sedezimal oder dezimal gemeint sein. In der Computertechnik unterscheidet man meistens die sedezimalen Zahlen durch ein hinten angestelltes „H", also 1234H ist sedezimal und 1234 ist dezimal.

Die Bezeichnung CR ist eine Abkürzung aus dem Englischen für carriage return, was zu deutsch etwa Wagenrücklauf bedeutet und aus der Schreibmaschinentechnik entlehnt ist. Hier stimmt die Bedeutung nicht mehr ganz, denn wir haben ja keinen „Wagen", der irgendwohin zurückläuft. Doch um einem Computer zu sagen, daß irgend etwas angeschlossen ist, hat sich diese Taste eingebürgert.

Die Bezeichnung „umw"auf der Taste „F umw" war eine Abkürzung für Umwandlung, also die Umwandlung zwischen Zahlensystemen. Mit dieser Funktion können wir

auch eine dezimale Zahl umwandeln, also den umgekehrten Vorgang durchführen. Dazu drücken wir jetzt die Taste „opt". opt ist die Abkürzung für Optionen, was soviel wie Zusatz bedeutet.

Dann steht auf der Anzeige der Wert 3CAd, also unser ursprünglicher Zahlenwert.

Übrigens werden auf der Anzeige bei der Ausgabe oft Groß- mit Kleinbuchstaben gemischt, da man mit den verwendeten Anzeigen nicht alle Buchstaben des Alphabets gut darstellen kann.

Nun drücken wir nochmals CR, und es erscheint 0. Der Rechner will also wieder eine Eingabe. Diesmal erwartet er aber eine dezimale Zahl. Wir geben zum Beispiel mal die Ziffern 3 2 7 6 7 ein. Auf der Anzeige erscheint 3.2.7.6.7. Hat man sich bei der Eingabe vertippt, gibt man am besten so viele Nullen ein, bis die Anzeige wieder leer ist. Und dann die Zahl nochmals von vorne. Nun tippen wir die Taste CR an und auf der Anzeige erscheint: S 7FFF.

S steht für Sedezimal, um zu kennzeichnen, in welchem Zahlensystem die Ausgabe erfolgt.

2.3 Versuch: Negative Zahlen

Wir drücken nun die Taste BEF, und danach erscheint in der Anzeige die Schrift: -bEF-

Damit zeigt HEXMON uns, daß ein Befehl als Eingabe erwartet wird; genauso wenn nach dem Einschalten der Versorgungsspannung HALLO-1.1 erscheint. Wenn man eine falsche Taste drückt, z.B. nochmals BEF, so erscheint ------- auf der Anzeige, um einen Eingabefehler anzugeben. Danach kann man aber auch wieder ganz normal eine Befehlstaste drücken.

Wir betätigen die Taste umw (die mit „F umw" beschriftet ist). Nun geben wir CR ein, denn wir wollen zuerst eine dezimale Zahl in eine sedezimale Zahl umwandeln. Es erscheint d 0 auf der Anzeige. Danach betätigen wir die Taste opt. Es erscheint S 0000 auf der Anzeige. Und dann geben wir CR ein. Damit erscheint 0., was wir haben wollten, denn nun können wir eine dezimale Zahl eingeben. Wir tippen mal die Zahl 12 ein. Auf der Anzeige erscheint 1.2.. Nun betätigen wir die Taste -. Auf der Anzeige erscheint -1.2. Nach der Eingabe des Minus-Zeichens darf man keine weitere Zifferntaste mehr eingeben. Nun drücken wir die Taste CR, und auf der Anzeige erscheint: S FFF4.

Was ist das? Kein Minus-Zeichen ist zu sehen. Die Zahl FFF4 ist die sogenannte Zweierkomplement-Darstellung. Das ist eine Zahlendarstellung, die bei Computern häufig zur Darstellung negativer Zahlen verwendet wird. Wie kommt man aber darauf. Dazu sehen wir uns die Zahl 12 einmal im Dual-Code an. Dort lautet sie 1100, wie man aus der Dual-Tabelle entnehmen kann.

Das Zweierkomplement erhält man, wenn man die Zahl zunächst komplementiert, das heißt jede Stelle invertiert, also aus einer 1 wird eine 0 und umgekehrt: Man erhält 0011. Dabei ist nun aber zu berücksichtigen, wie groß der Zahlenbereich ist, also die maximale Zahl, die dargestellt werden soll. Wir wollen die Zahl mit 16 Bit darstellen,

also 16 Dual-Stellen verwenden, dann lautet die Zahl 1111 1111 1111 0011. Nun muß man zur Bildung des Zweierkomplements noch die Zahl 1 addieren. Man erhält:

1111 1111 1111 0100.

Das ist das Zweierkomplement der Zahl. Wenn wir diese Zahl noch sedezimal darstellen, ergibt sich FFF4.

Addiert man die 1 nicht, so passiert etwas ganz Merkwürdiges. Wir wollen einmal die Zahl 0 nehmen. Diese lautet dual ebenfalls 0. Bilden wir das Komplement, z.B. für 16 Stellen, so erhalten wir 1111111111111111.

Das nennt man auch das Einerkomplement einer Zahl. Beim Einerkomplement gibt es zwei Darstellungen für die Zahl 0. Nämlich 0 und -0.

Das Zweierkomplement vermeidet diese Schwierigkeit, denn wenn man nun 1 auf den Wert addiert, so erhält man wieder die Zahl 0, sofern man den entstehenden Übertrag in die 17te Stelle vergißt. Und genau das tut man bei der Bildung des Zweierkomplements.

2.4 Versuch: „Der Speicher"

Der Speicher ist mit das wichtigste „Organ" in einem Computer. Er hat die Aufgabe, Daten und Programme festzuhalten.

Man unterscheidet dabei zwei grundsätzliche Arten von Speicherbausteinen.

Das RAM

Der Ausdruck entstammt dem Englischen und bedeutet Random Access Memory. Das fürchterliche Wort bedeutet lediglich soviel, daß man in diesem Speicher Daten ablegen kann und sie anschließend auch wiederfindet.

Das ROM

Auch dieser Ausdruck stammt vom Englischen. ROM ist die Abkürzung für Read Only Memory. Frei übersetzt bedeutet das soviel wie ein Speicher, aus dem man nur lesen kann, in den man aber keine Daten schreiben kann.

Wozu der Quatsch? Ein Speicher aus dem man nur Lesen kann! Wie kommen dann die Daten dort hinein?

Natürlich kann man diesen Speicher auch mit Daten belegen, nur geht das nicht so einfach. Und wenn einmal Daten eingespeichert sind, so kann man sie nicht mehr ändern. Bei ROMs geschieht das Einspeichern schon bei der Herstellung der Bausteine durch Aufdampfen von Metallschichten.

Es gibt noch verschiedene Abarten. Bei den PROMs (Programmable Read Only Memory) kann man auch nach der Herstellung der Bausteine die Daten einspeichern, indem man im IC einzelne Metallbrücken durch Anlegen von Spannung verdampfen läßt.

Dieser Vorgang ist ebenfalls einmalig, man kann keine neue Information in dem IC unterbringen.

Beim EPROM (Erasable Programmable Read Only Memory) wird die Information in Form von elektrischer Ladung auf dem IC gespeichert. Wenn man das IC durch ein Quarzglas mit UV-Licht bestrahlt, kann man den Dateninhalt wieder löschen und das IC

anschließend neu programmieren. Dies geht aber nicht beliebig oft, das IC ist nach einiger Zeit nicht mehr verwendbar. ROM ist eigentlich der Übergang zum RAM. Nur ein wesentlicher Unterschied besteht. Das Einschreiben der Daten dauert ungleich länger als wenn man Daten auslesen will. So dauert ein Programmiervorgang ca. 5min und das Löschen ca. 15min, während man ein Datum in ca. 200ns (Nanosekunden) auslesen kann.

Wir wollen uns mal den Speicher unseres Computers ansehen. Wir schalten den Computer an, und er meldet sich mit Hallo 1.1 (oder -bEF- oder --------).

1.Dazu wird die Taste speich gedrückt. In der Anzeige erscheint Adr 8.1.0.0.

Immer wenn Punkte hinter den Ziffern erscheinen, kann man auch andere Zahlen eingeben, indem man die Tasten 0 bis F auf der Tastatur drückt. Will man aber den angezeigten Wert beibehalten, so gibt man einfach CR ein.

2.Wir wollen eine neue Adresse angeben. Und zwar die Adresse 0. Dazu tippen wir solange die Taste 0, bis in der Anzeige lauter Nullen erscheinen.

2. Dann geben wir CR ein. In der Anzeige erscheint nun folgendes Bild: 0000 C.3. Der Wert C3 ist der Inhalt der Speicherzelle mit der Adresse 0.

3. Wir drücken erneut die Taste CR. Es erscheint: 0001 FF . Der Wert FF ist also der Inhalt der Speicherzelle 1.

4. Wir tippen +. Es erscheint 0002 0d auf der Anzeige. Man kann + oder CR verwenden, um sich den Inhalt einer weiteren Speicherzelle anzusehen.

5. Wir drücken aber jetzt einmal die Taste opt. Dann erscheint auf der Anzeige der Wert 0d.

6. Nun geben wir CR ein und es erscheint C38702. Dies sind drei Speicherzellen hintereinander dargestellt. Beim Z80 gibt es Codekombinationen, die mehrere Speicherzellen in logisch zusammenhängende Einheiten gruppieren. Beim Programmieren wird man noch mehr darüber hören.

7. Wir wollen aber noch nicht programmieren und drücken daher wieder die Taste opt, um wieder in die normale Ausgabeform zurückzugelangen. (0003 C3).

8. Jetzt tippen wir mal die Taste -. Es erscheint 0002 0d auf der Anzeige. Mit der Taste - gelangt man einen Speicherplatz zurück.

9. Nun wollen wir uns mal einen anderen Speicherbereich ansehen, denn bei Adresse 0 liegt nur EPROM-Speicher. Dazu drücken wir die Taste BEF. In der Anzeige erscheint die Meldung -bEF-.

10. Dann drücken wir wieder die Taste speich. In der Anzeige erscheint wieder die Meldung adr 8.1.0.0. .Erscheint hier der Wert 0.0.0.0. , so müssen wir die Adresse 8100 eintippen (8 und 1 und 0 und 0). An dieser Stelle wollen wir diesmal etwas schreiben, denn dort befindet sich RAM-Speicher. RAM-Speicher steht auf der SBCII-Karte von Adresse 8000H bis 8FFFH zur Verfügung. Die unteren 100H Bytes sollte man allerdings nicht verwenden, da HEXMON selbst diesen Speicherbereich benötigt. Also der Speicherbereich von 8000H bis 80FFH ist verboten.

11. Wir drücken die Taste CR. Dann erscheint der Inhalt der Speicherzelle 8100 auf der rechten Seite des Anzeigefeldes. Welcher Wert dort steht kann man nicht sagen, denn nach dem Spannungseinschalten nehmen diese Speicherzellen einen beliebigen Wert ein.

12. Wir tippen die Tasten 5 und A ein. Auf der Anzeige steht nun 8100 5.A. . Die Zahl wird erst dann abgespeichert, wenn wir die Taste CR drücken. Dann erscheint auch gleich die nächste Adresse und deren Speicherinhalt auf der Anzeige. Dort geben wir mal die Zahl 00 ein, also die Tasten 0 und nochmals 0 drücken. Auf der Anzeige erscheint nun 8101 0.0.

13. Dann wieder CR eingaben und der Inhalt der nächsten Speicherzelle erscheint.

14. Wir drücken zweimal die Taste „-". Auf der Anzeige muß der Inhalt der Speicherzelle 8100 erscheinen. Es steht dort der Wert 5A.

15. Nun „+" eintippen. Es erscheint der Inhalt der Speicherzelle 8101, also 00.

16. Die Taste BEF wird gedrückt und HEXMON zeigt -bEF- an. Wir haben beim letzten Teil die Werte 5A und 00 in zwei Speicherzellen abgelegt.

```
8100  5A
8101  00
```

Auf diese Weise kann man nicht nur Daten in dem RAM-Speicher ablegen, sondern auch Z80-Programme.

Der Befehl speich ist sehr wichtig. Wer hier noch keine Übung hat, soll folgende Daten einmal in den RAM-Speicher eingeben und anschließend kontrollieren, ob die Daten auch dort angekommen sind.

Übung 1: 8100: 00 55 AA 12 45 5A A5 C3 00 81

Übung 2: 8200: 01 02 04 08 10 20 40 80

Gibt man übrigens Daten in einem Speicher ein, in dem kein RAM-Speicher ist, so wird nach Drücken der CR-Taste der alte Inhalt wieder angezeigt und auch die alte Speicherzelle.

Beispiel:
Wir geben die Adresse 0 ein (speich, 0000, CR). Der Inhalt der Zeile 0 ist C3. Nun geben wir den Wert 45 ein. In der Anzeige erscheint dann 0000 4.5. . Drücken wir dann die Taste CR, so erscheint auf der Anzeige wieder 0000 C.3. .

2.5 Versuch: Verknüpfungen

Damit ein Computer rechnen kann, aber auch für logische Entscheidungen, braucht er bestimmte Elemente. Diese Elemente werden Verknüpfungen genannt. Eine Verknüpfung kann aus Eingangssignalen ein Ausgangssignal erzeugen. Dazu gibt es bestimmte Regeln. Die wichtigsten Verknüpfungsarten wollen wir kurz besprechen und dann auf dem NDR-Klein-Computer nachbilden und damit experimentieren.

Das Nicht-Glied

Dieser Baustein macht aus einem ankommenden 1-Signal ein 0-Signal und aus einem ankommenden 0-Signal ein 1-Signal.

Das Und-Glied

Das Und-Glied liefert an seinem Ausgang genau dann ein 1-Signal, wenn alle Eingänge des Und-Glieds zur selben Zeit ein 1-Signal führen.

Das Oder-Glied

Es liefert immer dann ein 1-Signal an seinem Ausgang, wenn mindestens einer seiner Eingänge ein 1-Signal besitzt, oder aber mehrere Eingänge ein 1-Signal anliegen haben.

Nun zum Experiment

Wir wollen die Verknüpfungen mit einem kleinen Z80-Programm realisieren. Zwei der Tasten auf dem Eingabefeld sollen die Eingangssignale darstellen. Als Ausgang wollen wir Leuchtdioden in der Anzeige verwenden, und zwar das Segment a.

Dazu ein Programm. Zunächst wollen wir das Programm einmal so hinnehmen, wie es ist, und die einzelnen Befehle nicht weiter behandeln.

```
! 8100: 3E FE D3 00 DB 00 2F 47 3E FD D3 00 DB 00 2F A0  !
! 8110: 2F F6 FE D3 01 18 E9                             !
```

Abb. 2-3: Programm „Verknüpfung"

Das Programm wird auf Adresse 8100 eingetippt. Der letzte Wert E9 steht auf Adresse 8116.

Nun müssen wir das Programm starten.

Dazu:

1. Zurück in den Befehlsmode (Anzeige -bEF-)

2. Drücken der Taste start. Auf der Anzeige erscheint die Meldung Adr x.x.x.x. . (x.x.x.x. bedeutet, daß die Zahl unterschiedlich sein kann).

3. Eingeben der Startadresse, also die Tasten 8 dann 1 dann 0 und dann 0 drücken.

4. Starten. Drücken der Taste CR. Die Anzeige erlischt.

Wenn man die Taste 0 drückt, passiert nichts.

Wenn man die Taste 1 drückt, passiert nichts.

Wenn man die Taste 0 und gleichzeitig die Taste 1 drückt, dann leuchten zwei Segmente (a) der Anzeige auf.

Wir haben eine Und-Verknüpfung realisiert.

Will man eine andere Verknüpfung realisieren, so geht das auch.

Beispiel ODER-Glied.

Dazu wird zunächst der RESET-Taster an der SBCII-Karte gedrückt, und HEXMON meldet sich wieder mit HALLO 1.1.

Mit speich wird die Adresse 810F eingegeben. Die Speicherzelle hat den Inhalt A0. Nun wird anstelle dieses Wertes der neue Befehl B0 geschrieben.

Es wird dann nach CR die Taste BEF gedrückt und das Programm mit start auf Adresse 8100 gestartet.

Nun leuchten die beiden Segmente, wenn entweder die Taste 0 gedrückt wird, oder die Taste 1 oder beide Tasten.

Wenn man will, kann man nun noch die Nicht-Verknüpfung ausprobieren. Dies ist der Befehl 2F, den man auf die Adresse 810F schreibt. Dann leuchten die Segmente, wenn man die Taste 1 nicht drückt. (Taste 0 ist wirkungslos, da die Nicht-Verknüpfung nur einen Eingang besitzt).

Mit dem Befehl A8 kann man selbst einmal experimentieren. Er wird auch auf die Adresse 810F geschrieben.

2.6 Versuch: Flip-Flop

Im Computer braucht man Speicher. Wir wollen hier mal eine einzelne Speicherzelle nachbilden.

Dazu tippen wir folgendes Programm auf Adresse 8100H ein:

```
! 8100: 3E 7F D3 01 3E FE D3 00 DB 00 E6 01 20 F6 3E FD !
! 8110: D3 00 DB 00 E6 01 20 F6 18 EA                   !
```

Abb. 2-4: Programm „Flip-Flop"

Das Programm wird auf Adresse 8100H gestartet. Das H hinter der Zahl steht für Hex, um das Zahlensystem zu kennzeichnen. Es leuchtet ein Punkt links auf der Anzeige.

Wenn man die Taste 0 drückt, so leuchtet der rechts daneben liegende Punkt.

Drückt man dann die Taste 1, so leuchtet wieder die linke Led.

Das Flip-Flop merkt sich also, welche Taste gedrückt wurde. Wenn man eine Taste mehrmals betätigt, so bleibt das Flip-Flop im neuen Zustand und ändert sich nicht mehr.

Drückt man beide Tasten (0 und 1) gleichzeitig, so leuchten beide LEDs auf. Dies ist beim einfachen Flip-Flop ein unzulässiger Zustand.

In der Praxis gibt es eine Vielzahl unterschiedlicher Flip-Flop-Typen mit verschiedenen Eigenschaften. Sie alle können verwendet werden, um Speicherzellen zu bilden.

In den RAM Bausteinen unseres Computers befinden sich ähnliche Flip-Flops zum Speichern der Daten. Es sind dies sogenannte statische Speicherzellen. Es gibt auch

dynamische Speicherzellen. Dort wird die Speicherung nicht durch Flip-Flops durchgeführt, sondern durch Kondensatoren, die eine Ladung tragen können oder keine, je nach Zustand der Speicherzelle. Dabei haben Kondensatoren den Nachteil, die Ladung nach und nach zu verlieren. Deshalb muß man sie von Zeit zu Zeit nachfüllen, und das geschieht beim sogenannten Refresh. Alle 2ms (oder bei anderen Bausteinen 4ms) werden alle Speicherzellen kurz angesprochen (sprich adressiert), und durch eine im RAM vorhandene Spezialschaltung wird die Ladung erneuert, falls vorher eine Ladung (wenn auch geringer) vorhanden war.

Dynamische Speicher sind im allgemeinen preiswerter und mit einer hohen Integrationsdichte (also viele Speicherzellen pro Quadratzentimeter) verfügbar. Sie sind aber auch komplzierter zu betreiben, und die Schaltungen um sie herum sind aufwendiger.

2.7 Versuch: Arithmetik

Es sollen jetzt mal ein paar Z80-Befehle behandelt werden. Wenn man anfängt die Befehle eines Mikroprozessors zu verwenden, so braucht man zunächst nicht gleich alle auf einmal zu lernen. Es genügen erst mal ein paar wenige Grundbefehle.

Es soll eine Addition durchgeführt werden. Dazu verwenden wir den Befehl ADD A, n, mit dem man eine 8-Bit-Zahl addieren kann. Der Wert wird dabei im Akkumulator des Z80 abgelegt. Der Akkumulator ist das A-Register. Man braucht dann noch einen Lade-Befehl, also LD A,n, was bedeutet: Lade den Wert n in den Akkumulator (A). Die Zahlen 5 und 4 sollen addiert werden. Also müssen wir zuerst die Zahl 5 in den Akkumulator laden und dann die Zahl 4 zum Inhalt addieren. Das Programm sieht dann so aus:

```
LD  A,5
ADD A,4
```

Nun muß das Programm noch in Maschinencode übersetzt werden. Das Ergebnis sieht dann so aus:

```
3E 05          LD  A,5
C6 04          ADD A,4
```

Man kann den Maschinencode auch untereinander schreiben und Adressen hinzufügen: Als Startadresse nehmen wir mal 8100H.

```
8100 3E LD  A,5
8101 05
8102 C6 ADD A,4
8103 04
```

Dieses Programm könnten wir nun eingeben und dann starten. Doch Halt! Was passiert mit dem Ergebnis? Der Wert steht anschließend im Akkumulator. Doch wie kann man den erkennen? Außerdem, was passiert, wenn das Programm am Ende ankommt und der Befehl auf Adresse 8104 ausführt?

Zum Testen verwenden wir die Einzelschritt-Taste STEP auf der Tastatur, mit der wir ein Programm schrittweise ausführen können.

1. Also zunächst mal das Programm eintippen:

   8100 3E 05 C6 04

2. Nun wieder in den Befehlsmode zurück (-bEF-), falls noch nicht geschehen.

3. Die Taste step drücken, es erscheint Adr x.x.x.x. auf der Anzeige.

4. Die Tasten 8 1 0 0 drücken und anschließend die Taste CR. Auf der Anzeige erscheint 8100 3E.

5. Nun die Taste 4 reg drücken. Mit dieser Taste erreicht man, daß man Registerinhalte während des Einzelschritts ansehen kann.

6. Wenn auf der linken Anzeigenseite nicht der Text AF erscheint, kann man mit den Tasten + oder - das Registerpaar anwählen.

7. Die Taste opt drücken und die Punkte beim Zahlenwert verschwinden.

8. Nun die Taste step drücken. In der Anzeige wird der Wert 05xx ausgegeben (xx kann unterschiedlich sein). Der Wert 5 wurde also geladen.

9. Nochmals step drücken. Es erscheint der Wert 9 in der Anzeige, also das Ergebnis der Summe 5 + 4.

10. BEF drücken, und man gelangt zurück in den Befehlsmode.

Anstelle des ADD A, n Befehls kann man auch eine Subtraktion setzen, also SUB A,n

Der Code lautet dann D6 xx.

2.8 Versuch: Displacement-Rechner

Wenn man beim Z80 relative Sprünge verwendet, so muß man das sogenannte Displacement berechnen. Es wird verwendet, um den Abstand zwischen dem Sprung und dem Sprungziel anzugeben. Relative Sprünge haben den Vorteil, daß der Programmcode verschiebbar ist, also nicht von der absoluten Lage des Programms abhängt.

Mit den relativen Sprüngen kann man im Bereich von -126 bis +129 springen.

Wenn man ein Z80-Programm von Hand codiert, so hat man zunächst nur absolute Adressen vorliegen. Daraus kann man die relative Adresse berechnen.

Beispiel:

```
        8100 00
START:  8101 18      JR ZIEL
        8102 dd
        8103 00
ZIEL:   8104 00
```

Der Wert dd ist zu berechnen. Dazu gibt es folgende Formel:

dd = ZIEL – START – 2

Also lautet der Befehl 18 01
Denn es gilt dd = 8104-8101-2 = 1

Bei Sprüngen zurück geht das genauso:

ZIEL: 8100 00
8101 00
START:8102 18
8103 dd

Für dd ergibt sich:

dd = 8100-8102-2
dd = FFFC

Man nimmt die letzten beiden Ziffern

also dd = FC

Der Sprung lautet dann: 18 FC

Für unser Programm verwenden wir Unterprogramme aus dem HEXMON.

```
8100 CD    CALL CLEAR       ; Anzeige löschen
8101 33
8102 00
8103 DD    LD IX,8000H      ; Adresse Anzeigefeld
8104 21
8105 00
8106 80
8107 21    LD HL,0          ; Wert in der Anzeige
8108 00                     ; vor dem Aufruf
8109 00
810A CD    CALL GETHL       ; und Wert nach HL holen
810B 27                     ; ist START-Wert
810C 00
810D EB    EX DE,HL         ; in DE merken
810E DD    LD IX,8000H      ; Adresse Anzeigefeld
810F 21
8110 00
8111 80
8112 21    LD HL,0          ; Wert in der Anzeige
8113 00                     ; vor dem Aufruf
8114 00
8115 CD    CALL GETHL       ; und Wert nach HL holen
8116 27                     ; ist ZIEL-Wert
8117 00
8118 AF    XOR A            ; Uebertrag löschen
8119 ED    SBC HL,DE        ; ZIEL-START bilden
811A 52
811B 2B    DEC DE           ; -1
811C 2B    DEC DE           ; -1
```

```
811D CD    CALL CLEAR       ; Anzeige löschen
811E 33
811F 00
8120 DD    LD IX,8000H      ; Start Anzeigefeld
8121 21
8122 00
8123 80
8124 7D    LD A,L           ; Nur 8 Bits ausgeben
8125 CD    CALL PRTAC       ; Ausgabeprogramm
8126 18
8127 00
8128 CD    CALL HOLETASTE   ; und warten bis
8129 0C                     ; Taste gedrückt
812A 00
812B 18                     ; Dann alles wiederholen
812C d3
```

Abb. 2-5: Programm „Displacement-Rechner"

Das Programm wird auf Adresse 8100H gestartet. Testen wir es mal.

1. Zuerst wird die Startadresse des Befehl eingegeben. Auf Adresse 812B unseres Programms beginnt ein solcher Sprung-Befehl. Wir geben mal die Adresse 812B als Start-Adresse ein.

2. Nun erscheint wieder ein Feld mit Nullen. Jetzt wird das Sprungziel eingegeben, wir geben mal die Adresse 8100 ein.

3. Auf der Anzeige erscheint nun der Wert d3, also das Ergebnis. Dies ist auch der Wert in unserem Programm.

4. Drückt man nun wieder die Taste CR, so erscheint wieder das Feld mit den Nullen und man kann eine neue Startadresse eingeben.

Das Programm läßt sich nur durch einen RESET mit dem Taster der SBCII-Karte durchführen.

Um den Umfang des Buches nicht zu sprengen, soll an dieser Stelle die Einführung in die Z80-Programmierung beendet sein. Im Literaturverzeichnis des Anhangs finden sich weiterführende Bücher, die man zur Fortbildung verwenden kann, und mit Hilfe von HEXMON kann man dann weitere Z80-Befehle kennenlernen.

3 HEXMON-Befehle

Die Befehle im HEXMON sind sehr komfortabel gestaltet. Hier eine genaue Beschreibung aller vorhandenen Befehle. Die Befehle sind dabei in der Reihenfolge aufgeführt, wie sie auf der Tastatur angeordnet sind.

3.1 O mve – Speicherbereiche verschieben

MVE ist die Abkürzung von MOVE, was soviel wie transportieren, verschieben, bewegen bedeutet. Damit lassen sich Speicherinhalte von einem Bereich in einen anderen kopieren. Das Programm verlangt dazu zunächst eine VON-Adresse, also die Startadresse von wo aus angefangen werden soll, Daten zu transportieren. Dann wird die BIS-Adresse abgefragt, also die Adresse, bis wohin der Bereich gehen soll. Und dann die NACH-Adresse, also die Adresse, wohin der angegebene Bereich kopiert werden soll.

Beispiel:
Die Daten von Adresse 0 bis FF sollen nach 8100 transportiert werden. Dazu gibt man an:

```
VON 0.0.0.0.
BIS 0.0.F.F.
NAC 8.1.0.0.
```

Nach Eingabe der letzten Zahl erscheint dann normalerweise wieder -bEF-. Der Transport ist dann ausgeführt. Nun ist beginnend bei Adresse 8100 bis 81FF derselbe Dateninhalt zu finden, wie auf Adresse 0000 bis 00FF.

Den MVE-Befehl kann man auch verwenden, um ein Byte in ein Programm einzufügen oder zu löschen.

Beispiel für das Einfügen eines Bytes:

```
VON 8.1.0.0.
BIS 8.1.F.F.
NAC 8.1.0.1.
```

Auf der Adresse 8100 ist nun Platz für ein weiteres Byte. Wenn man damit Programme verschiebt, muß man darauf achten, daß alle absoluten Adressenangaben innerhalb eines Befehls, natürlich durch den MVE-Befehl, unverändert bleiben. Somit laufen die Programme dann nicht mehr. Man muß nach dem Transport alle absoluten Adressen im Programm entsprechend umändern.

Beispiel für Löschen eines Bytes:

```
VON 8.1.0.1
BIS 8.2.0.0
NAC 8.1.0.0
```

Der Inhalt der Speicherzelle 8100 ist überschrieben worden und alle Speicherzellen dahinter (von 8101 bis 8200) sind zurückgeschoben worden.

Arbeitsweise des Befehls

Das Unterprogramm DOMVE in HEXMON übernimmt die Abarbeitung dieses Befehls.

Mit den Z80-Befehlen LDIR und LDDR kann man Speicherblöcke verschieben. Diese Befehle arbeiten sehr schnell und sind auch für viele andere Anwendungsfälle interessant. Der Unterschied zwischen LDIR und LDDR besteht in der Art, wie Speicherblöcke verschoben werden.

Bei LDIR steht in HL die Startadresse, in DE die Zieladresse und in BC die Anzahl der Bytes, die transportiert werden soll. Der Befehl könnte mit anderen Z80-Befehlen so dargestellt werden:

```
SCHLEIFE:
        LD A, (HL)
        LD (DE), A
        INC HL
        INC DE
        DEC BC
        LD A, C
        OR B
        JP NZ, SCHLEIFE
```

Die Register HL und DE werden immer nach dem Transport eines Bytes um eins erhöht.

Bei LDDR sieht das anders aus:

```
SCHLEIFE:
        LD A, (HL)
        LD (DE), A
        DEC HL
        DEC DE
        DEC BC
        LD A, C
        OR B
        JP NZ, SCHLEIFE
```

Dort werden die Register HL und DE nach jedem Transport um eins verringert.

Der Befehl LDIR verschiebt also Blöcke beginnend bei der tieferen Adresse und dann aufsteigend, und der LDDR-Befehl arbeitet umgekehrt.

Will man ein Byte in einen Speicherbereich einfügen oder aus einem Speicherbereich löschen, so überlappen sich Ziel- und Quellbereiche teilweise. Dies kann bei falscher

Anwendung des LDIR oder LDDR-Befehls zur Zerstörung des ursprünglichen Inhalts führen. In DOMVE wird unterschieden, wie ein Bereich transportiert wird.

Ist die Zieladresse größer als die Quelladresse, so wird der Befehl LDDR verwendet, wenn die Zieladresse kleiner als die Quelladresse ist, wird der Befehl LDIR verwendet. Dadurch wird eine Zerstörung von Daten vermieden. Bei LDDR muß allerdings am Ende des Speicherbereichs begonnen werden zu transportieren, und bei LDIR am Anfang.

3.2 1 brk - Haltepunkte setzen

Es können bis zu drei Haltepunkte gesetzt werden. Dazu wird eine Nummer auf dem linken Teil der Anzeige ausgegeben und rechts erscheint die Haltepunkt- Adresse (Breakpoint-Address). Man kann dann eine Zahl eingeben und mit CR den nächsten Haltepunkt eingeben. Ist die Adresse = 0, dann ist der Haltepunkt nicht gesetzt. Mit der Taste BEF kommt man auch hier wieder in den Befehlsmode zurück und jegliche Eingabe wird ignoriert.

Die Haltepunkte kann man sich jederzeit mit diesem Befehl auch einfach ansehen; dazu gibt man einfach keine Zahl ein, sondern nur CR. Dann wird der nächste Haltepunkt angezeigt.

Arbeitsweise des Befehls

Die Haltepunkt-Adressen werden in Speicherzellen des RAM-Bereichs mit den Bezeichnungen BRK1, BRK2 und BRK3 gemerkt. Jede Adresse belegt dabei zwei Bytes, da 16Bit-Adressierung verwendet wird. Danach ist noch ein extra Byte im Speicher reserviert. Dieses Byte wird für den START-Befehl benötigt. Immer wenn ein Start eines Programms durchgeführt wird, wird an alle Speicherzellen mit den Haltepunkt-Adressen ein RST6-Befehl (Code F7) geschrieben. Dort steht aber ein Anwenderbefehl. Dieser wird zuvor hinter BRK1, BRK2 oder BRK3 abgespeichert und somit gemerkt. Wenn dann der Haltepunkt auftritt, oder der Z80 die Adresse 30H anspringt, so wird der alte Befehlsinhalt wieder restauriert und das Anwenderprogramm bleibt dadurch nach der Ausführung unverändert.

3.3 2 ful - Speicherbereiche füllen

Ein Speicherbereich kann mit einem konstanten Wert aufgefüllt werden.

Dazu wird die VON-Adresse abgefragt, nämlich die Startadresse des Speicherbereichs, und als nächstes wird die BIS-Adresse abgefragt, also das Ende des Speicherbereichs. Danach wird nach einem Datenwert gefragt mit der Meldung DTA. Dieser Datenwert wird dann in den Speicherbereich geladen: Es wird bei jeder Speicherzelle geprüft, ob der Wert auch ankommt; wenn dies nicht der Fall ist, erscheint die Meldung ERR mit der Adresse, bei der der Fehler auftrat. Dieser Fehler tritt im ROM-Bereich oder in Bereichen auf, in denen kein Speicher vorhanden ist. Tritt er in einem RAM-Bereich auf, so kann es sich ggf. um ein defektes Speicherelement handeln.

Arbeitsweise des Befehls

Man kann Speicherbereiche mit dem LDIR-Befehl sehr schnell löschen, indem man Ziel und Quelle überlappen läßt, z.B.:

```
ld hl,start      ;Startadresse laden
ld de,start+1    ;Überlappendes Ziel
ld bc,ende-start-1  ; Anzahl der Bytes
ld (hl),wert     ; Datenwert
ldir
```

Abb. 3-1:
Programm „Speicherlöschen"

Jedoch ist es bei dieser Methode nicht möglich zu prüfen, ob der Wert auch ankam. Dazu wurde im HEXMON-Programm DOFUELL der Befehl in einzelne Operationen zerlegt und eine Abfrage eingebaut.

3.4 3 vgl – Speicherbereiche vergleichen

Da zum Beispiel beim MVE-Befehl nicht geprüft wird, ob der Datenwert auch im Ziel ankam, ist es ganz gut einen getrennten Befehl dafür zu haben. Es wird genau wie beim MVE-Befehl eine VON-Adresse, eine BIS-Adresse und eine NACH-Adresse abgefragt. Der Bereich zwischen VON und BIS (inklusiv) wird mit dem entsprechenden Bereich bei NACH verglichen. Tritt ein Fehler auf, so wird die Meldung PRF xxxx ausgegeben. Dabei steht anstelle von xxxx eine Adresse.

Ist kein Fehler aufgetreten, so wird wieder -bEF- ausgegeben.

Arbeitsweise des Befehls

In dem Unterprogramm DOVGL wird der Vergleich in der Schleife DO1VGL durchgeführt. Ein Abbruch erfolgt, wenn der Inhalt der Quelle nicht mit dem Inhalt des Zielbereichs übereinstimmt. Quelladresse und Zieladresse werden so lange erhöht, bis die Endadresse erreicht ist.

3.5 opt – Optionen

Diese Taste arbeitet nur im Zusammenhang mit anderen Befehlen. Wird sie getrennt betätigt, so erscheint auf der Anzeige die Meldung --------.

Mit der opt-Taste wird im allgemeinen das Ausgabeformat umgeschaltet. So kann man z.B. beim IOL-Befehl zwischen dualer und sedezimaler Zahlendarstellung umstellen, oder beim UMW-Befehl zwischen dezimaler und sedezimaler Eingabe.

3.6 – – Minus

Diese Taste arbeitet nur im Zusammenhang mit anderen Befehlstasten. Damit kann man im allgemeinen eine Speicherzelle zurückschreiten.

3.7 4 reg – Register bearbeiten

Die Registerinhalte des Z80 können mit diesem Befehl angesehen oder modifiziert werden. Die Registerinhalte werden jeweils bei einem STEP-Befehl oder beim Start eines Programmes zuvor in den Z80 übertragen und nach Ablauf des STEP-Befehls oder bei Erreichen eines RST6-Befehls wieder gerettet.

Nach Eingabe der REG-Taste erscheint zuerst der Inhalt des Registerpaars AF: Anzeige AF a.a.f.f. a.a. ist der Inhalt des Akkus, und f.f. der Inhalt des Flagregisters. Die Punkte geben an, daß man diese Werte nun verändern kann. Will man nur den Akkuinhalt verändern, so muß man den alten Inhalt des Flag-Registers wieder neu eingeben.

Die Bedeutung der einzelnen Bits innerhalb des Flagregisters:

```
S Z x H x P/V N C

sign=Vorzeichen
  zero=Null
      halfcarry=Übertrag für Korrektur
          parity/overflows=Paritäts/Überlauf-Bit
            Add/Sub-Merker
                carry=Übertrag
```

Abb. 3-2: Flagregister

x steht dabei für Stellen, die im Register beliebige Werte annehmen können und keine weitere Bedeutung haben.

Gibt man CR oder „+" ein, so erscheint das nächste Registerpaar. Mit „-" kann man das vorherige Registerpaar ansehen. Die Register erscheinen dabei in folgender Reihenfolge:

AF, BC, DE, HL, SP, IX, IY, AF', BC', DE', HL', rI

Zum Registerpaar rI ist noch zu sagen, daß die linke Anzeigenseite das R-Register darstellt, und die rechte das I-Register: rI r.r.i.i. .

Wird vor der Eingabe von CR eine Zahl eingegeben, so wird diese in das Registerpaar übernommen. Den Befehl beendet man durch Drücken der BEF-Taste.

Arbeitsweise des Befehls

Die Registerpaare werden in einem eigenen Speicherbereich im RAM festgehalten. Dieser RAM-Bereich kann dann durch den Befehl angesehen oder modifiziert werden.

Das Unterprogramm REGAUS sorgt dafür, daß der richtige Registername auf der Anzeige erscheint. Dabei wurde im Fall des IX-Registers für X der Buchstabe H verwendet, da er auf der Anzeige dem X am ähnlichsten ist.

REGAUS liefert auch die jeweils aktuelle Adresse des Registerpaars im Speicher. Dazu befindet sich vor jedem Registernamen in der Tabelle REGTAB die jeweilige Adresse. Dieses Verfahren hat den Vorteil, daß die Registerreihenfolge nicht von der realen Lage im Speicherbereich abhängig ist.

3.8 5 prf – Kassette Prüflesen

Nach Drücken der Taste wartet der Computer auf die Eingabe von der Kassette. Dazu muß die Baugruppe CAS abgeglichen sein und im BUS stecken. Beim Prüflesen wird der aktuelle Speicherinhalt mit dem auf der Kassette aufgezeichneten verglichen. Stimmt der Inhalt nicht überein, so erscheint auf der Anzeige die Meldung PRF xxxx. Anstelle von xxxx ist die Speicheradresse angegeben, bei der der Fehler auftrat. Eine weitere Fehlermöglichkeit ist ein Prüfsummfehler, der mit CHE auf der Anzeige gemeldet wird. Trat kein Fehler auf, so wird auf der Anzeige die Meldung St = xxxx ausgegeben. Anstelle von xxxx steht die Startadresse des geprüften Programms. Dabei ist dies die VON-Adresse, die beim Abspeichern angegeben wurde.

Arbeitsweise des Befehls

Beim Prüflesen wird die Speicherzelle MERKCAS auf 1 gesetzt. Dann wird wie beim Laden verfahren. Dadurch, daß die Zelle auf 1 liegt, wird verhindert, daß die Daten geladen werden. Sie werden nur verglichen.

3.9 6 spe – Auf Kassette speichern

Nach Eingabe des Befehls wird eine VON-Adresse angefordert. Dies ist die erste Speicherzelle des Speicherbereichs, der auf die Kassette gespeichert wird. Danach wird eine BIS-Adresse angefordert. Dies ist die letzte Speicherzelle, die noch mit abgespeichert werden soll.

Die Aufzeichnung erfolgt mit einer Prüfsumme, so daß bei der Wiedergabe automatisch kontrolliert werden kann, ob Lesefehler aufgetreten sind.

Bevor man die CR-Taste drückt, nachdem man die BIS-Adresse eingegeben hat, muß man den Kassettenrecorder starten. Nachdem die Daten aufgezeichnet sind, erscheint die Meldung -bEF- auf der Anzeige.

Arbeitsweise des Befehls

Bei der Aufzeichnung werden als erstes 20 Bytes FF aufgezeichnet. Diese dienen später dem Einrasten der Taktgewinnungsschaltung auf der CAS-Baugruppe (siehe Buch: Mikrocomputer selbstgebaut ..., Abb. 5.6.2). Dann wird zur Erkennung des Starts die Sequenz 00 und 3A aufgezeichnet. Damit wird erreicht, daß bei der Wiedergabe auch der richtige Datenstart erkannt werden kann, denn die Software prüft ob die Sequenz FF 00 3A aufgetreten ist. Nun folgt die eigentliche Information. Als erstes die Startadresse, dann die Endadresse und schließlich die Daten. Als letztes wird eine zwei Bytes umfassende Prüfsumme aufgezeichnet. Diese Prüfsumme erstreckt sich von der Startadresse bis zum letzten Byte. Damit kann man später die Daten auf korrekte Wiedergabe prüfen.

3.10 7 lad - Von Kassette laden

Daten werden von der Kassette mit der CAS-Baugruppe in den Speicher geladen. Dabei wird geprüft, ob die Daten auch wirklich im Speicher ankommen und falls nicht, wird die Meldung PRF xxxx ausgegeben. xxxx ist dann die Adresse, bei der der Fehler auftrat. Dies kann zum Beispiel passieren, wenn man beim Abspeichern einen falschen Bereich angegeben hat, also auch Informationen abgespeichert hat, die nicht im RAM lagen. Oder wenn ein Speicherfehler auftritt.

Tritt am Ende der Wiedergabe die Meldung CHE auf, so liegt ein Prüfsummenfehler vor. Dies beruht z.B. auf einen Bandlesefehler.

Ist aber alles ok, so erscheint auf der Anzeige die Meldung St = xxxx, wobei xxxx die Startadresse des geladenen Programms ist, also die Adresse, die als VON-Adresse beim Abspeichern angegeben wurde. Gibt man dann eine weitere Taste ein, so erscheint die Meldung -bEF- und man kann normal weiterarbeiten.

Wenn der Polaritätsschalter auf der CAS-Baugruppe falsch eingestellt ist, erkennt das Programm den Start der Aufzeichnung nicht und bleibt im Lade-Mode. Man stellt dann einfach den Schalter um und spult das Band zurück. Dann beginnt ein neuer Versuch.

Die Stellung des Schalters ist von Laufwerk zu Laufwerk verschieden.

Beim Laden selbst sind die Daten zusätzlich auf den Punkten der Anzeige sichtbar. Die linke Anzeige stellt dabei das Bit 7 dar und die rechte Bit 0. Man kann damit feststellen, ob die CAS-Baugruppe überhaupt arbeitet.

Arbeitsweise des Befehls

MERKCAS wird dazu mit 0 belegt, und dadurch wird vor dem Speicherprüfen zuerst ein Datenwert dorthin geschrieben. Die Routine GETRI sorgt dafür, daß die gelesenen Daten auf den Punkten der Anzeige sichtbar werden. Die Anzeige wird in dieser Betriebsart nicht gemultiplext, weshalb die Punkte heller als sonst leuchten. Ein multiplexen wäre aus zeitlichen Gründen nicht möglich oder nur umständlich realisierbar.

3.11 step - Einzelschritt

Damit kann man Z80-Programme im Einzelschritt-Verfahren durchlaufen. Dabei ist es möglich, zwischen verschiedenen Anzeigen zu wählen.

Zunächst wird nach der Adresse gefragt, bei der die Programmausführung beginnen soll. In der Anzeige erscheint: Adr x.x.x.x. . Anstelle von xxxx steht eine Adresse. Dort kann man die neue Adresse hinschreiben. Nachdem man CR eingegeben hat, erscheint die Adresse im linken Feld, und rechts der Inhalt der Speicherzelle. Tippt man nach Adresseingabe die Taste step, so wird dieser Befehl auch gleich ausgeführt, und in der Anzeige erscheint der nächste auszuführende Befehl. Nun kann man mit CR oder step Befehl für Befehl ausführen.

Man kann auch das Anzeigeformat umschalten. Drückt man die Taste opt, so wird nur der Befehl auf der Anzeige ausgegeben, jedoch mit so vielen Stellen, wie der Befehl tatsächlich belegt, also z.B. C30081 oder 18F3 etc. Drückt man erneut opt, so erscheint wieder das ursprüngliche Format.

Wenn man die Taste speich drückt, so erscheint in der rechten Anzeigenhälfte die Ausgabe x.x.x.x. . Jetzt kann man eine beliebige Speicheradresse eingeben und mit CR die Eingabe beenden. Auf der Anzeige erscheint aaaa-dd-, wobei aaaa die Adresse ist und dd der Inhalt. Mit step oder CR kann man nun das Programm weiter im Einzelschritt bearbeiten. Es wird immer der aktuelle Stand der angewählten Speicherstelle ausgegeben.

Man kann auch Register ansehen und bearbeiten. Dazu drückt man die Taste reg. Es erscheint das zuletzt verwendete Registerpaar. Mit „+" und „–" kann man das gewünschte Registerpaar einstellen und durch eine Zahleingabe mit nachfolgendem CR auch mit einem neuen Wert belegen. Danach wird dieses Registerpaar dauerhaft angezeigt, und nach jedem step oder CR erscheint der aktuelle Inhalt.

Mit der Taste start kann man das Programm auch schneller durchlaufen. Mit opt wird der Start-Befehl wieder gestoppt. Er wird auch gestoppt, wenn eine Haltepunkt-Adresse (Breakpoint) aufgetreten ist. Dann kann man durch Drücken von step den Haltepunkt überspringen und das Programm danach weiterlaufen lassen.

Bei der Start-Funktion wird das Z80-Programm mit ca. 1ms pro Schritt ausgeführt. Dadurch ergibt sich eine sehr starke Verlangsamung des Programmablaufs, weshalb man damit auch normalerweise schnell ablaufende Vorgänge besser beobachten kann.
Durch Drücken der Taste BEF kann man den Einzelschritt-Modus jederzeit verlassen.

Arbeitsweise des Befehls

Den Kern der Routine DOSTEP bildet das Unterprogramm STEP. Dieses Unterprogramm ist in der Lage, einen Z80-Befehl auszuführen. Dabei wird der Registersatz im Speicherbereich verwendet. Das Unterprogramm STEP kann auch Programme in EPROM-Bereichen bearbeiten, da es mit einer speziellen Technik arbeitet.

Das Unterprogramm STEP emuliert, wie man sagt, den Z80. Der Begriff Emulation wird immer dann verwendet, wenn ein Rechner den Befehlssatz eines anderen Rechners per Programm ausführt. In diesem Fall ist der Rechner, der Emuliert wird, identisch mit dem, der das Emulationsprogramm besitzt. In der Praxis gibt es auch Fälle, in denen das unterschiedlich ist. Der Vorteil ist z.B., daß man einen Rechner auf einem anderen ausprobieren kann, bevor der neue Rechner je gebaut wurde. Der Nachteil ist, daß die Emulation meist langsamer ist als der reale Rechner.

Zur Emulation wird bei uns der Z80-Befehl in einen RAM-Bereich geschrieben und dann dort gestartet. Sprungbefehle und Unterprogrammaufrufe werden durch Verändern des Programmzählers direkt bearbeitet.

Das Unterprogramm STEP ist recht komplex gegliedert, eine wichtige Routine darin ist jedoch das Unterprogramm LENGTH. Es hat die Aufgabe, die Länge eines Z80-Befehls zu bestimmen, und das ist nicht ganz einfach, da der Z80 keine einfachen Regeln dafür besitzt. Das Unterprogramm LENGTH wird in HEXMON auch anderweitig verwendet, so zum Beispiel zum Ausgeben der Befehle auf der Anzeige, wann immer man die richtige Anzahl von Bytes sehen will.

3.12 + - Plus

Dieses Taste ist kein einzeln verwendbarer Befehl, sie kann immer nur im Zusammenhang mit anderen Befehlen verwendet werden. So dient sie meistens dazu, die nächste Speicherzelle anzuwählen.

3.13 8 ios - IO setzen

Damit kann man einen Datenwert auf einen Z80-IO-Port ausgeben. Dazu wird als erstes die Port-Adresse verlangt. Anschließend kann ein Datenwert eingegeben werden. Dieser Datenwert wird nach betätigen von CR an das IO-Port ausgegeben.

Danach kann man einen neuen Datenwert eingeben, und dieser wird dann an die gleiche Port-Adresse ausgegeben. Dabei kann man auch beim nächsten Mal einfach nur CR eingeben, und es wird der noch in der Anzeige stehende Wert ausgegeben. Betätigt man die Taste BEF, so wird der Wert nicht ausgegeben und man gelangt zurück in den Befehls-Modus. Tippt man eine andere Taste als BEF oder CR, z.B. +, so wird der letzte Wert noch ausgegeben und danach gelangt man zurück in den Befehlsmode.

Dieser Befehl kann insbesondere zum Test von Hardware-Schaltungen verwendet werden, wenn man zum Beispiel eigene Zusatz-Karten aufgebaut hat und möchte dies ausprobieren, ohne gleich ein Testprogramm zu schreiben.

Arbeitsweise des Befehls

Die Ausgabe der Port-Werte geschieht mit dem Z80-Befehl OUT (C), E. 8080 – Benutzer wissen vielleicht, daß man beim 8080 Probleme mit diesem Befehl hatte, da beim Prozessor 8080 (und 8085) keine indirekte Port-Adressierung möglich war, also die Adresse des Portes fest im Speicher stand und nicht in einem Register, wie es beim Z80 möglich ist.

3.14 9 iol - IO lesen

Will man den Inhalt eines Z80-IO-Portes ansehen, so kann man diesen Befehl verwenden. Zuerst wird die Adresse des Portes eingegeben. Nach Eingabe von CR erscheint der Inhalt des Portes auf der rechten Seite der Anzeige in sedezimaler Schreibweise. Tippt man auf die Taste opt, so wird der Wert in dualer Schreibweise ausgegeben. Wenn man nochmals opt eingibt, wird der Wert wieder sedezimal dargestellt. Nun kann man die Taste start drücken. Dann wird der Datenport in 1ms-Abständen abgefragt und der aktuelle Wert angezeigt. Dies ist sowohl mit sedezimaler als auch mit dualer Ausgabe möglich. Abgebrochen wird entweder durch Eingabe von BEF oder CR. Bei CR kann man wieder erneut start drücken, und bei BEF gelangt man in den Befehlsmodus (-bEF-) zurück.

Arbeitsweise des Befehls

Die Ausgabe in sedezimaler oder dualer Schreibweise geschieht in dem Unterprogramm IODATAUS. Dabei entscheidet der Inhalt des Register D, ob sedezimal (D = O) oder dual (D = 1) ausgegeben werden soll. Der Datenwert steht im Akku, und nachher im Register E.

3.15 A prp – Eprom programmieren

Dazu wird die Baugruppe PROMMER benötigt, sowie POW22/26 oder eine andere Spannungsquelle. Auf der PROMMER-Karte muß der richtige DIL-Stecker sitzen, um den gewünschten EPROM-Typ einzustellen. (Siehe Buch „Mikrocomputer selbstgebaut und programmiert", Abb. 5.4.2.ff).

Es wird als erstes die VON-Adresse angefordert. Dies ist die Adresse im RAM-Speicher, von wo ab programmiert werden soll. Dann erscheint die BIS-Adresse. Dies ist die letzte Speicherzelle, die noch ins EPROM übertragen werden soll. Und schließlich wird die NACH-Adresse eingegeben. Dies ist die Adresse im EPROM, von wo ab im EPROM die Daten zu liegen kommen sollen. Diese Adresse ist normalerweise mit 0 anzugeben, es sei denn, man will mehrere Programme in einem EPROM abspeichern. Dann erscheint die Meldung Cr, und man kann jetzt das EPROM in die Fassung einsetzen. Dann drückt man die Taste CR erneut, und der Programmiervorgang beginnt. In der linken Anzeigenhälfte wird die aktuelle Adresse im RAM angegeben, und rechts der programmierte Datenwert. Tritt ein Programmierfehler auf, so erscheint die Meldung burnEd auf der Anzeige. „Burned" bedeutet zu deutsch „gebrannt", und man meint damit den Programmiervorgang.

Arbeitsweise des Befehls

Der EPROM-Programmierer muß auf 50ms Monoflop-Zeitdauer eingestellt werden. Dazu kann man den Befehl prm verwenden. Dann wird jeweils 50ms lang ein Byte programmiert und anschließend ca. 10ms gewartet, bis das Monoflop wieder startbereit ist. Nach jedem Programmiervorgang wird geprüft, ob das Byte auch wieder gelesen werden kann. Am Anschluß werden dann nochmals alle Zellen auf Übereinstimmung geprüft. Will man die Karte ohne EPROM testen, so muß man hier einen Speicherbereich, der mit FF gefüllt ist, als Datenmuster verwenden.

3.16 B prl – Eprom lesen

Der Inhalt eines EPROMs kann in einen Speicherbereich übertragen werden. Dazu wird als erstes die VON-Adresse eingegeben, dies ist die Adresse im EPROM. Normalerweise ist sie 0. Dann wird die BIS-Adresse eingegeben, normalerweise ist dies die Endadresse des EPROMs. Wenn man EPROMs bearbeiten will, die größer sind als der verfügbare RAM-Speicher, so kann man sie auch stückchenweise einlesen. Als letztes wird die

NACH-Adresse abgefragt, und dies ist die Adresse des RAM-Speichers, also wohin die Daten geladen werden sollen. Gibt es beim Laden einen Vergleichsfehler im RAM (z.B. kein RAM dort), so erscheint die Fehlermeldung PRF xxxx mit der Fehleradresse (xxxx).

3.17 start – Programme starten

Damit lassen sich Programme ausführen. Dazu wird die Startadresse eingegeben und CR betätigt. Das Anwenderprogramm wird gestoppt, wenn eine Haltepunkt-Adresse auftritt. Diese muß aber hier unbedingt in einem RAM-Adreßbereich liegen, da dorthin ein RST6-Befehl geschrieben wird. Der ursprünglich dort stehende Befehl wird gerettet und beim Wiedereintritt in HEXMON (durch Adresse 30H) an die alte Stelle zurückgeschrieben.

Alle Register werden aus dem speziellen Speicherbereich zuvor in den Prozessor geladen und beim Auftreten des Haltepunktes gerettet.

3.18 speich – Speicherbereiche modifizieren

Damit ist es möglich, den Speicherinhalt anzusehen und zu verändern. Zuerst wird die Adresse eingegeben, dann erscheint der Speicherinhalt auf der rechten Seite. Man kann nun einen neuen Wert eingeben und CR drücken, dann erscheint der Inhalt der nächsten Speicherzelle. Wenn man den Wert belassen will, so gibt man nur die Taste CR ein. Tippt man die Taste + ein, so gelangt man auch zur nächsten Speicherzelle, ein ggf. neu eingetippter Wert wird jedoch nicht übernommen. Bei Betätigen der „–"-Taste erscheint die Speicherzelle mit der nächst niederen Adresse. Wenn man die Taste opt drückt, so wird nur der Befehlscode auf der Anzeige ausgegeben, und zwar mit so vielen Bytes, wie der Befehl belegt, also z.B. C35581 oder 18FE oder 47. Dabei gelangt man mit der CR-Taste zum nächsten Befehl, und mit der „+"-Taste zur nächsten Speicherzelle. Bei C35581 schreitet die CR-Taste und drei Speicherzellen weiter, und die „+"-Taste um eine Speicherzelle. Mit der „–"-Taste gelangt man eine Speicherzelle zurück. Drückt man erneut die opt-Taste, so wird wieder die Adresse und der Inhalt angegeben.

Arbeitsweise des Befehls

Mit dem Unterprogramm LENGTH ist es möglich die Anzahl der Bytes, die ein Z80-Befehl belegt, zu ermitteln. Dieses Unterprogramm wird zur Ausgabe im Z80-Befehlsmodus (opt) verwendet.

3.19 C prm – Eprom-Programmierer abgleichen

Ein sehr nützliches Hilfsmittel ist diese eingebaute Meßroutine, denn sie kann ein Oszilloskop ersparen. Wenn die PROMMER-Karte eingesteckt ist, erscheint auf der Anzeige die Monoflop-Zeitdauer in Millisekunden. Sie muß mit dem Trimmer auf der

PROMMER-Karte auf 50ms abgeglichen werden. Der Wert muß dabei nicht bis zur letzten Ziffer eingestellt werden. Ist die PROMMER-Karte nicht eingesteckt, so erlischt die Anzeige, und man muß die RESET-Taste an der SBCII-Karte betätigen.

Arbeitsweise des Befehls

Vorausgesetzt wird bei der Meßroutine, daß die SBCII-Karte mit einem 4MHz-Takt arbeitet. Werden 2 MHz verwendet, dann stimmt die Zeitausgabe nicht mehr. Die Messung erfolgt durch eine Zählschleife, die die Pulsdauer des Monoflops mißt. Das Monoflop wird jedesmal neu angestoßen, um wieder einen Puls zu erzeugen. Beim Meßvorgang darf man keine Eproms im Programmier-Sockel lassen, da diese möglicherweise zerstört würden.

3.20 D per – Periodendauer messen

Hiermit kann man die Periodendauer eines Signals in einem bestimmten Meßbereich messen. Gedacht ist dieses Programm zum Abgleich der CAS-Baugruppe. Diesmal genügt es aber nicht, die Baugruppe einfach auf dem BUS zu stecken, sondern man muß eine Leitung legen.

Dazu wird auf der IOE-Karte, mit der auch HEXIO verbunden ist, Bit 7 des IO-Ports 0 mit einer Leitung verbunden. Dies ist die Meßleitung (Abb. 1-3). Der Anschluß erfolgt in der Nähe des Platinenrandes der IOE-Karte (mit Messeingang beschriftet). Man kann den Anschluß auch auf der HEXIO-Karte finden und dort anlöten.

Diese Leitung wird zur Periodendauermessung mit Punkt c (Pin 4 des 6850) auf der CAS-Baugruppe (Lötseite) verbunden. Dann erscheint die Periodendauer in Mikrosekunden.

Man kann nun mit Trimmer Tr1 (CAS-Baugruppe) die Periodendauer auf ca. 833μs abgleichen.

Arbeitsweise des Befehls

Auch hier wird die Zeitdauer mit einer Zählschleife ermittelt. Da jedoch die Meßzeit sehr kurz ist, wird die Messung 10 mal durchgeführt und das Mittel ausgegeben, um die Meßgenauigkeit zu erhöhen. Der Meßbereich reicht von ca. 100 μs bis ca. 30000 μs. Ist die Periodendauer des gemessenen Signals größer oder kleiner als dieser Wertebereich, so erhält man eine falsche Angabe.

3.21 E pul – Pulsbreite messen

Dieser Befehl arbeitet wie der vorherige Befehl, nur daß die 0-Signaldauer gemessen wird. Dazu wird das Meßkabel mit dem Punkt b (Pin 3 des 6850) auf der CAS-Baugruppe, Lötseite, verbunden. Mit dem Trimmer Tr2 kann man dann einen Wert von ca. 625 μs einstellen. Dazu muß die CAS-Baugruppe auch auf Aufnahme gestellt werden und der Teststecker (siehe Abb. 5.6.12 in Mikrocomputer selbstgebaut und programmiert) eingesteckt sein.

Arbeitsweise des Befehls

Ähnlich wie bei der Periodendauer-Messung.

3.22 F umw – Zahlensysteme umwandeln

Damit kann man eine sedezimale Zahl in eine Dezimalzahl umwandeln, und umgekehrt.

Zuerst wird die Eingabe einer sedezimalen Zahl verlangt. Gibt man sie ein und betätigt CR, so erhält man das sedezimale Ergebnis. Dabei wird die sedezimale Zahl als Zweier-Komplementzahl behandelt.

Drückt man dann nochmals CR, so kann man eine neue Zahl eingeben. Drückt man aber opt, so erscheint diese Zahl wieder im sedezimalen Zahlensystem. Folgt danach CR, so kann man eine dezimale Zahl eingeben, die nach CR in eine sedezimale Zahl umgerechnet wird. Man kann bei der dezimalen Zahl auch ein negatives Vorzeichen durch betätigen der „–"-Taste eingeben, jedoch darf man dann keine weiteren Ziffern der Zahl mehr eintippen.

Bei der Ausgabe auf der Anzeige wird das Zahlensystem durch voranstellen des Buchstabens S (sedezimal) oder d (dezimal) gekennzeichnet.

Arbeitsweise des Befehls

Die Umrechnung einer sedezimalen Zahl in eine dezimale Zahl geschieht durch eine fortlaufende Division durch 10. Dabei wird immer der Rest als Ergebnis verwendet.

Bei der Umrechnung einer Zahl vom dezimalen ins sedezimale Zahlensystem wird die gemerkte Zahl mit 10 multipliziert und der neue Wert addiert. Siehe auch Programmlisting DOCONV.

3.23 BEF – Befehle eingeben

Mit dieser Taste gelangt man immer in den Befehlsmodus -bEF- zurück. Drückt man sie dann erneut, so erscheint auf der Anzeige -------- als Fehlerhinweis. Man kann dann aber immer eine Befehlstaste eingeben.

3.24 CR – Zahleingabe abschließen

Diese Taste ist kein eigener Befehl, sondern wird an anderen Befehlen verwendet. Meist wird sie zum Abschluß von Zahleinheiten verwendet, aber auch zum Fortschreiten oder Erhöhen.

4 Unterprogramme für den Anwender

In diesem Kapitel sollen alle nützlichen Unterprogramme beschrieben werden, die man für eigene Programme verwenden kann.

Alle diese Unterprogramme sind durch eine Sprungtabelle am Anfang des HEXMON zusammengefaßt. Wichtig ist, daß man nur Adressen dieser Sprungtabelle verwendet und nicht die eigentlichen Unterprogrammadressen. Denn sollte sich HEXMON mal ändern, so müßte man alle eigenen Programme umschreiben.

Verwendet man dennoch Unterprogramme mit absoluten Adressen inmitten von HEXMON, so sollte man am Anfang seines eigenen Programms ebenfalls eine Sprungtabelle konstruieren, so daß man bei Änderung dieser Adressen nur die Adressen in der Sprungtabelle verändern muß und nicht an unterschiedlichen Stellen des eigenen Programms.

4.1 RI 3

Das Unterprogramm wartet auf ein Zeichen, das vom Kassettenrecorder gelesen wird. Wenn das Zeichen ankommt, so steht es anschließend im Akkumulator (Register A) des Z80. Das Unterprogramm funktioniert nur zusammen mit der CAS-Baugruppe.

```
Beispiel: CD 03 00 ANF: CALL RI ;Warten bis 0 auftritt
          FE 00          CP 0
          20 F9          JR NZ, ANF
```

4.2 POO 6

Ein Zeichen, das im C-Register steht, wird zum Kassettenrecorder übertragen. Dieses Unterprogramm arbeitet nur bei vorhandener CAS-Baugruppe.

```
Beispiel: 0E 41     LD C, 41H ; Wert 41H ausgeben
          CD 06 00 CALL POO
```

4.3 ANZEIGE 9

Im ANZFELD auf Adresse 8000H bis 8007H stehen die einzelnen Segmentcodes. 8000H ist dabei die linke Anzeige. Bei Aufruf des Unterprogramms wird eine Anzeigestelle für ca. 1ms ausgegeben. Damit man ein kontinuierliches Bild erhält, muß man dieses Unterprogramm oft aufrufen.

Gleichzeitig erhält man im Register A einen Tastencode, falls eine Taste gedrückt wurde, sonst erscheint der Wert FF. Dabei ist zu beachten, daß immer nur die aktuelle Spalte abgefragt wird, also bei Drücken einer Taste nur bei jedem achten Mal der Tastencode angegeben wird.

```
Beispiel: CD 09 00 LP: CALL ANZEIGE ; Ausgabe ANZFELD
          18 FB        JR LP        ; dauerhaft
```

4.4 HOLETASTE C

Dieses Unterprogramm wartet solange, bis eine Taste des Tastenfeldes gedrückt wird. Dabei wird in dieser Zeit das ANZFELD auf der Anzeige ausgegeben. Der Tastencode erscheint dann im Akkumulator.

```
Beispiel: CD 0C 00   LP:  CALL HOLETASTE;  Warten bis
          FE 47           CP 47H         ;  BEFEX gedrückt
          20 F9           JR NZ, LP      ;
```

Die Tastencodes kann man im HEXMON-LISTING (Anhang G) auf einer der ersten Seiten nachsehen.

4.5 TONUM F

Im Akku steht vorher ein Tastencode. Wenn es eine Taste des numerischen Feldes (0 bis F) ist, so erscheint im Akku der Wert 0 bis F entsprechend der Beschriftung der Tasten. Ist es kein numerischer Wert, so erscheint im Akku der alte Tastencode und das CARRY ist gesetzt.

```
Beispiel: CD 0C 00   CALL HOLETASTE;  Tastencode holen
          CD 0F 00   CALL TONUM     ;  und umrechnen
```

4.6 TOSEG 12

Im Akku ist ein Datenwert 0 bis F. Nach Aufruf dieser Routine befindet sich im Akku der entsprechende Siebensegmentcode, der für eine Ausgabe an das Segmentport direkt verwendet werden kann.

```
Beispiel: 3E 0F        LD A, 0FH          ;  0F umrechnen
          CD 12 00     CALL TOSEG         ;  in Segmentcode
          32 00 80     LD (ANZFELD),A     ;  und an erster Stelle
          CD 09 00  LP: CALL ANZEIGE       ;  dann ausgeben
          18 FB        JR LP              ;  ohne Ende
```

4.7 PRINT 15

Im Registerpaar HL wird eine Adresse angegeben. 8 Bytes werden dann beginnend bei dieser Adresse ins ANZFELD übertragen. Das Unterprogramm wird benötigt, um Texte in das Anzeigefeld zu bringen.

Beispiel:

```
8100:  21 0B 81          LD HL, TABELLE  ; Adresse laden
8103:  CD 15 00          CALL PRINT      ; ausgeben
8106:  CD 09 00    LP:   CALL ANZEIGE    ; und anzeigen
8109:  18 FB             JR LP           ; immer wieder
810B:  87 86 92 87                       ; TEST
810F:  FF FF FF FF                       ; Leerfeld
```

Die Segmentcodes für Texte kann man im Anhang C nachschlagen.

4.8 PRTAC 1B

Das Registerpaar IX enthält die Adresse im ANZFELD, von wo ab der sedezimale Zahlenwert des AKKUs abgelegt werden soll. Er wird mit zwei Ziffern abgelegt. Nach dem Aufruf zeigt das IX-Registerpaar hinter das letzte Zeichen.

```
Beispiel: 3E 5A          LD A, 5AH       ; Wert im Akku laden
          DD 21 00 80    LD IX, 8000H
          CD 18 00       CALL PRTAC
          CD 09 00  LP:  CALL ANZEIGE
          18 FB          JR LP
```

Der Rest der Anzeige bleibt undefiniert; man müßte die Anzeige mit dem Unterprogramm CLEAR zuerst löschen.

4.9 PRTHL 1B

Hier werden vier sedezimale Ziffern, die im HL-Registerpaar stehen, beginnend bei der Adresse, die im IX-Registerpaar steht, ausgegeben. Anschließend zeigt HL auf die nächste Speicherzelle.

```
Beispiel: 21 5A 12       LD HL, 125AH    ; Testmuster laden
          DD 21 00 80    LD IX, 8000H    ; ANZFELD
          CD 1B 00       CALL PRTHL      ; dort ausgeben
          CD 09 00  LP:  CALL ANZEIGE    ; und anzeigen
          18 FB          JR LP           ; immerfort
```

4.10 PRTBIN 1E

IX zeigt normalerweise auf den Anfang von ANZFELD; der Akkuinhalt wird binär dort abgelegt.

```
Beispiel: 3E 5A          LD A, 01011010B ; Test binär
          DD 21 00 80    LD IX, ANZFELD  ; Zieladresse
          CD 1E 00       CALL PRTBIN     ; Dort ausgeben
          CD 09 00  LP:  CALL ANZEIGE    ; und anzeigen
          18 FB          JR LP           ; immerfort
```

4.11 PRTDEZ 21

Diesmal zeigt IX auf die letzte Stelle im ANZFELD, HL enthält die auszugebende Zahl. Die Zahl wird mit Nullunterdrückung rechtsbündig und mit Vorzeichen ausgegeben. Sie belegt maximal 6 Stellen (- 32768). Die Ausgabe erfolgt im Dezimalsystem.

```
Beispiel: 21 D2 04          LD HL, 04D2H      ; Test dezimal (04D2H = 1234)
          DD 21 07 80       LD IX, ANZFELD+7; ans Ende ANZFELD
          CD 21 00          CALL PRTDEZ       ; dort ausgeben
          CD 09 00    LP:   CALL ANZEIGE      ; und anzeigen
          18 FB             JR LP             ; unbegrenzt
```

Wenn man das Programm startet, erscheint auf der Anzeige Adr 1234, da der Text Adr noch in ANZFELD steht.

4.12 GETC 24

Eine zweiziffrige Sedezimalzahl wird eingelesen. IX enthält die Adresse im ANZFELD. Im Register C steht vor dem Aufruf der „Default"-Wert, also ein Wert, der auf der Anzeige erscheint, bevor eine Zahl eingetippt wird.

Nach dem Aufruf steht dort der neue Wert, falls ein neuer eingetippt wurde.

Die Eingabe wird beendet, wenn eine nichtnumerische Taste gedrückt wird. Der Tastencode steht dann im Register A.

```
Beispiel: DD 21 00 80       LD IX, ANZFELD    ; von Anfang an
          0E 12             LD C, 12H         ; Default-Wert
          CD 24 00          CALL GETC         ; einlesen neues C
```

4.13 GETHL 27

Vier Sedezimal-Ziffern werden eingelesen. Der Wert steht diesmal im HL-Register, und der Default-Wert ebenfalls. (Siehe GETC).

```
Beispiel: DD 21 00 80       LD IX, ANZFELD    ; linke Hälfte
          21 00 00          LD HL, 0          ; Default z.B. 0
          CD 27 00          CALL GETHL        ; und neuen Wert
```

4.14 GETDEZ 2A

IX zeigt auf der letzten Stelle im ANZFELD; sonst wie GETHL, nur daß eine dezimale Eingabe verwendet wird. Dabei kann man die Taste „–" am Schluß der Eingabe drücken, um eine negative Zahl einzugeben.

```
Beispiel: DD 21 07 80       LD IX, ANZFELD+7; Rechtsbündig
          21 00 00          LD HL, 0          ; Default 0
          CD 2A 00          CALL GETDEZ       ; Wert holen
```

4.15 STEP 2D

Damit kann man einen Z80-Befehl getrennt ausführen. HL zeigt auf den auszuführenden Befehl. Dann wird der Registersatz in den Speicherbereich 801DHff. geladen und verwendet. Nach dem Aufruf zeigt HL auf den nächsten auszuführenden Z80-Befehl.

Beispiel:

```
8100:  21 0B 0E        LD HL, 0E0BH     ; SCHLEIFE Einsprung
8103:  31 00 83        LD SP, 8300H     ; Stack trennen
8103:  CD 2D 00   LP:  CALL STEP        ; Befehl ausführen
8106:  18 FB           JR LP            ; Immer weiter
```

Mit diesem kleinen Programm wird HEXMON selbst schrittweise ausgeführt. Die Anzeige zeigt die einzelnen Segmente nacheinander auf, da auch das Multiplexen verlangsamt ist. Die Adresse 0E0BH kann sich bei anderer HEXMON Version ändern, daher zuvor im aktuellen Listing nachsehen, ob dies die SCHLEIFE ist.

Andere Eingänge in HEXMON, wie 0 oder 30H, kann man bei diesem Experiment nicht verwenden.

4.16 BREAK 30

Wenn man auf diese Adresse springt, landet man wieder im HEXMON. Es werden alle Register gerettet und die Haltpunkte (BREAKPOINTS) entfernt.

Mit dem RST6-Befehl geht das am besten. Dabei wird auch der Programmzählerstand festgehalten, der als Adresse auf der Anzeige erscheint.

Beispiel: F7 RST6

Damit kann man die Ausführung eigener Programme beenden, um nachher wieder in das Monitorprogramm HEXMON zurückzugelangen.

Auf der Anzeige erscheint die Meldung brE xxxx, und man muß auf die Taste BEF drücken, um wieder in das Monitorprogramm HEXMON zurückzugelangen.

4.17 CLEAR 33

ANZFELD wird mit dem Code FF belegt. Somit werden alle Stellen der Anzeige dunkel, wenn eine Ausgabe auf das Anzeigefeld erfolgt.

```
Beispiel: CD 33 00        CALL CLEAR     ; ANZFELD wird gelöscht
          CD 09 00   LP:  CALL ANZEIGE   ; Die Anzeige bleibt
          18 FB           JR LP          ; dunkel
```

4.18 INTLOC 38

Dies ist eine Adresse, die vom Interrupt angesprungen wird, wenn IM1 verwendet wird. Dort steht ein Sprung in den RAM-Bereich, wo man selbst einen Sprung auf eine eigene Interrupt-Routine legen kann.

4.19 LENGTH 38

HL enthält die Adresse eines Z80-Befehls. Nach Aufruf erhält man im Register B die Anzahl der Bytes (1 ... 4), die der Befehl belegt.

```
Beispiel: 21  00  00      LD HL, 0        ; Adresse laden
          CD 3B 00        CALL LENGTH     ; Länge berechnen
          F7              RST6            ; Ende
```

Nach Aufruf enthält das Register B den Wert 3, da der Sprungbefehl auf Adresse 0 3 Bytes belegt.

4.20 LASTMEM 3E

Man erhält im Registerpaar HL nach Aufruf die letzte Speicherzelle, die noch RAM enthält. Es wird beginnend bei Adresse LAST (siehe Listing) gesucht und der erste zusammenhängende Bereich genommen.

```
Beispiel: CD 3E 00        CALL LASTMEN    ; HL berechnen
          F7              RST6            ; Ende
```

Normalerweise steht nach Aufruf in HL der Wert 8FFF.

ACHTUNG: Ist der RAM-Speicher sehr groß (z.B. 64k Bytes), so kann die Ausführung des Befehls lange dauern (einige Sekunden).

4.21 SUCHL 41

Wie LASTMEN, hier kann man jedoch im HL-Register die Startadresse der Suche vorgeben. Wichtig, wenn mehrere unabhängige RAM-Bereiche vorhanden sind.

```
Beispiel: 21 00 84        LD HL, 8400H    ; Start der Suche
          CD 41 00        CALL SUCHL      ; RAM-Bereich
          F7              RST6            ; Ende
```

In HL steht hier das gleiche Ergebnis wie vorher. Wenn man ein System hat, indem noch andere RAM-Bereiche vorhanden sind, kann man in HL die Startadresse angeben, und es wird dann das Ende des Bereichs gesucht.

4.22 GETRI 44

Wie RI, jedoch erfolgt zusätzlich eine binäre Ausgabe der gelesenen Daten auf den Punkten der Siebensegmentanzeige.

```
Beispiel: CD 44 00    LP: CALL GETRI
          18 FB           JR LP
```

Mit dem kleinen Programm kann man die CAS-Schnittstelle testen. Die ankommenden Daten erscheinen als Dualzahl auf dem Anzeigefeld.

4.23 NMILOC 66

Hierhin führt der NMI-Interrupt. Dort steht ein Sprung in den RAM-Bereich, und dort kann man wiederum selbst einen Sprung hinschreiben, wenn man den nicht-maskierbaren Interrupt verwenden will.

5 Wie funktioniert HEXMON

HEXMON ist ein Programm, das die Aufgabe hat, Befehle von der Tastatur anzunehmen und darauf zu reagieren.

Die Programmstruktur sieht also vereinfacht so aus:

```
START
WIEDERHOLE
   HOLE TASTE
   FUEHRE BEFEHL AUS
UNBEGRENZT
```

Abb. 5-1: HEXMON-Hauptschleife

Da verschiedene Befehl vorhanden sind, muß zur Ausführung des Befehls erstmal abgefragt werden, welcher Befehl gemeint ist, und dann wird in ein entsprechendes Unterprogramm verzweigt.

Beim Holen eines Tastencodes muß gleichzeitig die Anzeige aufrecht erhalten werden. Dazu gibt es in HEXMON ein zentrales Unterprogramm mit dem Namen ANZEIGE. Es gibt bei jedem Aufruf für ca. 1ms eine Anzeigenstelle aus und verwendet dazu den Speicherbereich ANZFELD. Eine Speicherzelle mit dem Namen ANZPOI merkt sich, welche Stelle zuletzt angezeigt wurde und wird durch ANZEIGE jedesmal um eins erhöht. Ist der Endwert von 7 erreicht, so wird ANZPOI wieder auf 0 gesetzt. Damit wird ein Zähler von 0 bis 7 verwirklicht.

ANZEIGE fragt auch die der Anzeige zugeordnete Tastenspalte ab und liefert einen entsprechenden Tastencode, falls eine Taste gedrückt wurde.

Es gibt dann noch die Routine HOLETASTE, die so lange ANZEIGE aufruft, bis ein Tastencode vorliegt.

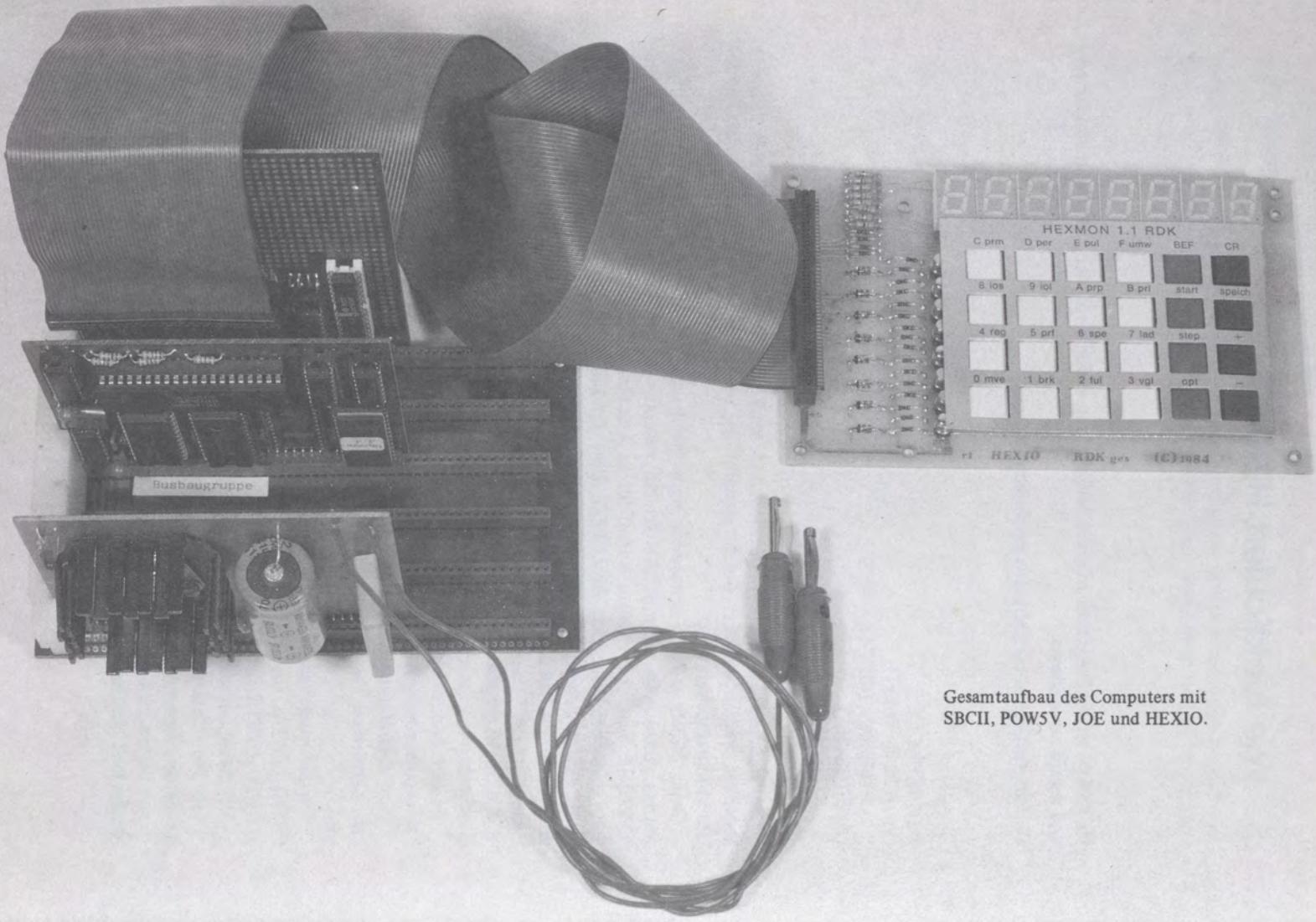
Nun übernimmt aber HOLETASTE auch das Entprellen der Tasten und sorgt dafür, daß, wenn eine Taste dauerhaft gedrückt wird, nur einmal der Tastencode übergeben wird.

Für die Ausgaben in den einzelnen Befehlen stehen dann noch Unterprogramme wie PRTHL, PRTAC und PRTDEZ zur Verfügung. Für komplexe Eingaben von Zahlen gibt es GETC, GETHL und GETDEZ.

Mit diesen Grundbefehlen kann man fast alle Programmteile realisieren.

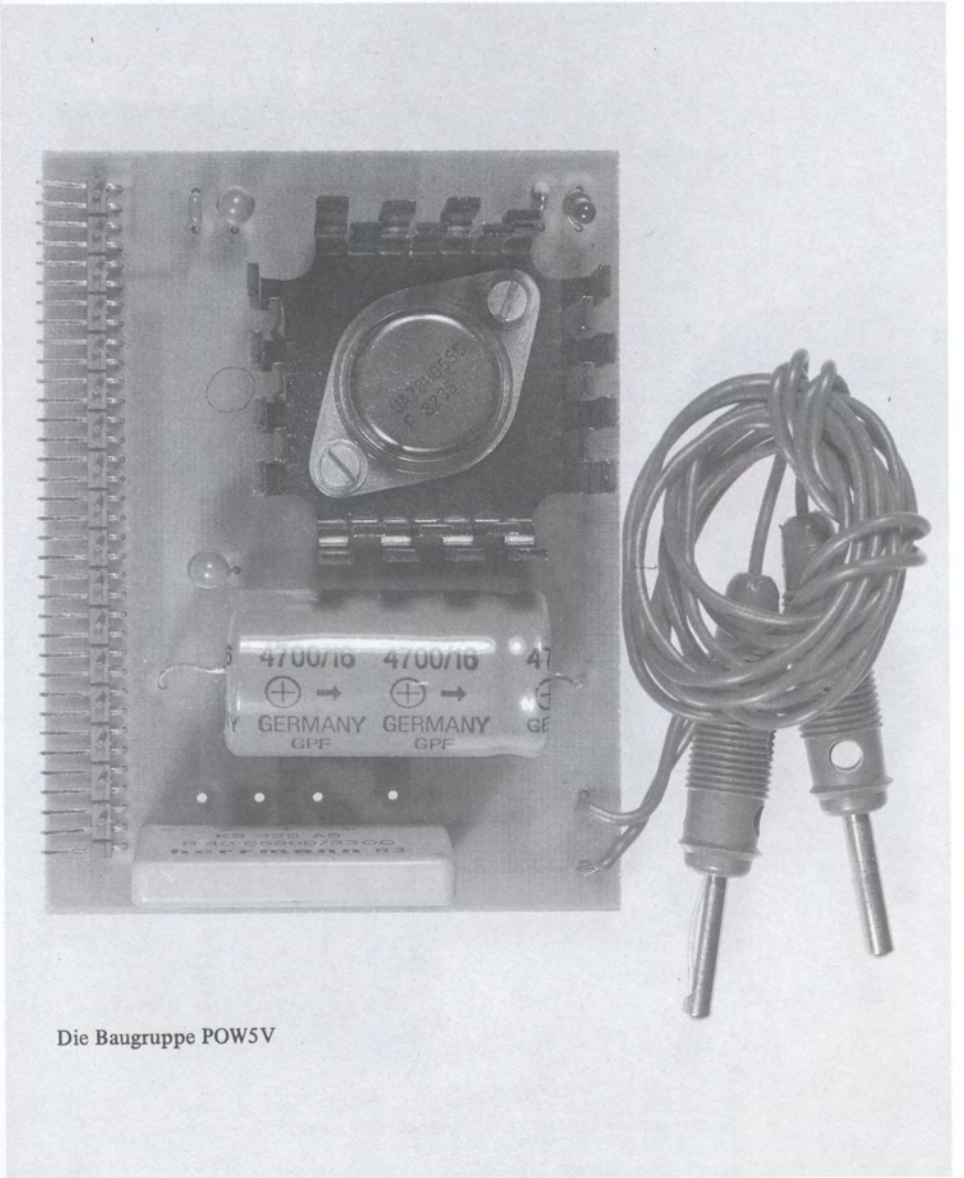
Mit dem Unterprogramm STEP lassen sich Befehle im Einzelschritt ausführen, was für die Realisierung des Einzelschritt-Befehls wichtig ist.

Am besten man blättert einmal im Listing, in dem auch zahlreiche Kommentare vorhanden sind, und lernt so HEXMON verstehen.



Gesamtaufbau des Computers mit SBCII, POW5V, JOE und HEXIO.

Anhang



Die Baugruppe POW5V

A Schaltungsaufbau und Test

In diesem Abschnitt wird der Aufbau des Computers behandelt.

A.1 POW5V

Der Computer benötigt eine Gleichspannung von 5V. Dazu dient eine Spannungsversorgung mit Brückengleichrichter und 5V-Spannungsregler (Abb. A-1).

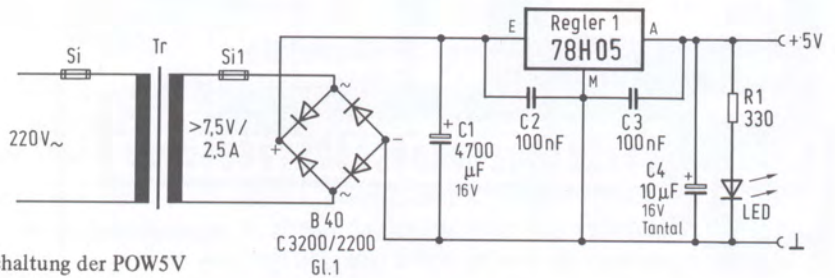


Abb. A-1: Schaltung der POW5V

Diese Schaltung kann auf einer Leiterplatte aufgebaut werden.

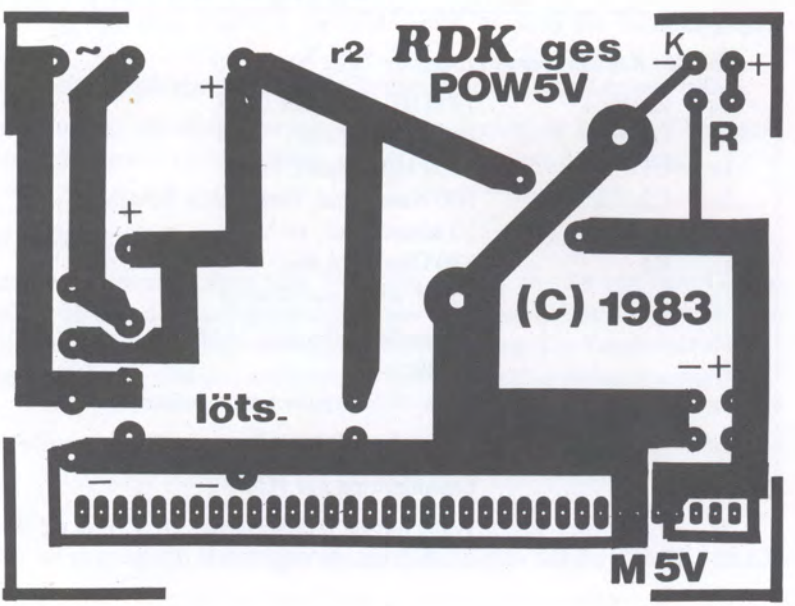


Abb. A-2: Lötseite der POW5V-Leiterplatte

Die Anordnung der Bauteile auf der Leiterplatte zeigt die folgende Abb.:

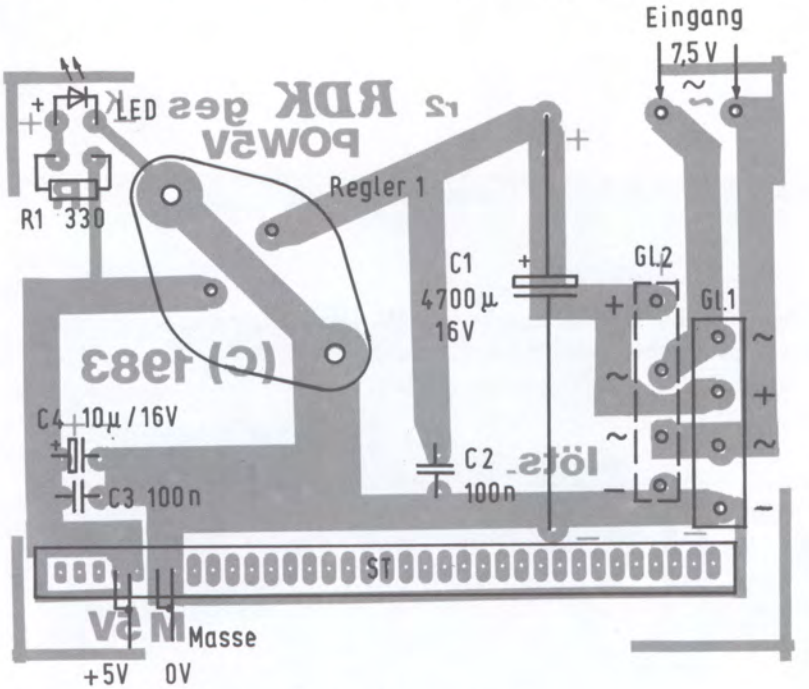


Abb. A-3: Bestückungsplan der POW5V-Baugruppe

Für den Aufbau werden folgende Teile benötigt:

1x	Regler 1	78 H 05 oder TDB 0123
1x	GL 1	B40C5000/3300
1x	C1	4700 Mikrofarad, 16 V
2x	C2, C3	100 Nanofarad, keramische Scheibe
1x	C4	10 Mikrofarad, 16 V
1x	R1	330 Ohm, 1/4 Watt
1x	LED	3 mm, rot, Leuchtdiode
1x		Steckerleiste 36polig (AMP)
1x		Kühlkörper
2x		M3 x 10 Schrauben mit Muttern
1x		Leiterplatte POW5V von ges (Bezugsquelle siehe Anhang)
1x		Bauanleitung zur POW5V
1x		Trafo mit ca. 7,5 V (besser 10 V) und ca. 30 Watt mit VDE-Zeichen.

Für alle, die sich die Baugruppe selbst aufbauen wollen und nicht den fertigen Bausatz im Handel besorgen, eine kurze Bauanleitung.

Beim Aufbau geht man wie folgt vor:

1. Einlöten des Gleichrichters.
Auf Polung achten.

2. Einschalten. Messen mit einem Voltmeter, ob am Ausgang des Gleichrichters auch eine gleichgerichtete Spannung anliegt.
Ausschalten.

3. Einlöten des Kondensators C1.
Auf Polung achten.

4. Einlöten des Spannungsreglers mit Kühlkörper.
Bitte darauf achten, daß die Anschlußdrähte des Reglers keinen Kontakt zum Kühlkörper haben, ggf. Isolierhülsen verwenden.

5. Einlöten der kleinen Kondensatoren. Beim Tantal-Kondensator (oder Elektrolyt) auf Polung achten.

6. LED und Widerstand einlöten. Wenn man die LED falsch polt, ist das nicht so schlimm, die leuchtet nur nicht.

7. Einlöten der 36poligen Stiftleiste. Bei manchen Leiterplatten sind nur 35 Pole vorhanden, dann wird einfach ein Stift der Stiftleiste abgetrennt und dann die 35polige Leiste eingelötet. Eigentlich benötigt man nur 2 bzw. 4 Pole, die anderen dienen nur der mechanischen Stabilität.

8. Einschalten. Die Spannung am Ausgang muß +5 V betragen und die LED leuchten. Ist nur die Spannung da, aber leuchtet die LED nicht, so muß die LED umgepolt werden. Ausschalten.
Also dann LED vorsichtig auslöten (LEDs sind wärmeempfindlich) und erneut testen.

9. Mit einem Voltmeter die Ausgangsspannung kontrollieren, sie muß 5 V betragen. Dabei ist eine Abweichung von +–5% zulässig, also 4,75 V min und 5,25 V max.

Wie funktioniert die Schaltung?

Am Ausgang des Transformators liegt eine Wechselspannung an, die von dem Gleichrichter GL1 in eine pulsierende Gleichspannung umgewandelt wird. Der Kondensator C1 sorgt für eine Glättung und der Regler stabilisiert die Spannung. Die Kondensatoren C2 bis C4 dienen der Entstörung und müssen für eine sichere Funktion vorhanden sein.

A.2 Prüfstift

Mit dem Prüfstift ist es möglich, Messungen an den Schaltungen durchzuführen, um z.B. Fehler zu suchen.

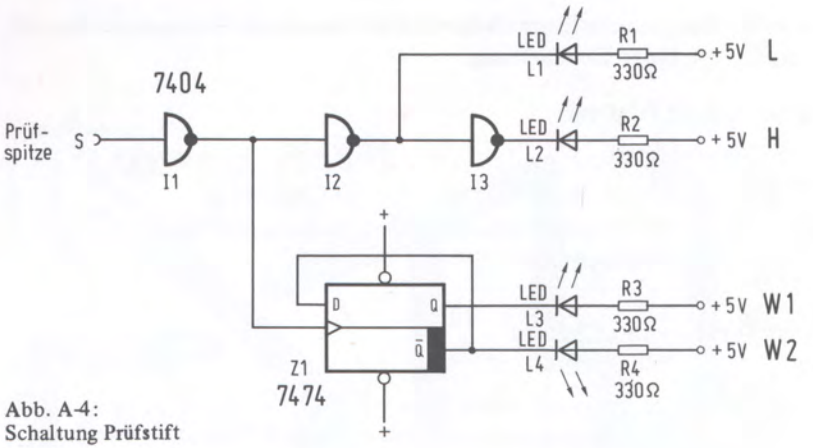


Abb. A-4:
Schaltung Prüfstift

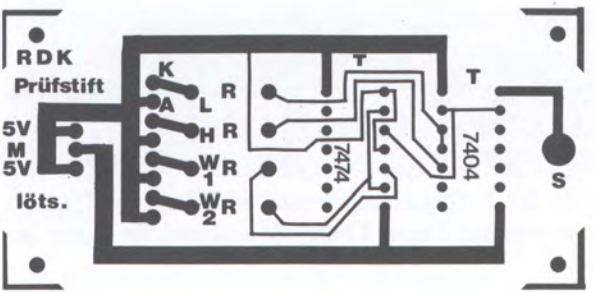


Abb. A-5:
Lötseite der
Leiterplatte Prüfstift

Die Schaltung läßt sich auf einer kleinen Leiterplatte unterbringen.

Die Bestückung der Leiterplatte ist recht einfach, da nur wenige Bauteile vorhanden sind.

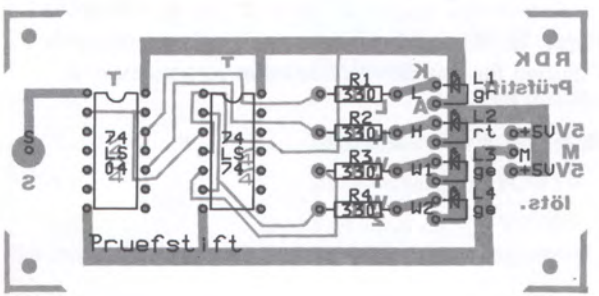
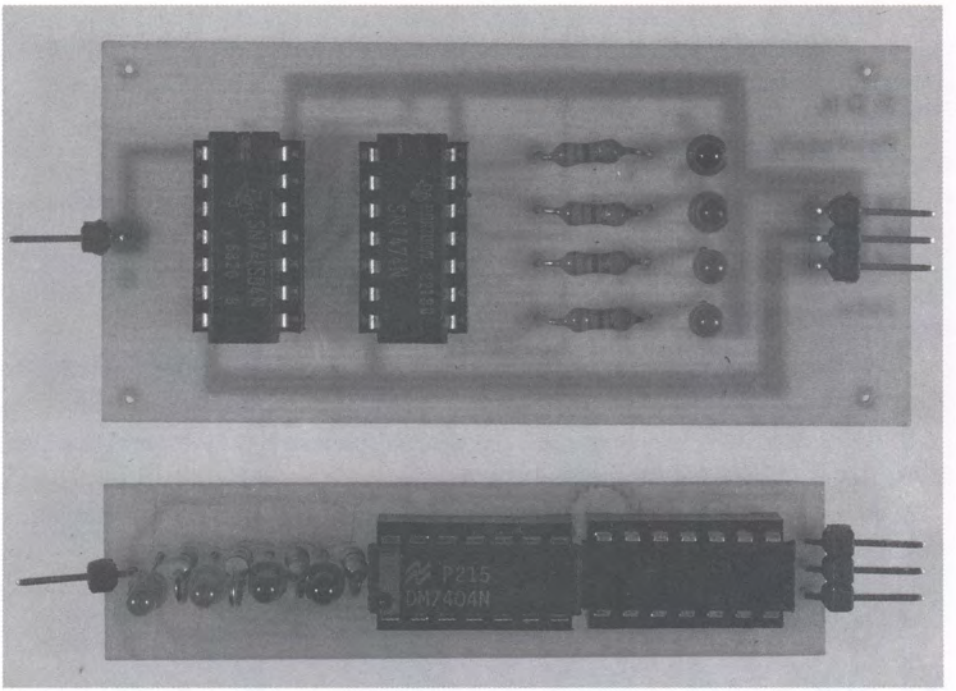


Abb. A-6:
Bestückungsplan
Prüfstift



Stückliste:

1x	I1 - I3	IC 74 04 (oder 74 LS 04)
1x	Z1	IC 74 74 (oder 74 LS 74)
1x	L1	Leuchtdiode grün 3 mm
1x	L2	Leuchtdiode rot 3 mm
2x	L3, L4	Leuchtdiode gelb 3 mm
4x	R1 - R4	330 Ohm 1/4 Watt
2x		IC-Fassung 14polig
1x		Leiterplatte Prüfstift
1x		Winkelstifte für Prüfspitze

Beim Aufbau geht man wie folgt vor:

1. Einlöten der IC-Fassungen. Falls die IC-Fassungen eine Markierung an einer der Schmalseiten besitzen, sollte man diese in Richtung der Nase der ICs anordnen. Die Nase ist auch im Bestückungsplan als Halbkreis eingezeichnet.

2. Einlöten der vier Widerstände 330 Ohm (Farbringe orange, orange, braun, gold (oder silber)).

3. Einlöten der LEDs. Dabei muß auf die Polung geachtet werden, denn sonst leuchten sie nicht. Den Anschluß K (Kathode) findet man, wenn man die Leuchtdiode im Durchlicht betrachtet. Dann ist es die längere Leitung im Inneren der LED.
Wenn man die Leitung nicht sehen kann, so hilft nur probieren.

4. Einlöten des Winkelstiftes. Er dient als Prüfspitze (S).

5. Der Prüfstift wird über zwei Leitungen mit der Versorgungsspannung verbunden. Auf der Leiterplatte ist der +5V und Masse-Anschluß beschriftet.
Die +5V-Leitung verbindet man mit der +5V-Leitung der POW5V-Baugruppe und die Masse-Leitung mit der Masse-Leitung an der POW5V (auch manchmal mit 0V beschriftet).

6. Einsetzen der ICs. Dabei muß die Nase des ICs, also die Einkerbung, wie im Bestückungsplan angedeutet liegen.

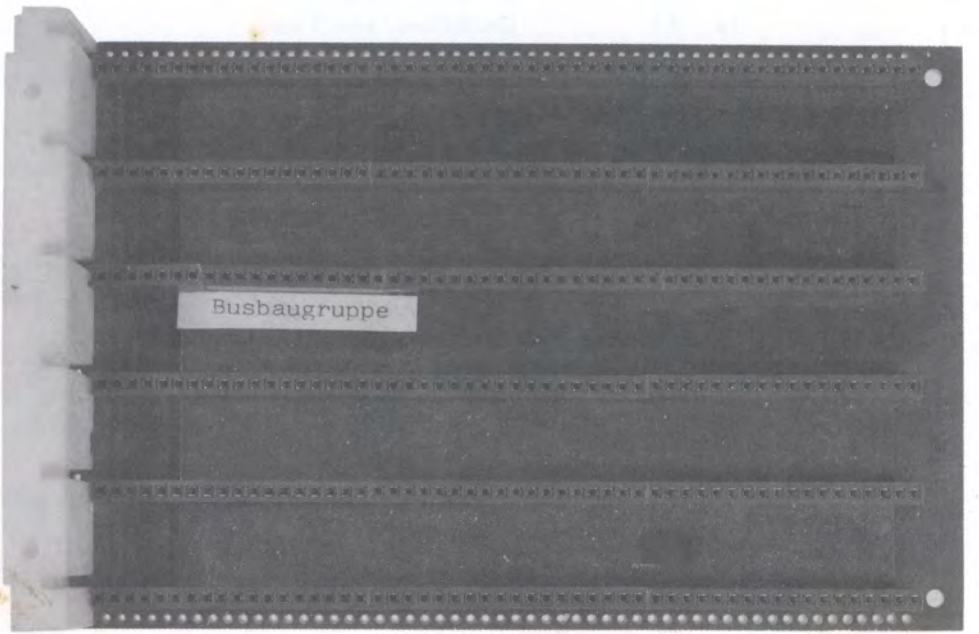
7. Einschalten der Versorgungsspannung. Die Leuchtdiode H (L2, rot) muß leuchten, sowie eine der beiden gelben Dioden leuchtet.

8. Wenn man mit der Prüfspitze 0V (Masse) antippt, so muß die Leuchtdiode L (L1, grün) leuchten.
Die gelben Leuchtdioden springen ggf. hin und her.

Damit ist der Prüfstift fertig.

A.3 BUS

Computerbaugruppen können Leitungen gemeinsam benutzen, man nennt das BUS. Zum Aufbau unseres Computers braucht man eine BUS-Karte, auf die man mehrere Baugruppen stecken kann, und die dadurch miteinander verbunden werden.



Die Signalbezeichnung der Leitungen ist nachfolgend abgedruckt:

-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				
A16	o	48				
A17	o	49				
A18	o	50				
A19	o	51				
0V	o	52				
0V	o	53				
Reserve	0	54				

Abb. A-7:
BUS
Signalbezeichnungen

Den Bus kann man auf einer Leiterplatte aufbauen.

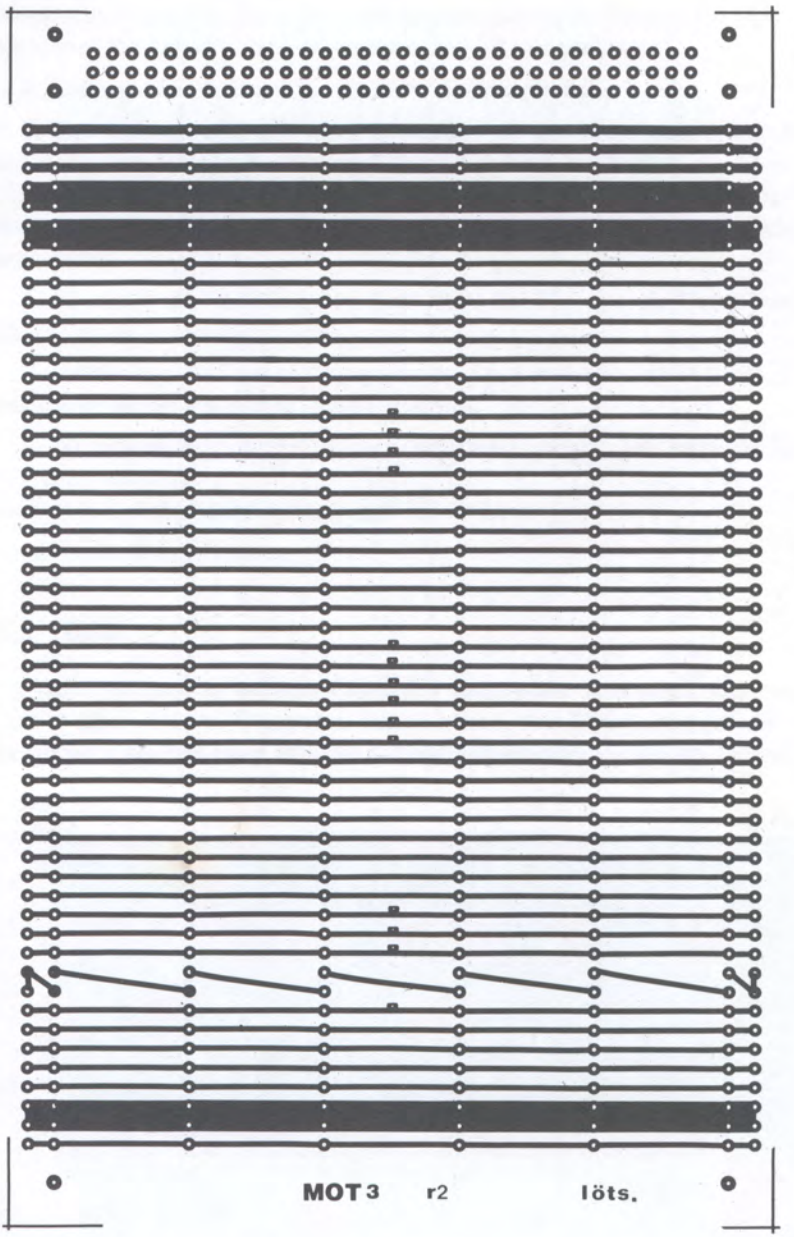


Abb. A-8: Lötseite der BUS-Leiterplatte

Die Bestückungsseite der BUS-Leiterplatte enthält eine große Fläche mit 0V, um die Störsicherheit zu verbessern.

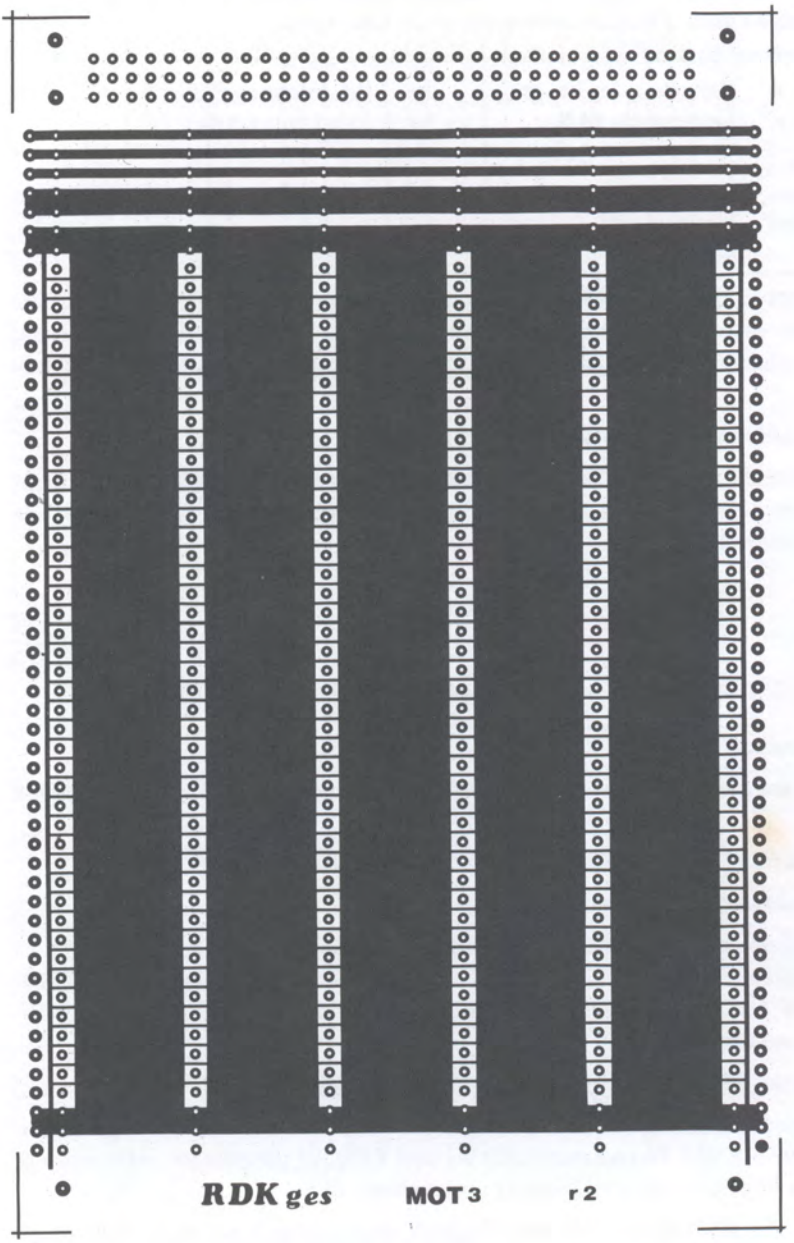


Abb. A-9: Bestückungsseite der BUS-Leiterplatte

Stückliste:

Für jeden Steckplatz benötigt man 3 Buchsenleisten (Typ AMP), für das kleine System genügen auch 2 Buchsenleisten mit je 18 Kontakten.
Auf die Busplatte passen 6 Steckreihen.

Ferner:	1x	Halterung der Fa. ges	1x	Bauanleitung BUS
	1x	Leiterplatte BUS 2	1x	Satz Gummifüße

Wenn man einen größeren Ausbau plant, kann man auch eine doppelt so breite Busplatte verwenden.

Zum Aufbau:
Die Buskarte mit Buchsen bestücken. Dabei darauf achten, daß die Buchsen fest auf der Karte aufliegen. Also lieber zuerst zwei Eckpunkte einer Leiste anlöten und dann auf die Oberseite schauen, ob die Leiste flach aufliegt. Dann erst die restlichen Anschlußbeine anlöten.

Achtung: Oberseite nicht mit Lötseite verwechseln.

Achtung: Manchmal passen die Buchsen wegen der engen Fertigungstoleranzen nicht genau aneinander; also zuerst ausprobieren und dann ggf. mit einer kleinen Feile die Buchsen passend machen.

A.4 SBCII

Die eigentliche Zentraleinheit ist auf dieser Baugruppe untergebracht.

Aufbau der Schaltung

Hier eine Kurzanleitung, eine ausführliche Anleitung ist zum Bausatz auf Anforderung erhältlich.

1. Einlöten aller IC-Fassungen.

2. Widerstände und Kondensatoren einlöten.

3. Die Baugruppe auf die BUS-Karte stecken. Dabei muß die Versorgungsspannung an den beiden breiteren Leiterbahnen mit den entsprechenden der Bus-Karte verbunden sein. Man sollte den Weg der Versorgungsspannung von der POW5V-Karte zur SBCII-Karte einmal sorgfältig verfolgen.

4. Einsetzen des ICs 74121.

5. Nun Spannung einschalten. Mit dem Prüfstift wird an Pin 1 des 74121 gemessen. Bei Betätigen des RESET-Tasters muß LED W1 und LED W2 umspringen, da immer ein kurzer Puls, das Flip-Flop auf dem Prüfstift umschaltet.

6. Einsetzen des ICs 74LS04. Mit dem Prüfstift wird an Pin 6 des noch nicht eingesteckten Z80 gemessen. Die LEDs W1 und W2 sowie L und H müssen alle leuchten.

7. Ausschalten.

Die nachfolgende Schaltung zeigt die Baugruppe im Detail:

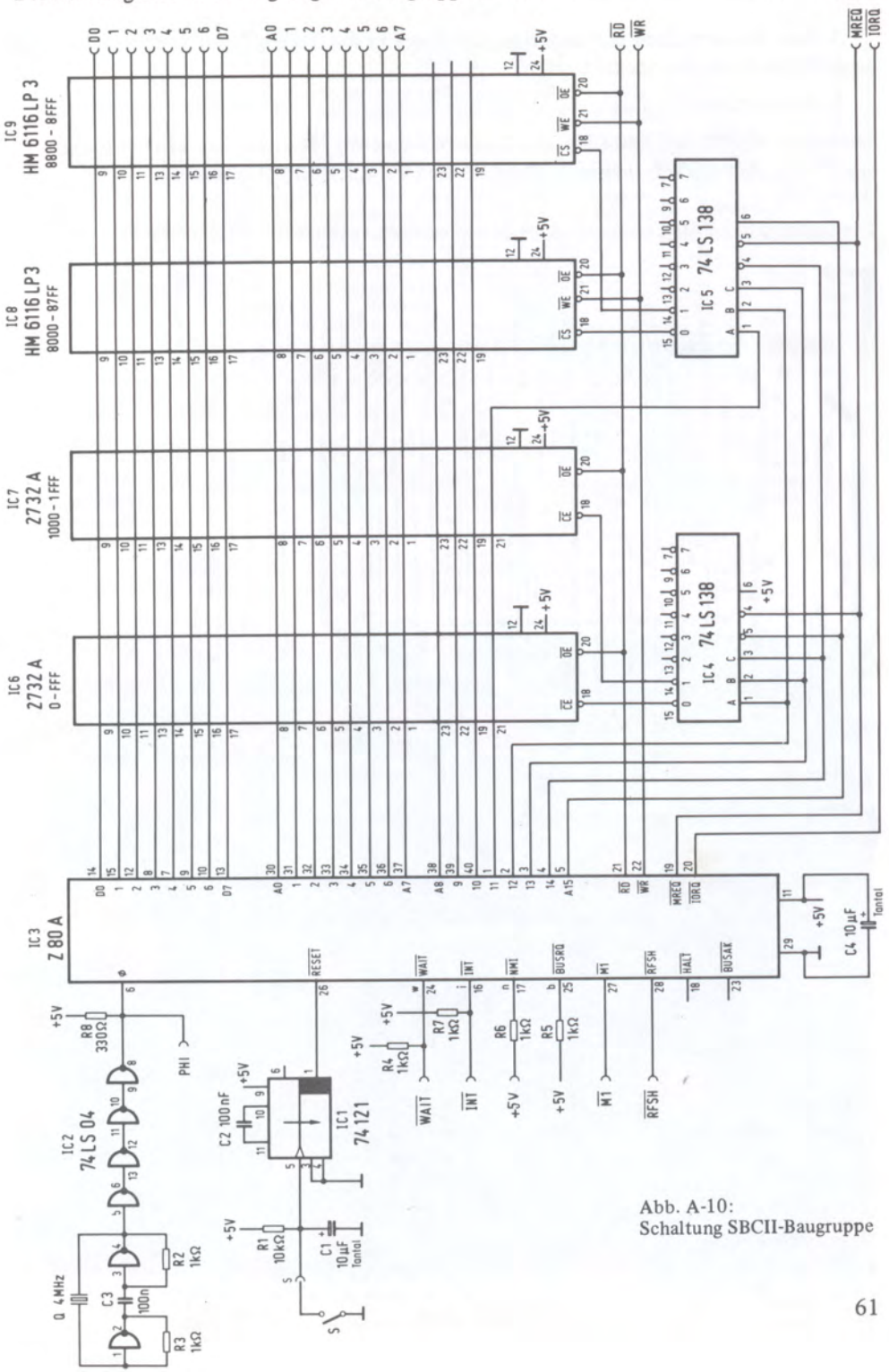


Abb. A-10:
Schaltung SBCII-Baugruppe

8. Nun alle restlichen ICs einsetzen, die Rams in die Sockel 2 und 3 (IC 8 und IC 9) und HEXMON in den Sockel 0 (IC 6).

9. Einschalten.

Messen an Pin 20 der Z80-CPU. Die LEDs W1 und W2 leuchten. Die LED H leuchtet, die LED L aber nur sehr schwach. Dann ist die CPU-Karte SBCII soweit ok.

Die gesamte Schaltung wird auf einer doppelseitigen Leiterplatte untergebracht.

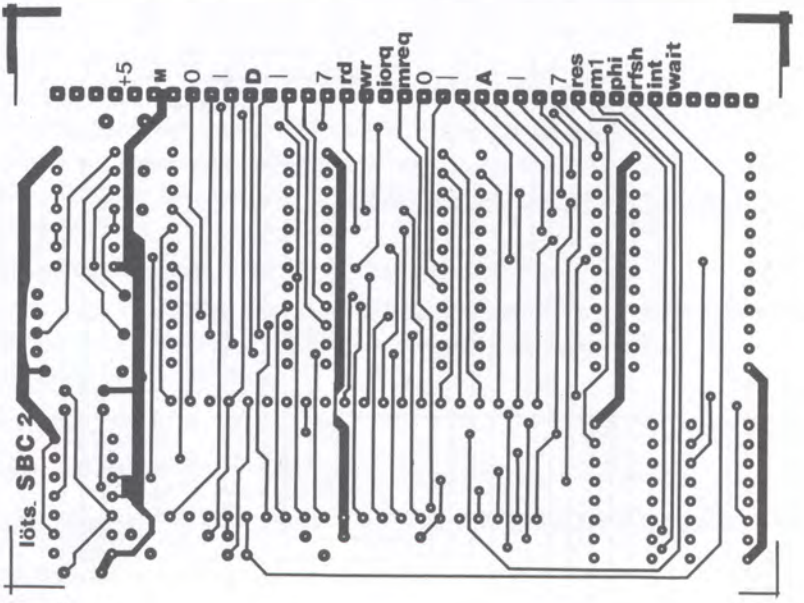


Abb. A-11: Lötseite der Baugruppe SBCII

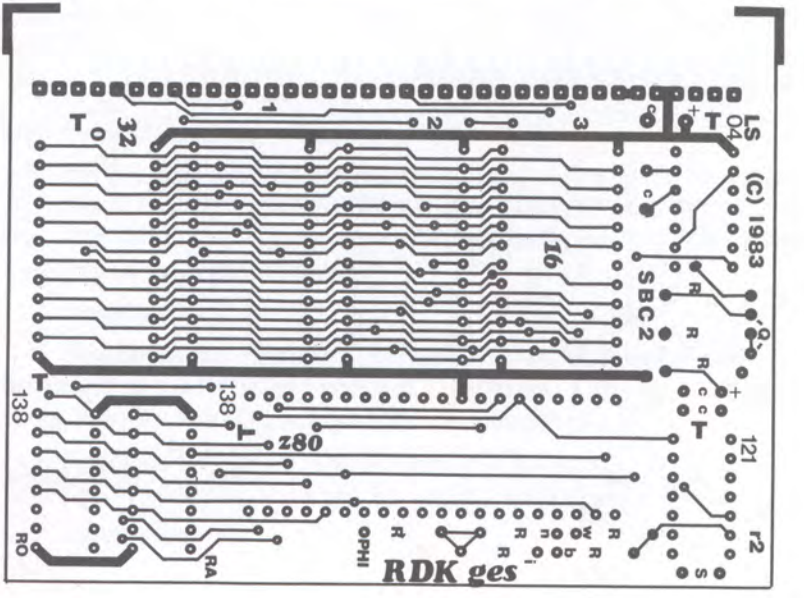


Abb. A-12: Bestückungsseite der Baugruppe SBCII

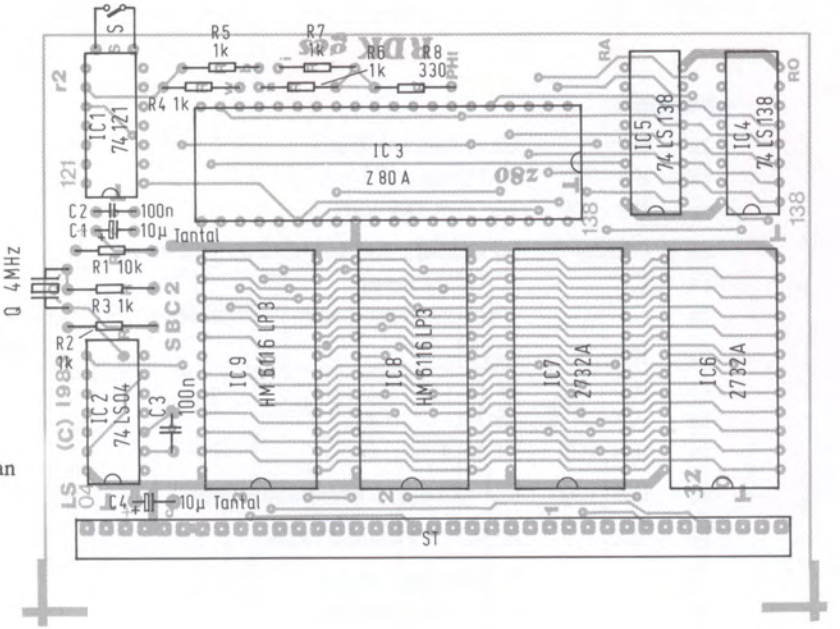
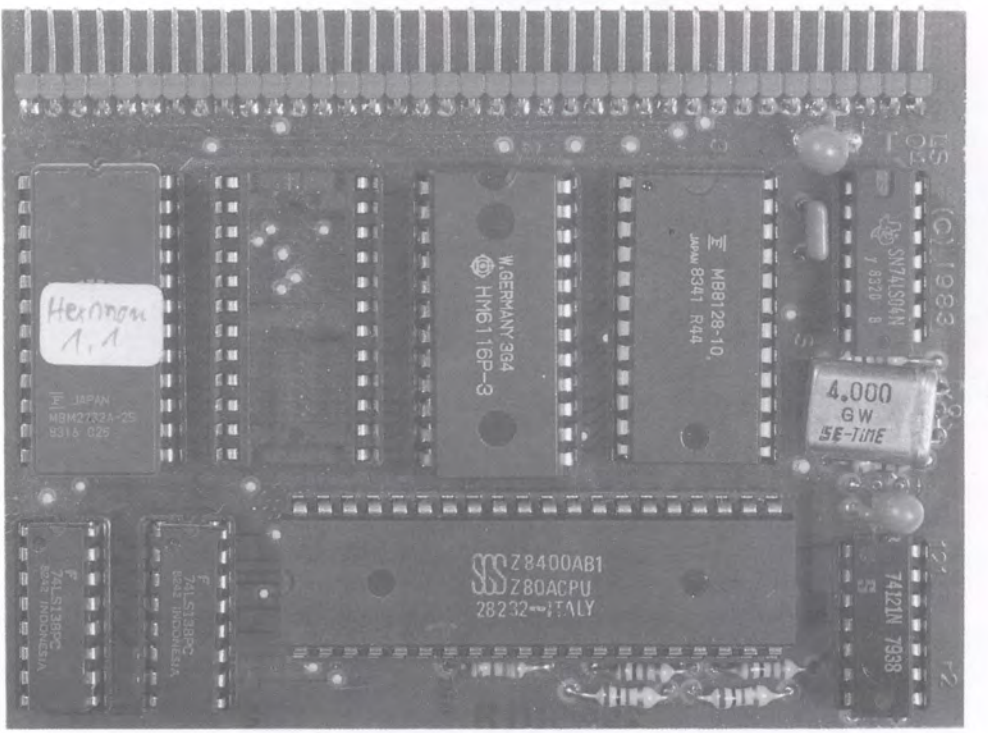


Abb. A-13:
Bestückungsplan
SBCII



Stückliste:

1x	R1	10 kOhm 1/4 Watt
1x	R8	330 Ohm 1/4 Watt
6x	R2 - R7	1 kOhm 1/4 Watt
2x	C1, C4	10 Mikrofarad (Tantal, Elko) 16V
1x	C2	100 Nanofarad
1x	C3	100 Nanofarad (oder 10 Nanofarad), Wert unkritisch.
1x	IC1	74121
1x	IC2	74 LS 04
1x	IC3	Z80 A
2x	IC4, IC5	74 LS 138
2x	IC8, IC9	HM 6116 LP3 (oder äquivalente)
1x	IC6	HEXIO EPROM 2732A
2x		IC-Fassung 14polig
2x		IC-Fassung 16polig
4x		IC-Fassung 24polig
1x		IC-Fassung 40polig
1x	Q	Quarz 4 MHz
1x		Steckerleiste 36polig (AMP)
1x	S	Taster für RESET
1x		Leiterplatte SBCII von ges
1x		Bauanleitung SBCII

A.5 IOE

IOE ist die Abkürzung für Input/Output Expander. Das bedeutet soviel wie Ein/Ausgabe-Erweiterung. Diese Baugruppe wird benötigt, um die Eingabetastatur und die Anzeige anzuschließen. Dabei kann man aber auch weitere IOE-Baugruppen im Computer verwenden, um den Computer z.B. für Steuerungen zu verwenden.

Die Schaltung ist auf einer zweiseitigen Leiterplatte untergebracht.

Nachfolgend die Schaltung:

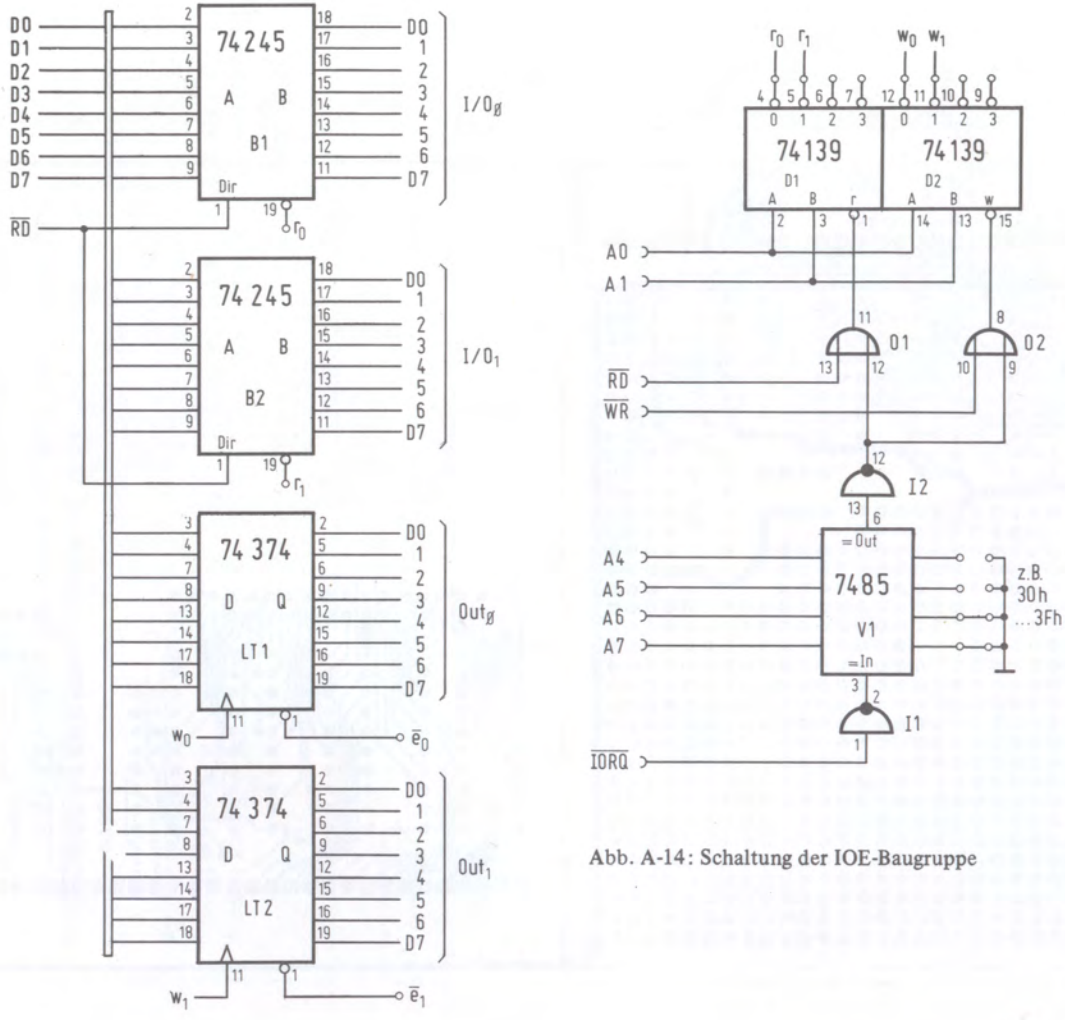


Abb. A-14: Schaltung der IOE-Baugruppe

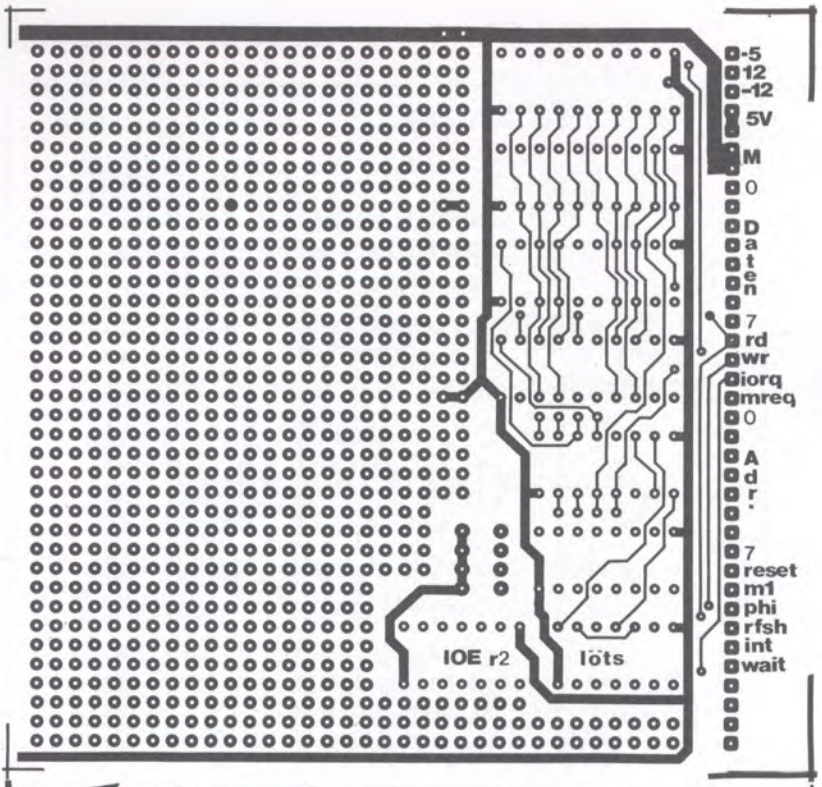


Abb. A-15:
Lötseite der
IOE-Leiterplatte

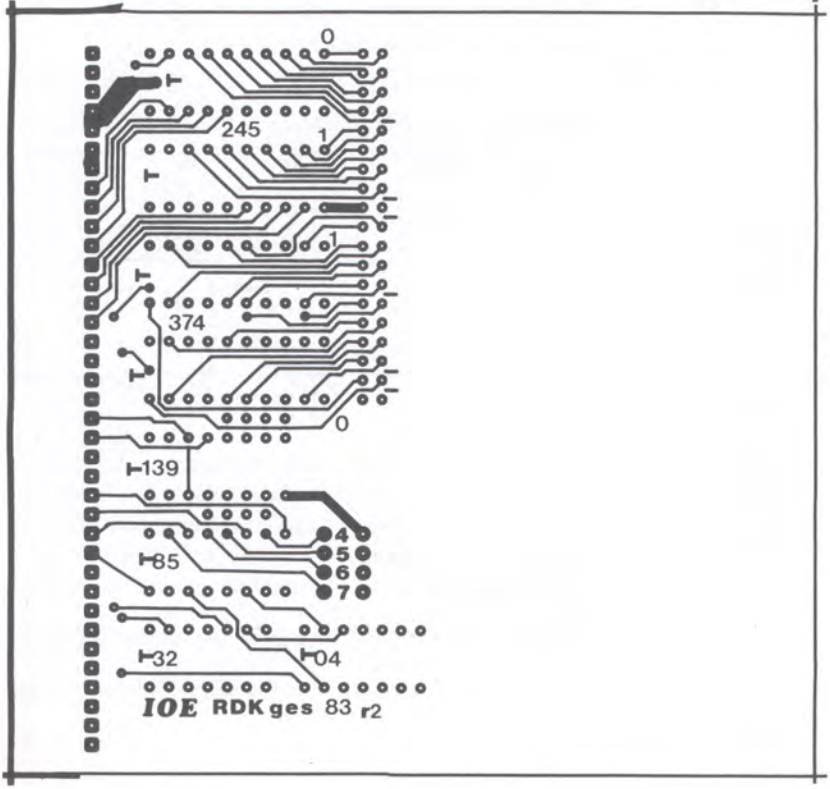


Abb. A-16:
Bestückungsseite
der IOE-
Leiterplatte

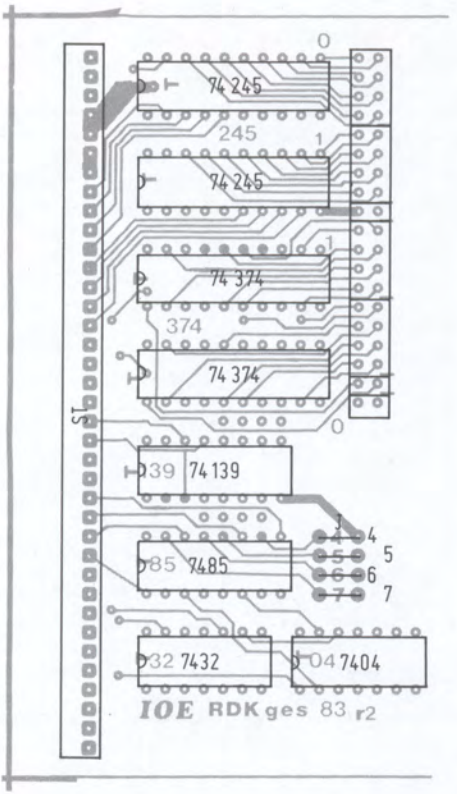
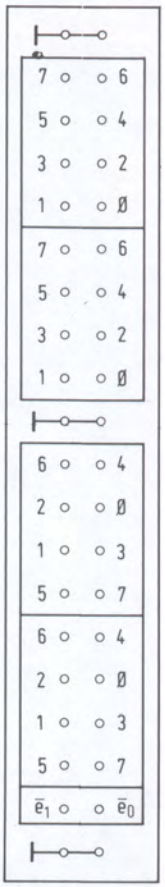


Abb. A-17: Bestückungsplan der IOE-Baugruppe



Stückliste:

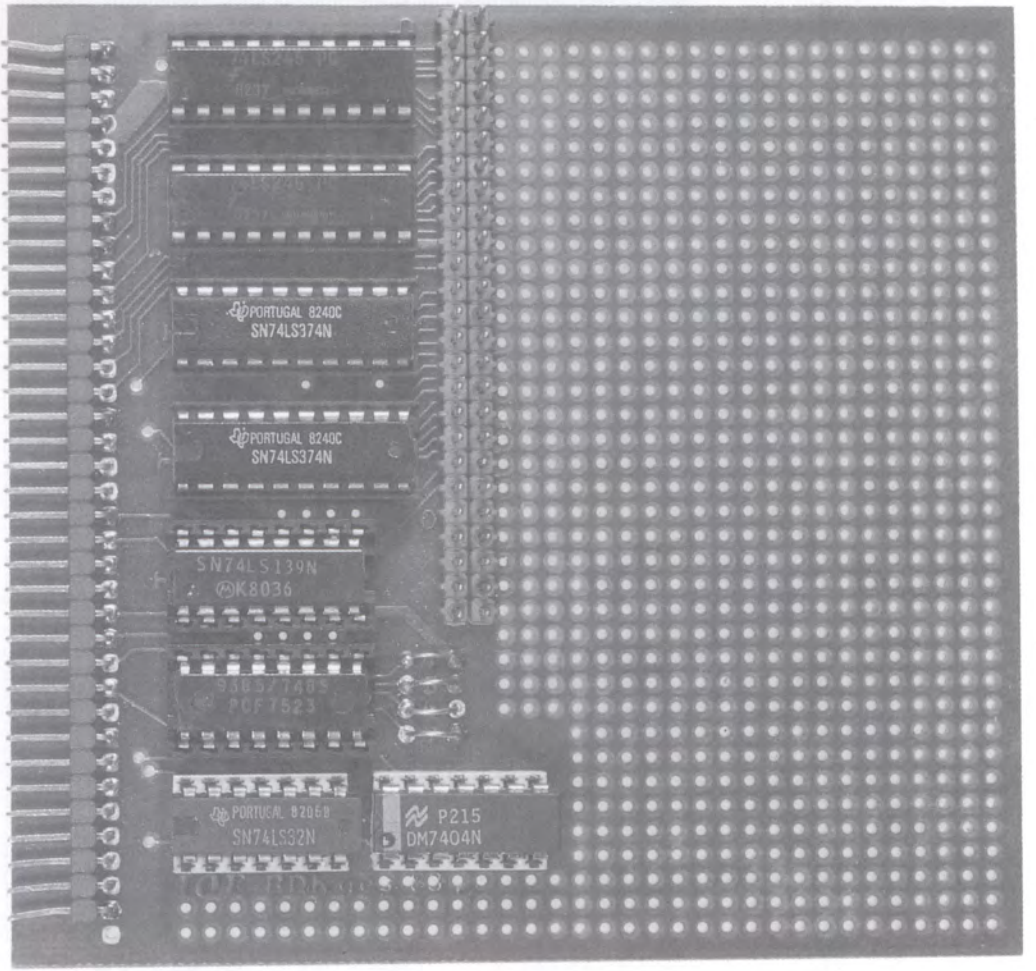
2x	B1, B2	74 LS 245	2x	IC-Fassung 14polig
2x	LT1, LT2	74 LS 374	2x	IC-Fassung 16polig
1x	I1, I2	74 LS 04	4x	IC-Fassung 20polig
1x	O1, O2	74 LS 32		
1x	D1, D2	74 LS 139		

1x C1 10 Mikrofarad 16V (nicht im Schaltplan, kann an der Eingabeleiste zwischen +5V und Masse frei verdrahtet werden).

1x Steckerleiste 36polig (AMP)

1x Leiterplatte IOE von ges

1x doppelreihige Stiftleiste 50polig für HEXIO-Verbindung,



Aufbau der Karte:

1. Alle Bauteile bestücken. (Zuerst die Fassungen, dann die Bauteile). Eine 50polige Stiftleiste zusätzlich neben den Treibern einlöten. (Siehe Inbetriebnahme-Kapitel in diesem Handbuch).

2. Die IOE-Karte auf den BUS stecken, wie auch POW5V und SBCII.

3. Einschalten.

4. An Pin 19 des 74LS245 (IO/0 am Rande der Karte) messen. Die LEDs W1 und W2 sowie H leuchten, die LED L ist praktisch dunkel.

5. An Pin 11 der beiden 74LS374 sieht die Messung genauso aus.

Damit ist die IOE-Karte grob vorgeprüft.

6. Nicht vergessen, die +5V-Leitung an die Stiftleiste anzulöten. (Siehe 1.2 Kapitel Inbetriebnahme).

A.6 HEXIO

HEXIO ist die zentrale Baugruppe dieses Buches. Dort befindet sich die Eingabetastatur und auch die Anzeige.

1. Genau darauf achten: *Es wird nur auf der Lötseite gelötet.*

2. Alle 47-Ohm-Widerstände einlöten. Zur Bauteile-Lage siehe Bestückungsplan auf den folgenden Seiten.

3. Alle 1k-Ohm-Widerstände einlöten.

4. Alle Dioden 1N4148 (oder ähnliche Silizium-Dioden) einlöten. Bei den Dioden auf die richtige Polung achten. Die Kathode ist durch einen Querring auf den Dioden gekennzeichnet. Bei mehreren Ringen ist dies der dickste Ring.

5. Alle Transistoren einlöten. Dabei ggf. Unterlegscheiben verwenden, also die Transistoren nicht direkt flach aufliegend einlöten.

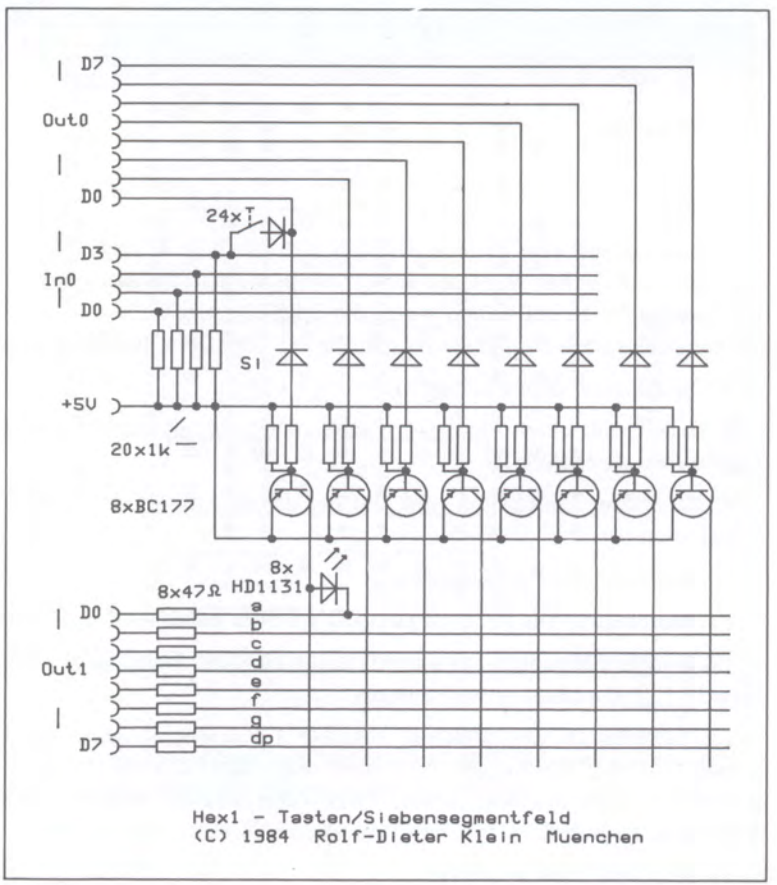
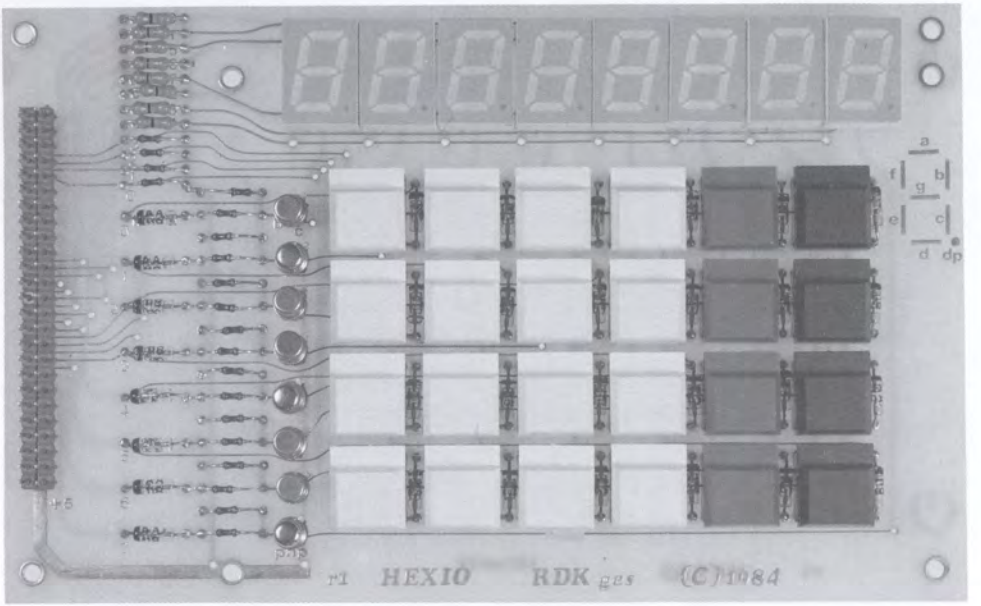


Abb. A-18: Schaltplan HEXIO



HEXIO-Baugruppe

6. Nun erst mal eine Anzeige einlöten, z.B. die linke Anzeige. Möglichst nur sehr kurz löten und auf flaches Aufliegen achten, wenn möglich Sockel verwenden. Der Punkt bei der Anzeige ist rechts unten, wenn die HEXIO-Karte so vor einem liegt, daß die Bestückungsseite oben ist und die Anschlüsse zur 50poligen Stiftleiste links liegen.

7. Die 50polige doppelreihige Stiftleiste einlöten.

8. HEXIO mit einer 50poligen Flachbandleitung mit der IOE-Karte verbinden (siehe Kapitel Inbetriebnahme).

9. Nach dem Einschalten muß auf der linken Anzeige ein Buchstabe H erscheinen. Ausschalten.

10. Restliche Anzeigen einlöten.

11. Einschalten. Die Meldung HALLO 1.1 muß lesbar sein. Ausschalten.

12. Nun eine Minidigit-Taste rechts unten einlöten. Dabei steht auf der Unterseite der Taste 0, 1, 2. 0 kommt unten zu liegen.

13. Einschalten. Die Meldung HALLO 1.1 erscheint, dann die Taste drücken, es müssen ------- auf der Anzeige erscheinen. Falls dies nicht der Fall ist, die Lage der Taste überprüfen, sowie die Orientierung. Dazu kann man ein Meßgerät verwenden, oder den Prüfstift. Ausschalten.

14. Restliche Tasten einlöten.

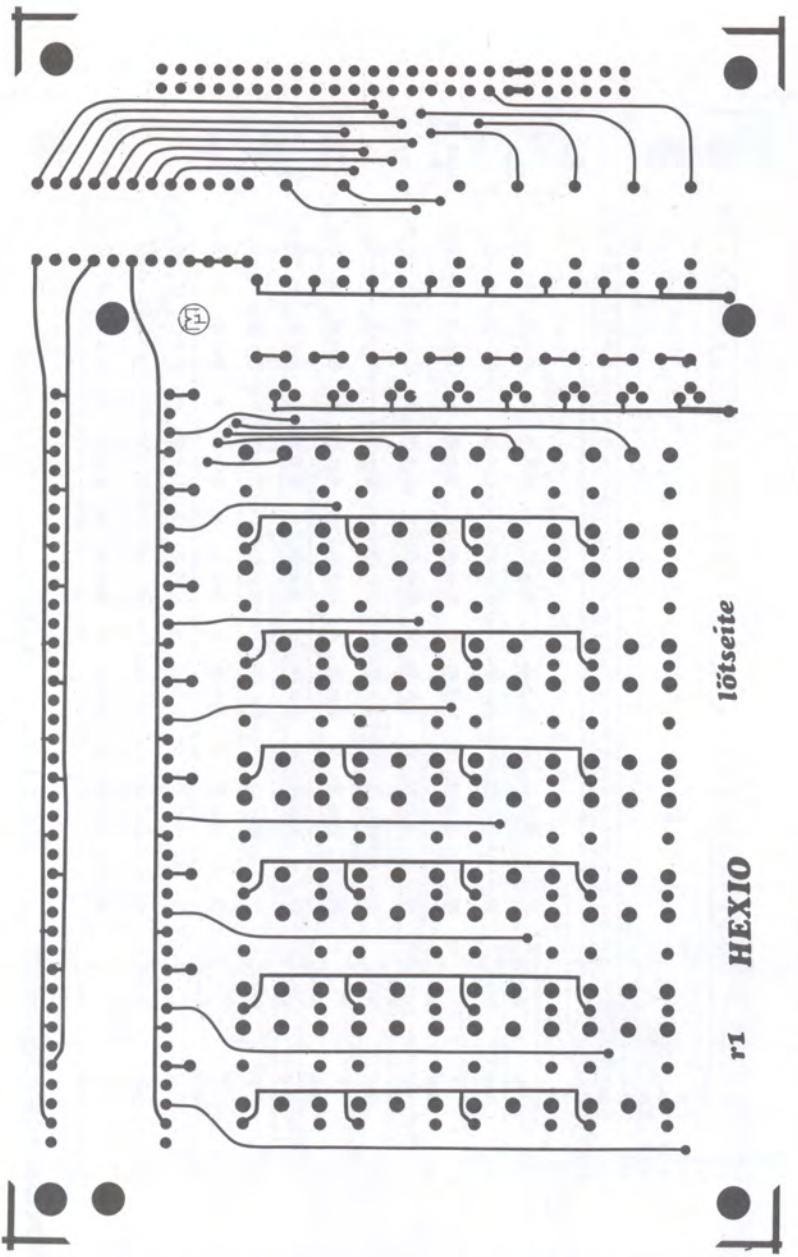


Abb. A-19: Lötseite der HEXIO-Leiterplatte

15. Ausschneidebogen aus dem Anhang verwenden und eine Tastenschablone anfertigen. Die Tastenschablone kann man über die einzelnen Tasten legen.

16. Wer will, besorgt sich ein rotes Plexiglas-Filter und legt es über die Anzeigen; sie werden dadurch besser ablesbar.

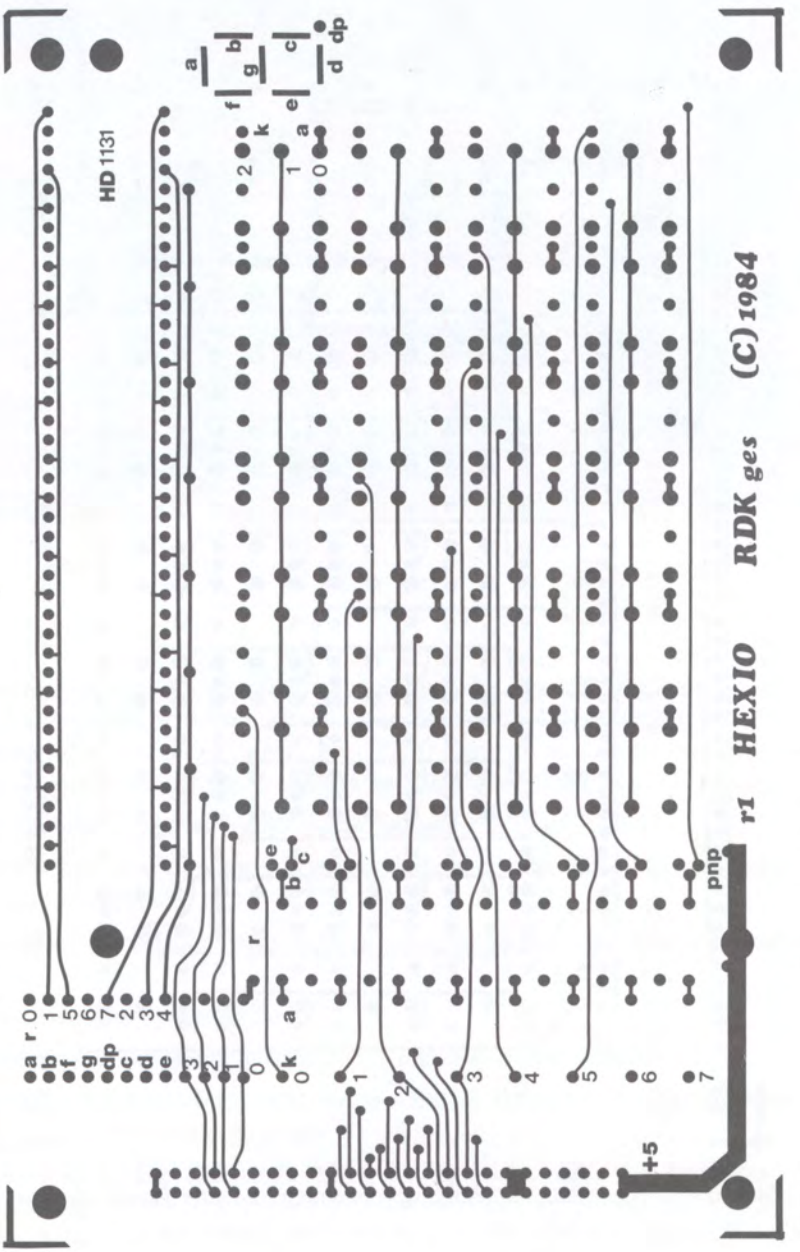


Abb. A-20: Bestückungsseite der HEXIO-Leiterplatte

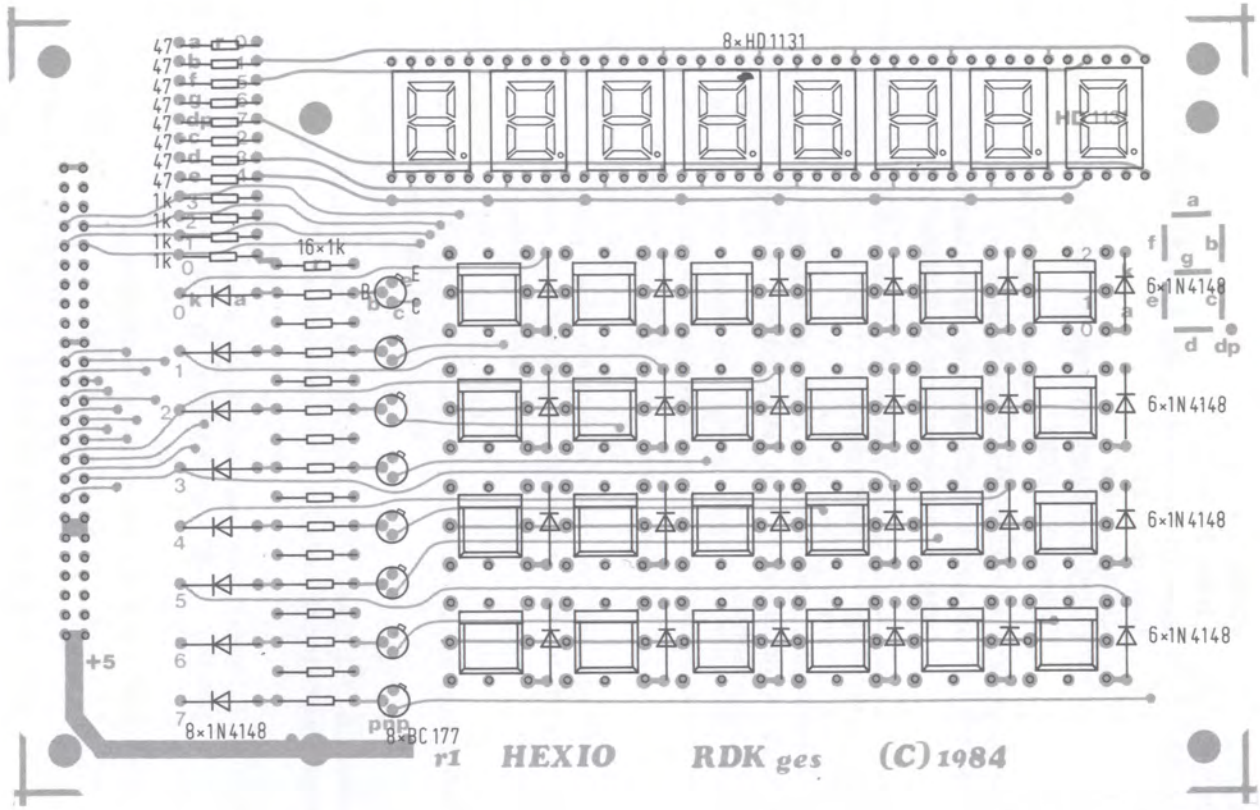


Abb. A-21: Bestückungsplan

Stückliste der HEXIO-Baugruppe:

8x	47 Ohm 1/4 Watt
20x	1 kOhm 1/4 Watt
32x	Diode 1N 4148 Si oder (oder AA 118 Ge etc.)
8x	Transistoren BC 177
24x	Minidigit-Tasten
8x	Leuchtanzeigen HD 1131
1x	Leiterplatte HEXIO
1x	doppelreihige Stiftleiste 50polig
1x	50-polige Flachbandleitung mit Buchsen an beiden Enden, passen zu den verwendeten Stiftleisten.

Weitere Schaltungsbeschreibungen zum NDR-Klein-Computer (wie die Baugruppe CAS und PROMMER) findet man im Buch „Mikrocomputer selbstgebaut und programmiert" siehe Literaturverzeichnis.

B Kurzbefehlsliste

Hier werden nochmals kurz die Bedeutungen der einzelnen Tasten beschrieben.

0 mve Abkürzung für MOVE (engl.: bewege).
Damit kann man einen Speicherbereich verschieben. Es wird geprüft, ob der Bereich mit dem Quellbereich überlappt, und jeweils so transportiert, daß die Quelle vollständig übertragen wird.

1 brk Abkürzung für BREAK (engl.: unterbrechen)
Es lassen sich bis zu drei Unterbrechungsadressen angeben, bei denen die Programmausführung gestoppt wird. Die Adresse 0 ist ein Sonderfall und bedeutet, daß der Breakpoint nicht gesetzt ist.

2 ful Abkürzung für FÜLLEN.
Damit kann man einen Speicherbereich mit einem konstanten Wert auffüllen. Es wird geprüft, ob der Wert auch vom Speicher angenommen wurde, und im Fehlerfall eine Adresse ausgeben.

3 vgl Abkürzung für VERGLEICH.
Zwei Speicherbereiche werden miteinander verglichen. Wenn die Dateninhalte nicht übereinstimmen, wird die Adresse ausgegeben, bei der die erste Nicht-Übereinstimmung gefunden wurde.

opt Abkürzung für OPTION (engl.: Zusatz)
Dient dem Umschalten, z.B. von Zahlbereichen oder von Darstellungsarten auf der Anzeige. Ist kein selbständiger Befehl.

Abkürzung für MINUS
Damit kann man innerhalb von Befehlen zurückgehen, z.B. die vorherige Adresse anwählen, oder das vorherige Register. Ist kein selbstständiger Befehl.

4 reg Abkürzung für REGISTER
Die Register des Z80 lassen sich ansehen und modifizieren. Wenn man im STEP-Befehl ist, kann man auch Register ansehen und modifizieren.

5 prf Abkürzung für PRÜFEN
Daten werden vom Kassettenrekorder gelesen und mit dem aktuellen Speicherinhalt verglichen.

6 spe Abkürzung für SPEICHERN
Daten werden vom Speicher auf den Kassettenrekorder gespeichert.

7 lad Abkürzung für LADEN
Daten werden vom Kassettenrekorder geladen, und am Schluß wird die Prüf-

summe kontrolliert, so daß Aufzeichnungsfehler erkannt werden. Die gelesenen Daten werden auch in binärer Form auf den Punkten der Anzeige ausgegeben.

step — Abkürzung für STEP (engl.: schreiten)
Damit kann man ein Z80-Programm im Einzelschritt-Verfahren durchlaufen. Nach jedem Schritt kann man Registerinhalte betrachten, oder Speicherzellen. Ferner kann man auf einen Dauerbetrieb schalten und dabei sich verändernde Register oder Speicherzellen beobachten.

\+ — Abkürzung für PLUS
Wird in Befehlen verwendet, z.B. zum Anwählen der nächsten Speicherzelle, oder nächstes Register.

8 ios — Abkürzung für IO-SETZEN
Damit kann man Z80-IO-Ports mit einem Wert belegen. So kann man eigene Zusatzschaltungen kontrollieren.

9 iol — Abkürzung für IO-LESEN
Dadurch ist es möglich, den Wert an Z80-IO-Ports einzulesen. Es kann dabei mit „opt" auf eine duale Zahlenschreibweise umgeschaltet werden, und mit „start" ist ein Dauereinlesen möglich.

A prp — Abkürzung für EPROM-PROGRAMMIERUNG
Damit kann man einen Speicherbereich in ein EPROM übertragen.

B prl — Abkürzung für EPROM-LESEN
Damit kann man einen EPROM-Inhalt in einen Speicherbereich übertragen.

start — Abkürzung für START
Ein Programm wird gestartet, wobei alle Breakpoints, die gültig sind, in das Programm übertragen werden. Das Programm endet dann dort mit der Meldung BRE xxxx, wobei xxxx die Adresse des Programmzählers ist.

speich — Abkürzung für SPEICHER
Der Speicherinhalt kann angesehen und modifiziert werden. Mit „opt" kann man auch eine Z80-Befehlsausgabe erreichen, so daß immer die aktuelle Anzahl von belegten Bytes angezeigt wird. „CR", „+" und „–" können verwendet werden.

C prm — Abkürzung für PROMMER
Auf der Anzeige erscheint ein Zahlenwert. Dieser Zahlenwert gibt eine Zeit in Millisekunden an. Damit wird die Zeit des Monoflops auf der PROMMER-Karte gemessen. So kann man das Monoflop auf 50ms abgleichen. Der Befehl ist nur möglich, wenn die PROMMER-Karte auf den BUS gesteckt ist.

D per — Abkürzung für PERIODENDAUER
Damit kann man z.B. die CAS-Karte abgleichen. An Bit 7 des IO-Ports auf der IOE-Karte wird eine Meßleitung angeschlossen. Die Periodendauer der dort anliegenden Frequenz wird in μs ausgegeben. Der Taktgeber der CAS-Karte kann damit auf 833 μs abgeglichen werden.

E pul Abkürzung für PULSDAUER
Damit wird die 0-Signal-Pulsdauer in μs gemessen. Das Monoflop auf der CAS-Karte kann damit abgeglichen werden. Es muß auf ca. 624 μs eingestellt werden.

F umw Abkürzung für UMWANDLUNG
Sedezimal nach Dezimal und umgekehrt. Damit kann man die beiden Zahlensysteme ineinander umrechnen.

BEF Abkürzung für BEFEHL
Mit dieser Taste kann man von den unterschiedlichen Eingabeformen aus anderen Befehlen wieder zur Befehlseingabe gelangen. Dies gelingt natürlich nicht, wenn z.B. ein Anwenderprogramm gestartet wird, das sich „aufhängt".

CR Abkürzung für CARRIAGE RETURN (engl.: Wagenrücklauf)
Damit werden Zahleneingaben beendet.

C Zeichendarstellung auf der Anzeige

C.1 Codierungstabelle

Da wir nur sieben Segmente pro Anzeige haben, ist es nicht ganz leicht, beliebige Buchstaben und Zeichen auf der Anzeige darzustellen. Damit dies aber möglichst einheitlich geschieht, sein hier eine Liste mit Vorschlägen abgedruckt. Die Codierung ist jeweils daneben abgedruckt.

```
      0        !
    *****      !
5 *  6  * 1    !
    *****      !  7   6   5   4   3   2   1   0
4 *     * 2    !  dp  g   f   e   d   c   b   a
    *****      !
      3   * 7  !        0=Segment leuchtet
```

```
*****          0
*   *
*   *      11000000        C0
*   *
*****

    *          1
    *
    *      11111001        F9
    *
    *

*****          2
    *
*****      10100100        A4
*
*****

*****          3
    *
*****      10110000        B0
    *
*****
```

4 10011001 99

5 10010010 92

6 10000010 82

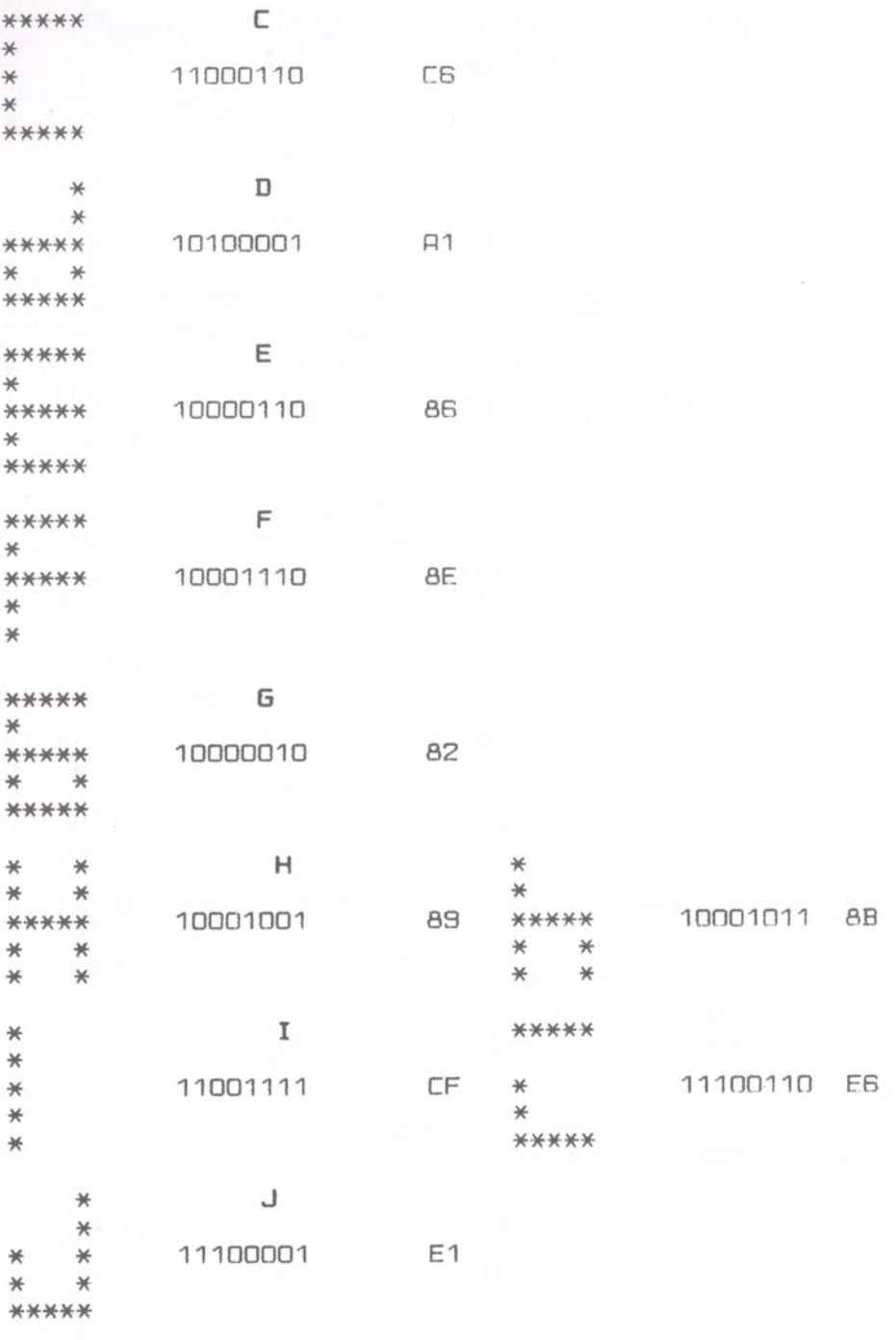
7 11111000 F8

8 10000000 80

9 10011000 98

A 10001000 88

B 10000011 83


C
11000110
C6
D
10100001
A1
E
10000110
86
F
10001110
8E
G
10000010
82
H
10001001
89
I
11001111
CF
J
11100001
E1
10001011
8B
11100110
E6

K 10000101 85

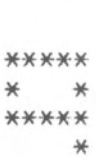
L 11000111 C7

M 10101010 AA

N 10101011 AB

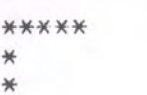
O 10100011 A3

P 10001100 8C



Q 10011000 98

R



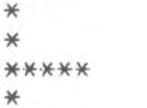
10101111 AF

S

10010010 92

10010011 93

T



10000111 87

U

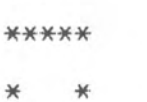
11000001 C1

11100011 E3

V

10000001 81

W



11100010 E2

X

10001111 8F

10001001 89

Y

10010001 91

```
*****          Z

*****      10110110        B6

*****

                 ( (

*****      10100111        A7
*
*****

                 ) )

*****      10110011        B3
    *
*****


    *            +
    *
*****      10111001        B9
    *
    *


                 -

*****      10111111        BF


                 ,

           01110111        77
***** *

                 .

           01111111        7F
      *
```

C.2 Testprogramm für Segmentcodierung

Mit dem nachfolgenden Programm können auch eigene Codierungen überprüft werden, um so unterschiedliche Segmentausgaben zu erreichen.

```
Start:
      8100 CD    CALL CLEAR        ; löschen der Anzeige,
      8101 33                      ; ANZFELD wird mit
      8102 00                      ; FF gefuellt
ANFANG:
      8103 DD    LD IX,8006H       ; Die Adresse der Anzeige
      8104 21                      ; an zweitletzter
      8105 06                      ; Stelle
      8106 80
      8107 CD    CALL GETC         ; Eingabe von Benutzer
      8108 24                      ; sedezimale Zahl holen
      8109 00                      ; kommt in C an.
      810A 79    LD A,C            ; Wert nach A bringen
      810B 32    LD (8000),A       ; im Anzeigefeld ablegen
      810C 00                      ; an erster Stelle.
      810D 80
      810E 18    JR ANFANG         ; und zurückspringen
      810F F3                      ; 8103-8110
```

Abb. C.1: Programm „Segmentcode testen"

Nach dem Programmstart wird zunächst das Anzeigefeld (ANZFELD) mit dem Unterprogramm CLEAR gelöscht. ANZFELD ist ein Speicherbereich, der von HEXMON verwendet wird, um sich die einzelnen Segmentcodes für die Ausgabe zu merken. Dann wird mit dem Unterprogramm GETC eine sedezimale Zahl eingelesen, die zwei Stellen umfassen kann. Der eingelesene Wert wird als Segmentcode im Anzeigefeld gespeichert und dann beim nächsten Aufruf von GETC zusätzlich mit angezeigt, da GETC nicht nur eine Eingabefunktion besitzt, sondern auch gleichzeitig die Ausgabe auf der Anzeige übernimmt. Das Programm wird auf Adresse 8100H gestartet. H steht für HEX, damit ist gemeint, daß 8100 eine HEX-Zahl ist, also eine sedezimale Zahl.

D Listings der Beispielprogramme

In diesem Anhang-Teil findet man die Listings der Beispielprogramme, die im Einführungsteil nur als HEX-Dump vorliegen.

D.1 Programm „Verknüpfungen"

Das Ergebnis der Verknüpfung wird auf den Segmenten a der ersten beiden Anzeigen ausgegeben. Zur Eingabe dienen die Tasten 0 und 1.

```
8100 3E    LD A,0FEH   ; Anwahl der ersten Tastenspalte
8101 FE
8102 D3    OUT (0),A   ; Ausgabe auf den enstprechenden
8103 00                ; Ansteuerport
8104 DB    IN A,(0)    ; Tastenreihe lesen
8105 00
8106 2F    CPL         ; negieren, da 0=gedrückt
8107 47    LD B,A      ; und den Wert merken
8108 3E    LD A,0FDH   ; zweite Tastenspalte anwählen
8109 FD
810A D3    OUT (0),A   ; und ansteuern
810B 00
810C DB    IN A,(0)    ; Tastenreihe lesen
810D 00
810E 2F    CPL         ; negieren, da 0=gedrückt
810F A0    AND B       ; VERKNUEPFUNG BILDEN B o A -> A
8110 2F    CPL         ; negieren, da 0=Segment an
8111 F6    OR 11111110B ; nur ein Segment verwenden
8112 FE
8113 D3    OUT (1),A   ; Segmente anwählen
8114 01
8115 18    JR 8100     ; und alles wiederholen
8116 E9                ; daher Sprung zum Start
```

Abb. D-1: Listing „Verknüpfungen"

Wenn man auf Adresse 810F einen anderen Befehl legt, so kann man auch andere Verknüpfungen realisieren.

Zum Beispiel: B0 bedeutet OR B
2F bedeutet CPL (complement)
A8 bedeutet XOR B (exclusive or)

D.2 Programm "Flip-Flop"

```
8100 3E    LD A,7FH     ; Punkt als Led verwenden
8101 7F
8102 D3    OUT (1),A    ; Auf Segment-Port ausgeben
8103 01
8104 3E    LD A,0FEH    ; erste Reihe anwählen
8105 FE
8106 D3    OUT (0),A    ; Ziel ist Ansteuer-Port
8107 00
8108 DB    IN A,(0)     ; Tasten-Port einlesen
8109 00
810A E6    AND 1        ; TASTE 0 interessant
810B 01
810C 20    JR NZ,8104   ; wenn nicht gedrückt, neu
810D F6                 ; abfragen
810E 3E    LD A,0FDH    ; sonst zweite Reihe abfragen
810F FD
8110 D3    OUT (0),A    ; dazu Ansteuerport verwenden
8111 00
8112 DB    IN A,(0)     ; nun Tastenport einlesen
8113 00
8114 E6    AND 1        ; TASTE 1
8115 01
8116 20    JR NZ,810E   ; wiederholen, wenn nicht gedrückt
8117 F6
8118 18    JR -8104     ; sonst zurück zur ersten Reihe
8119 EA
```

Abb. D-2: Listing „Flip-Flop"

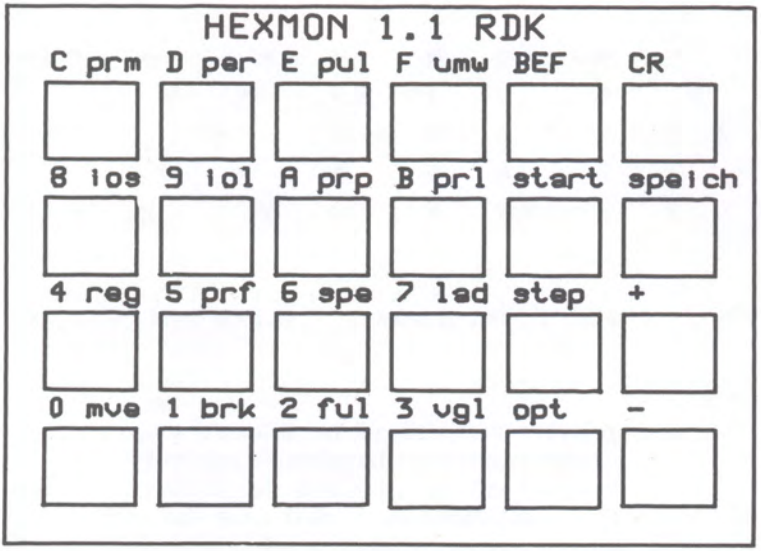
In diesem Programm werden verschiedene Schleifen durchlaufen. Die Struktur sieht wie folgt aus:

```
Punkt ausgeben
Wiederhole
  Wiederhole
    Anwahl Spalte 0
    Lese Taste 0
  Bis Taste gedrückt
  Wiederhole
    Anwahl Spalte 1
    Lese Taste 1
  Bis Taste gedrückt
Unbegrenzt
```

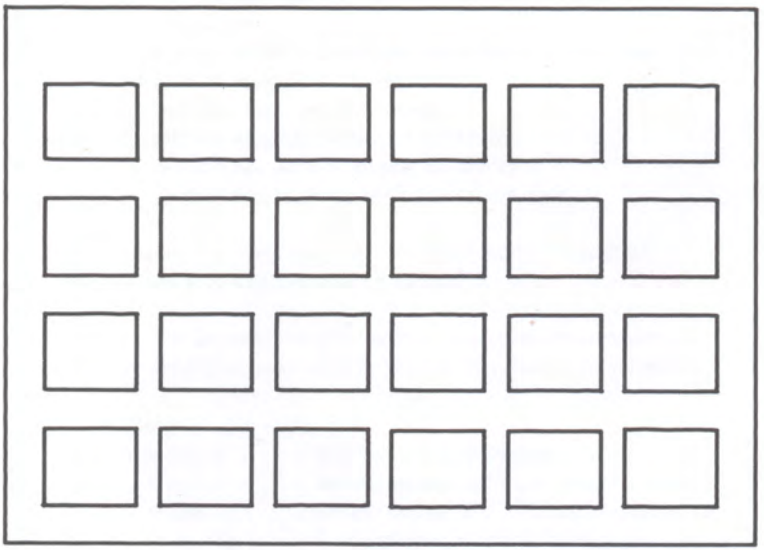
Die Ausgabe des Punktes auf der entsprechenden LED erfolgt durch die Auswahl der Spalte. Dort wird zum einen die Spalte zum Lesen der Taste eingestellt, zum anderen die Spalte zur Ausgabe der Segmente.

E Ausschneide-Tafel

E.1 HEXMON Belegung



E.2 Leerfeld



F HEXMON-Listing

```
HEXMONITOR FUER DIE SBCII-KARTE + HEXIO REV 1.1a        MACRO-80 3.43   27-Jul-81       PAGE 1

                                .z80
                                title HEXMONITOR FUER DIE SBCII-KARTE + HEXIO REV 1.1a
                                ;***********************************************
                                ;* HEXMONITOR   Version 1.1  (C) 1984 Muenchen *
                                ;* Rolf-Dieter Klein     rev 840520 1.1a       *
                                ;***********************************************

                                ; Damit ist es moeglich, mit der SBCII + IOE + HEXIO
                                ; Programme einzugeben, zu veraendern, zu testen
                                ; und mit CAS kann man sie auch auf einer Kassette
                                ; abspeichern.
                                ; Mit der PROMMER-Karte kann man erstellte Programme
                                ; auch in einem EPROM abspeichern
                                ; Fuer den Test von Programmen
                                ; gibt es eine Einzelschritt-Funktion,
                                ; mit der man auch Programme in EPROMS
                                ; Schritt fuer Schritt ablaufen lassen kann.

0000                             aseg   ; Anweisung fuer den Uebersetzer (Assembler)
                                 org 0  ; Startadresse im ROM ist 0

                                ; Die Sprungtabelle erlaubt es, einzelne interessante
                                ; Unterprogramme auch in eigenen Programmen zu verwenden

0000                            basis:
0000    C3 0DFF                  jp start       ; Start des Hauptprogramms
0003    C3 0287                  jp ri          ; Unterprogramm LESEN von Kassette nach A
0006    C3 0290                  jp poo         ; Unterprogramm SCHREIBEN auf Kassette von C
0009    C3 0069                  jp anzeige     ; fuer Ausgabe auf der Siebensegmentanzeige
```

```
000C  C3 00BB           jp holetaste  ; wie oben, jedoch mit Einlesen der Tastatur
000F  C3 00CE           jp tonum      ; von Tastencode umrechnen in 0..F
0012  C3 0106           jp toseg      ; von 0..F umrechnen in Segmentcode
0015  C3 02A6           jp print      ; Textausgabe auf Anzeige
0018  C3 012A           jp prtac      ; Akku ausgeben HEX (sedezimal), 0..FF
001B  C3 0145           jp prthl      ; HL ausgeben   HEX 0..FFFF
001E  C3 014E           jp prtbin     ; A Binaer-Ausgabe 0..11111111
0021  C3 0162           jp prtdez     ; HL Dezimal-Ausgabe  -32768..32767
0024  C3 01ED           jp getc       ; ein Byte nach C holen von der Tastatur (HEX)
0027  C3 0218           jp gethl      ; zwei Bytes nach HL holen (HEX)
002A  C3 01C2           jp getdez     ; Dezimalzahl einlesen mit Vorzeichen
002D  C3 053B           jp step       ; Einen Schritt ausfuehren, HL ist Anfangsadresse
                                      ; RST 6, liegt auf Adresse 30h
0030  C3 0645           jp break      ; Haltepunkt  fuer Start
0033  C3 02B3           jp clear      ; Loeschen des Anzeigefeldes ANZFELD
0036  00               nop           ; auffuellen, damit INT-Adresse
0037  00               nop           ; auf 38h zu liegen kommt.
0038  C3 800B           jp intloc     ; INT auf Adresse 38h, springt in RAM-Teil
003B  C3 02BC           jp length     ; HL=Adresse, B=Laenge des Z80-Befehls
003E  C3 0D9A           jp lastmem    ; Speichertest, HL ist dann letzte RAM-Zelle
0041  C3 0D9D           jp suchl      ; Speichertest, HL=Start der Suche
0044  C3 029A           jp getri      ; RI lesen mit zus. LED-Ausgabe
                        org 66h       ; NMI auf Adresse 66h, springt in RAM-Teil
0066  C3 8008           jp nmiloc


                        ; ---------- Unterprogrammsammlung fuer ----------
                        ;            Anzeigefeld und Tastenfeld
                        ;

0001                    segment  equ 1  ; Port fuer Anwahl des Segments
0000                    lednr    equ 0  ; Port fuer Anwahl der Anzeige und Tasten
0000                    tastin   equ 0  ; Port fuer Tasteneingabe

                        ; segment
                        ; 7     6       5       4       3       2       1       0
                        ; dp    g       f       e       d       c       b       a
                        ;
                        ; 0= Segment leuchtet 1= Segment aus


                        ; lednr
                        ; 7     6       5       4       3       2       1       0
                        ; anz7  anz6    anz5    anz4    anz3    anz2    anz1    anz0
                        ; x     x       ta5     ta4     ta3     ta2     ta1     ta0
                        ;
                        ; 0= Tastenspalte oder Anzeige aktiviert
```

```
                        ; tastin
                        ; 7     6       5       4       3       2       1       0
                        ; x     x       x       x       tb3     tb2     tb1     tb0
                        ;
                        ; 0= Taste in der angewaehlten Spalte betaetigt

                        ;           a
                        ;         *******
                        ;      f *     * b
                        ;        *     *
                        ;         *******
                        ;      e *     * c
                        ;        *     *
                        ;         *******
                        ;           d       * dp
                        ;


0069                    anzeige::       ; Bei jedem Aufruf wird eine Siebensegment-
                                        ; Anzeige fuer ca. 1ms eingeschaltet
                                        ; dabei wird auch das Tastenfeld abgefragt
                                        ; und ggf. ein Wert im Akku zurueckgegeben
                                        ; es wird aber nicht gewartet, wenn eine
                                        ; Taste gedrueckt bleibt.
                                        ; FF=keine Taste betaetigt
0069    E5              push hl         ; alle Register werden gerettet
006A    D5              push de
006B    C5              push bc
006C    3A 800E         ld a,(anzpoi)   ; Index in das Anzeigefeld ANZFELD
006F    3C              inc a           ; gleich naechsten Wert anwaehlen
0070    E6 07           and 7           ; 0..7 sind erlaubt
0072    32 800E         ld (anzpoi),a   ; und wieder abspeichern
0075    21 8000         ld hl,anzfeld   ; Basis-Adresse des Anzeigefeldes
0078    5F              ld e,a          ; merken fuer Tastencodierung
0079    4F              ld c,a          ; aktuellen Index berechnen
007A    06 00           ld b,0
007C    09              add hl,bc       ; damit ANZFELD+ANZPOI ausgerechnet
007D    0E 7F           ld c,01111111b  ; Startwert fuer Bit-Maske
007F    3C              inc a           ; Zaehler von 1 bis 8 moeglich
0080    47              ld b,a          ; dort Zaehler unterbringen
0081                    anzlp:
0081    CB 01           rlc c           ; Circular nach links schieben
0083    10 FC           djnz anzlp      ; sooft wie Zaehler
0085    3E FF           ld a,0ffh       ; Segmente erst mal ausschalten
0087    D3 00           out (lednr),a   ; um Stoerungen zu vermeiden
0089    7E              ld a,(hl)       ; dann erst neuen Segmentcode
008A    D3 01           out (segment),a ; ausgeben
008C    79              ld a,c          ; dann die Segmente einschalten
```

```
008D  D3 00           out (lednr),a  ; und die Maske als Index ausgeben
008F  01 00A6         ld bc,1000/6   ; 6 Mikrosekunden pro Durchlauf
0092                  anzlp1:        ; bei 4MHz, daher ca. 1ms Warten
0092  0B              dec bc         ; Warteschleife
0093  79              ld a,c
0094  B0              or b
0095  C2 0092         jp nz,anzlp1   ; sonst Segmente zu kurz an
                                     ; ferner wird auch ein
                                     ; Entprellen der Tasten erreicht
0098  DB 00           in a,(tastin)
009A  E6 0F           and 0fh        ; bit 0 bis 3 entscheiden
009C  FE 0F           cp 0fh         ; Taste gedrueckt, dann
009E  20 08           jr nz,gettast  ; Tastencode bilden
00A0  3E FF           ld a,0ffh      ; ist auch gleich Ergebnis
00A2  D3 00           out (lednr),a  ; dann aber ausschalten, um damit
                                     ; gleiche Helligkeit fuer alle
                                     ; Segmente zu erreichen
00A4  C1              pop bc         ; Register wieder zurueck
00A5  D1              pop de
00A6  E1              pop hl
00A7  C9              ret            ; 0ffh in A, keine Taste gedrueckt
                                     ; sonst andere Routine
00A8                  gettast::
00A8  CB 03           rlc e          ; 00000xxx  schieben
00AA  CB 03           rlc e
00AC  CB 03           rlc e
00AE  CB 03           rlc e          ; bis 0xxx0000 erreicht ist
00B0  B3              or e           ; dann in A 0xxxtttt  Gesamtcode
00B1  F5              push af        ; merken und dann Anzeige aus
00B2  3E FF           ld a,0ffh
00B4  D3 00           out (lednr),a
00B6  F1              pop af         ; wieder zurueck
00B7  C1              pop bc         ; und Register vom Start
00B8  D1              pop de
00B9  E1              pop hl
00BA  C9              ret            ; und fertig
00BB                  holetaste::    ; Hier wird eine einzelne Taste geholt
                                     ; und dabei entprellt, wie auch gewartet,
                                     ; bis die Taste wieder losgelassen ist.
                                     ; Das Loslassen wird dabei zu Beginn
                                     ; geprueft, um eine Reaktion bei
                                     ; Tastendruck zu ermoeglichen.
                                     ; Das ganze Tastenfeld wird
                                     ; durchlaufen, bis eine Taste gedrueckt ist,
00BB                  hol10:         ; hier wiederholen bis keine Taste gedrueckt
00BB  06 08           ld b,8         ; 8 mal durchlaufen, dann sind alle Tastenreihen
00BD                  hol11:         ; geprueft, und auch alle Anzeigen durchlaufen.
00BD  CD 0069         call anzeige   ; a=0ffh, dann keine Taste gedrueckt
00C0  FE FF           cp 0ffh        ;
```

```
00C2  20 F7           jr nz,hol10   ; Wenn irgendeine Taste gefunden, dann warten.
00C4  10 F7           djnz hol11    ; Erst erfuellt wenn 8 Mal kein Code da,
00C6                  hol1:         ; denn nur dann ist keine Taste gedrueckt.
00C6  CD 0069         call anzeige  ; Erst wenn Taste da, wieder Programm verlassen.
00C9  FE FF           cp 0ffh       ; Dann warten bis neue Taste da
00CB  28 F9           jr z,hol1     ;
00CD  C9              ret           ; Taste da und steht in Register A


                      ;
                      ; Tastendefinitionen
                      ;
                      ; Tastencode:   0xxxaaaa
                      ;                xxx          binaer codiert 0..7
                      ;                   a3a2a1a0  0=Taste gedrueckt, nur ein Bit
                      ;
                      ;      o o o o o o   a3
                      ;      o o o o o o   a2
                      ;      o o o o o o   a1
                      ;      o o o o o o   a0
                      ;
                      ;      ----xxx----
                      ;      0 ..... 5
                      ;
                      ; fuer xxx=0 steht 000, x=1 ist 001, x=5 ist 101 etc.
                      ;


0057                  crex          equ 01010111b  ; Code fuer die Taste CR
005B                  speichex      equ 01011011b  ; entsprechend bei allen
005D                  plusex        equ 01011101b  ; anderen Codes
005E                  minusex       equ 01011110b  ; Diese Abkuerzungen

0047                  befex         equ 01000111b  ; werden in HEXMON verwendet,
004B                  startex       equ 01001011b  ; um die Lesbarkeit zu
004D                  stepex        equ 01001101b  ; erleichtern
004E                  optex         equ 01001110b  ; Die Anordnung entspricht
                                                   ; der Tastenfeldanordnung
0037                  convex        equ 00110111b  ; ist Taste umw
003B                  prlex         equ 00111011b
003D                  ladex         equ 00111101b
003E                  vglex         equ 00111110b

0027                  mes3ex        equ 00100111b  ; ist Taste pul
002B                  prpex         equ 00101011b  ;
002D                  speex         equ 00101101b
002E                  fillex        equ 00101110b  ; ist Taste ful
```

```
0017                    mes2ex      equ 00010111b   ; ist Taste per
001B                    iolex       equ 00011011b
001D                    prfex       equ 00011101b
001E                    brkex       equ 00011110b

0007                    mes1ex      equ 00000111b   ; ist Taste prm
000B                    iosex       equ 00001011b
000D                    regex       equ 00001101b
000E                    mveex       equ 00001110b

00CE                    tonum::     ; In Akku ein Tastencode.
                                    ; Unterprogr. liefert als Ergebnis 0..F,
                                    ; falls Codierung moeglich,
                                    ; sonst Carry gesetzt und alter Code in A
00CE  C5                 push bc    ; damit koennen Zahlen eingelesen werden
00CF  E5                 push hl
00D0  21 00E6            ld hl,numtab
00D3  06 10              ld b,16    ; 0..F sind 16 Moeglichkeiten
00D5                    to1num:
00D5  BE                 cp (hl)    ; = dann gefunden
00D6  28 07              jr z,to2num
00D8  23                 inc hl     ; zum naechsten Eintrag
00D9  23                 inc hl
00DA  10 F9              djnz to1num ; bis alle durchsucht
00DC  37                 scf        ; CARRY setzen, Code belassen
00DD  18 04              jr to3num
00DF                    to2num:     ; ja gefunden
00DF  23                 inc hl
00E0  7E                 ld a,(hl)  ; dann Code laden
00E1  37                 scf
00E2  3F                 ccf        ; kein Carry, wenn Bereich 0..F eingegeben
00E3                    to3num:     ; wurde
00E3  E1                 pop hl
00E4  C1                 pop bc
00E5  C9                 ret



00E6                    numtab::    ; Tabelle fuer Tastenzuordnung
00E6  0E 00              defb 00001110b,0       ; 0    links steht der Tastencode
00E8  1E 01              defb 00011110b,1       ; 1    und rechts das Ergebnis
00EA  2E 02              defb 00101110b,2       ; 2    nach der Umwandlung
00EC  3E 03              defb 00111110b,3       ; 3
00EE  0D 04              defb 00001101b,4       ; 4
00F0  1D 05              defb 00011101b,5       ; 5
00F2  2D 06              defb 00101101b,6       ; 6
00F4  3D 07              defb 00111101b,7       ; 7
00F6  0B 08              defb 00001011b,8       ; 8
00F8  1B 09              defb 00011011b,9       ; 9
```

```
00FA  2B 0A           defb 00101011b,10     ; A
00FC  3B 0B           defb 00111011b,11     ; B
00FE  07 0C           defb 00000111b,12     ; C
0100  17 0D           defb 00010111b,13     ; D
0102  27 0E           defb 00100111b,14     ; E
0104  37 0F           defb 00110111b,15     ; F

0106                  toseg::         ; in A Wert 0..F , danach
                                      ; Code fuer Segmentausgabe
0106  C5              push bc         ; in A
0107  E5              push hl
0108  21 011A         ld hl,segtab
010B  E6 0F           and 0fh         ; aus Sicherheit gegen Progr. Fehler
010D  4F              ld c,a
010E  06 00           ld b,0
0110  09              add hl,bc       ; Index in Tabelle bilden
0111  4E              ld c,(hl)       ; und Segmentcode holen
0112  3A 8011         ld a,(doton)    ; = 80h  Maske fuer Dot (gethl,getc,getl)
0115  2F              cpl             ; Maske 7fh
0116  A1              and c           ; und Ergebnis in A, Dot dient dem
0117  E1              pop hl          ; zusaetzlichen Einschalten der Punkte
0118  C1              pop bc          ; in der Segmentanzeige
0119  C9              ret


011A                  segtab::                ; Siebensegmentcodes fuer 0..F
011A  C0              defb 11000000b ;0         es ist jeweils der Segmentcode
011B  F9              defb 11111001b ;1         fuer jede einzelne Ziffer
011C  A4              defb 10100100b ;2         aufgefuehrt
011D  B0              defb 10110000b ;3
011E  99              defb 10011001b ;4
011F  92              defb 10010010b ;5
0120  82              defb 10000010b ;6
0121  F8              defb 11111000b ;7
0122  80              defb 10000000b ;8
0123  98              defb 10011000b ;9
0124  88              defb 10001000b ;A
0125  83              defb 10000011b ;B
0126  C6              defb 11000110b ;C
0127  A1              defb 10100001b ;D
0128  86              defb 10000110b ;E
0129  8E              defb 10001110b ;F


012A                  prtac::                 ; A als HEX-Zahl (sedezimal) ausgeben
                                              ; ix zeigt auf Segmentpuffer
                                              ; danach zeigt ix auf leeren Platz
012A  F5              push af                 ; im Segementpuffer
```

```
012B  E6 F0              and 0f0h      ; erstmal hoeherwertige Stelle ausgeben
012D  0F                 rrca          ; dazu werden die oberen vier Bits
012E  0F                 rrca          ; auf die niederwertige Stelle geschoben
012F  0F                 rrca
0130  0F                 rrca          ; in Position 0000xxxx bringen
0131  CD 0106            call toseg    ; Code berechnen
0134  DD 77 00           ld (ix+0),a   ; und ausgeben
0137  DD 23              inc ix        ; dann naechste Stelle
0139  F1                 pop af
013A  E6 0F              and 0fh       ; diesmal niederwertige Stelle
013C  CD 0106            call toseg    ; umcodieren
013F  DD 77 00           ld (ix+0),a   ; und ausgeben in den Segmentpuffer
0142  DD 23              inc ix        ; und naechste Stelle anwaehlen
0144  C9                 ret           ; damit zeigt IX hinter den letzten Platz


0145                     prthl::       ; HL - Register ausgeben als HEX-Zahl
                                       ; IX zeigt auf Segmentpuffer
0145  7C                 ld a,h        ; erstmal hoeherwertige Stelle
0146  CD 012A            call prtac    ; ausgeben
0149  7D                 ld a,l        ; dann niederwertige Stelle
014A  CD 012A            call prtac    ; ausgeben
014D  C9                 ret

014E                     prtbin::      ; Akku binaer ausgeben, Start ix
014E  C5                 push bc       ; retten der Register
014F  06 08              ld b,8        ; 8 Bits, also die Schleife 8 Mal
0151                     prt1bilp:     ; durchlaufen.
0151  DD 36 00 C0        ld (ix+0),11000000b   ; 0 ausgeben
0155  07                 rlca          ; Bit7 zuerst ausgeben dann 6,5,...
0156  30 04              jr nc,prt2bi  ; daher zuerst ins Carry schieben
0158  DD 36 00 F9        ld (ix+0),11111001b   ; 1 ausgeben wenn Carry da
015C                     prt2bi:       ; sonst weiter
015C  DD 23              inc ix        ; naechste Stelle
015E  10 F1              djnz prt1bilp ; 8 Mal insgesamt, fuer alle Stellen
0160  C1                 pop bc
0161  C9                 ret


0162                     prtdez::      ; HL mit Vorzeichen dezimal ausgeben
                                       ; IX zeigt auf Ende (LSB) der Anzeige
                                       ; ACHTUNG, dies ist anders
                                       ; als in prthl, aber Ausgabe ist rechtsbuendig
0162  0E 00              ld c,0        ; c ist Vorzeichen-Merker
0164  CB 7C              bit 7,h       ; Vorzeichen feststellen
0166  28 0A              jr z,prt1dez
0168  0E 01              ld c,1        ; wenn negativ wird die Zahl negiert
016A  E5                 push hl       ; hl:=-hl
016B  D1                 pop de        ; Zweierkomplement bilden
```

```
016C  21 0000           ld hl,0       ;
016F  AF                xor a
0170  ED 52             sbc hl,de     ; 0-hl wird als Operation ausgefuehrt
0172                    prt1dez:      ; weiter geht's
0172  06 10             ld b,16       ; 16 Stellen, hl/10->hl
                                      ; Rest in A
0174  AF                xor a         ; Startwert = 0
0175                    prt2dez:      ; Schleife fuer Divisionsroutine
0175  CB 25             sla l         ; AHL LSB nach links schieben
0177  CB 14             rl h          ; dabei mit 0 auffuellen
0179  CB 17             rl a
017B  FE 0A             cp 10         ;
017D  38 04             jr c,prt3dez  ; wenn kein Carry
017F  D6 0A             sub 10        ; dann Subtraktionsausfuehrung
0181  CB C5             set 0,l       ; und Teilergebnis setzen
0183                    prt3dez:
0183  10 F0             djnz prt2dez  ; 16 Mal das Ganze
0185  CD 0106           call toseg    ; umrechnen in Siebensegmentcode
0188  DD 77 00          ld (ix+0),a   ; Ablegen im Speicher
018B  DD 2B             dec ix        ; rueckwaerts
018D  7C                ld a,h
018E  B5                or l          ;=0 Dann fertig
018F  20 E1             jr nz,prt1dez ; erneut dividieren
0191  79                ld a,c        ;=0 dann positiv
0192  B7                or a
0193  C8                ret z         ; und somit fertig
0194  DD 36 00 BF       ld (ix+0),10111111b    ; sonst Minus ablegen (Segementcode)
0198  DD 2B             dec ix        ; und updaten max 6 Stellen (-32767 z.B.)
019A  C9                ret
019B                    getd1:        ; Routine fuer dezimales Einlesen
019B  CD 00BB           call holetaste ; Tastencode wird geholt
019E  FE 5E             cp minusex    ; wenn ein Minus in die Tastatur
01A0  20 0B             jr nz,getd2   ; eingegeben wird, so muss
01A2  D5                push de       ; der alte Wert negiert werden,
01A3  EB                ex de,hl      ; danach koennen aber keine weiteren
01A4  21 0000           ld hl,0       ; Stellen mehr eingegeben werden,
01A7  AF                xor a         ; da der untere Algorithmus nicht
01A8  ED 52             sbc hl,de     ; fuer negative Zahlen gueltig ist.
01AA  D1                pop de        ; die Routine kehrt aber dennoch ohne
01AB  AF                xor a         ; Fehler zurueck.
01AC  C9                ret
01AD                    getd2:        ; Der Wert in HL wird mit 10 multipliziert
01AD  CD 00CE           call tonum    ; und der Wert der Taste (0..9) darauf addiert
01B0  D8                ret c         ; Rueckkehr wenn keine Zahl. A..F ist keine
01B1  D5                push de       ; dezimale Zahl, wird aber hier nicht erkannt
01B2  C5                push bc       ; man koennte dies mal abfragen.
01B3  E5                push hl       ; Multiplizieren durch geschickte Addition
01B4  D1                pop de        ; hl*10->hl
01B5  29                add hl,hl     ; (hl*4+hl)*2
```

```
01B6  29              add hl,hl     ;
01B7  19              add hl,de     ;
01B8  29              add hl,hl     ; berechnen 10*hl + Wert
01B9  4F              ld c,a        ; der Wert steht noch immer im Akku
01BA  06 00           ld b,0
01BC  09              add hl,bc     ; Das ergibt neuen Wert in HL
01BD  C1              pop bc        ; und somit das neue Ergebnis
01BE  D1              pop de
01BF  37              scf
01C0  3F              ccf
01C1  C9              ret


01C2                  getdez::      ; ix-> Anzeige, HL holen, Dezimaleingabe
                                    ; cr = Abschluss der Eingabe
                                    ; Endezeichen in A, also der Tastencode
                                    ; der zum Abbruch der Zahleingabe
                                    ; fuehrte.
01C2  3E 80           ld a,80h      ; die Punkte auf der Anzeige einschalten
01C4  32 8011         ld (doton),a  ; um eine Eingabe zu kennzeichnen
01C7  DD E5           push ix
01C9  E5              push hl       ; Wert retten, da er in der nachfolgenden
01CA  CD 02B3         call clear    ; Berechnung gebraucht wird
01CD  CD 0162         call prtdez   ; Ausgabe des aktuellen Wertes, IX-> Ende
01D0  E1              pop hl        ; Wert wieder da
01D1  DD E1           pop ix        ;
01D3  CD 019B         call getd1    ; Wert Dezimal und Tasteneingabe
01D6  38 02           jr c,getf1hl  ; bei Abbruch durch die Tastatur weiter
01D8  18 E8           jr getdez     ; sonst das Ganze wiederholen
01DA                  getf1hl:
01DA  F5              push af       ; Die Punkte muessen noch geloescht werden
01DB  AF              xor a         ;
01DC  32 8011         ld (doton),a  ;
01DF  CD 02B3         call clear    ; und die ganze Anzeige loeschen
01E2  DD E5           push ix       ; IX zeigt auf das Ende des Puffers
01E4  E5              push hl       ; HL ist das Ergebnis, daher merken
01E5  CD 0162         call prtdez   ; und ohne Punkte neu ausgeben
01E8  E1              pop hl        ;
01E9  DD E1           pop ix
01EB  F1              pop af        ; in A bleibt der Terminator
01EC  C9              ret           ; also die zuletzt gedrueckte Taste.


01ED                  getc::        ; C=alter Wert der angezeigt wird
                                    ; 0..FF ist der Eingabebereich
                                    ; ix-> Anzeige, Akku holen
                                    ; cr = Abschluss der Eingabe, oder anderes
                                    ; Zeichen, Abschlusszeichen in A
```

```
01ED   E5              push hl        ; dazu wird das Unterprogramm getl
01EE   69              ld l,c         ; verwendet.
01EF   CD 01F5         call getl      ;
01F2   4D              ld c,l
01F3   E1              pop hl
01F4   C9              ret

01F5                   getl:          ; wie oben, jedoch mit L als Datenregister
01F5   3E 80           ld a,80h       ; Punkte anschalten in Hex-Ausgabe,
01F7   32 8011         ld (doton),a   ; um eine Eingabe zu kennzeichnen
01FA   DD E5           push ix
01FC   7D              ld a,l         ; Wert ausgeben
01FD   CD 012A         call prtac
0200   DD E1           pop ix
0202   CD 0242         call getaa     ; Wert holen und HL verrechnen
0205   38 02           jr c,getfl     ; dann Wert in Register L, Terminator in A
0207   18 EC           jr getl        ; sonst wiederholen
0209                   getfl:         ; .. wieder loeschen
0209   F5              push af
020A   AF              xor a
020B   32 8011         ld (doton),a
020E   DD CB 00 FE     set 7,(ix+0)   ; und auch in Anzeige loeschen
0212   DD CB 01 FE     set 7,(ix+1)
0216   F1              pop af         ; in A bleibt der Terminator,
0217   C9              ret            ; also der zuletzt gedrueckte Tastencode



0218                   gethl::        ; ix-> Anzeige, HL holen, diesmal 0..FFFF
                                      ; cr = Abschluss der Eingabe
                                      ; Endezeichen in A
0218   3E 80           ld a,80h       ; Punkte einschalten
021A   32 8011         ld (doton),a
021D   DD E5           push ix
021F   CD 0145         call prthl     ; Ausgabe des aktuellen Wertes
0222   DD E1           pop ix
0224   CD 0242         call getaa     ; Wert
0227   38 02           jr c,getfhl
0229   18 ED           jr gethl       ; und wiederholen
022B                   getfhl:
022B   F5              push af
022C   AF              xor a
022D   32 8011         ld (doton),a   ; Punkte
0230   DD CB 00 FE     set 7,(ix+0)   ; auch im Anzeigepuffer loeschen
0234   DD CB 01 FE     set 7,(ix+1)
0238   DD CB 02 FE     set 7,(ix+2)
023C   DD CB 03 FE     set 7,(ix+3)
0240   F1              pop af         ; in A bleibt der Terminator
0241   C9              ret
```

```
0242                    getaa:          ; Unterprogramm zur Umrechnung
0242  CD 00BB            call holetaste ; Tastencode
0245  CD 00CE            call tonum
0248  D8                ret c          ; dann stop hier A=Zeichen, HL=Wert
0249  C5                push bc        ; merken
024A  29                add hl,hl      ; Multiplikation durch geschickte
024B  29                add hl,hl      ; Addition.
024C  29                add hl,hl
024D  29                add hl,hl      ; berechnen 16*hl + Wert
024E  4F                ld c,a
024F  06 00             ld b,0
0251  09                add hl,bc      ; ergibt neuen Wert
0252  C1                pop bc
0253  37                scf
0254  3F                ccf
0255  C9                ret

                        ; Unterprogramme die haeufig benoetigt werden

0256                    getadr::                ; Adresse holen mit Meldung
0256  E5                push hl                 ; Retten des Startwertes
0257  21 0EF2           ld hl,adrmsg            ; und die Meldung ausgeben

025A                    getaddr:                ; gemeinsamer Programmcode
025A  CD 02A6           call print              ; Ausgabe des Textes
025D  E1                pop hl                  ; und dann ab
025E  DD 21 8004        ld ix,anzfeld+4         ; Position 4 einlesen
0262  CD 0218           call gethl              ; dabei 4 Stellen eingeben
0265  C9                ret                     ; hl = Ergebnis A gueltig

0266                    getvon:                 ; wie oben, jedoch mit anderer
0266  E5                push hl                 ; Meldung  VON xxxx
0267  21 0F02           ld hl,vonmsg
026A  18 EE             jr getaddr

026C                    getnach:                ; NAC xxxx
026C  E5                push hl
026D  21 0F12           ld hl,nachmsg
0270  18 E8             jr getaddr

0272                    getbis:                 ; BIS xxxx
0272  E5                push hl
0273  21 0F0A           ld hl,bismsg
0276  18 E2             jr getaddr

0278                    getmit:                 ; MIT xxxx
0278  E5                push hl
0279  21 0F1A           ld hl,mitmsg
027C  18 DC             jr getaddr
```

```
                        ;
                        ; Kassettenschnittstelle mit dem 6850
                        ; es wird die Baugruppe CAS verwendet

00CA                    cmdcas  equ     0cah    ; Befehlsport
00CB                    datcas  equ     0cbh    ; Datenport

027E                    casinit::       ; 6850 programmieren
027E    3E 53            ld a,53h       ; Baudrate * 1 einstellen
0280    D3 CA            out (cmdcas),a
0282    3E 50            ld a,50h       ; und Baustein bereit setzen
0284    D3 CA            out (cmdcas),a ;
0286    C9               ret

0287                    ri::            ; Ein Zeichen von der Kassette lesen
0287    DB CA            in a,(cmdcas)  ; dazu erst warten bis der 6850 bereit ist,
0289    E6 01            and 1          ; und ein Zeichen angekommen ist.
028B    28 FA            jr z,ri        ; sonst nochmals versuchen und zurueck
028D    DB CB            in a,(datcas)  ; dann Zeichen in Register A einlesen
028F    C9               ret            ; und Ende.

0290                    poo::           ; ein Zeichen von Register C ausgeben.
0290    DB CA            in a,(cmdcas)  ; Erst mal warten bis ggf. vorheriges
0292    E6 02            and 2          ; Zeichen uebertragen wurde,
0294    28 FA            jr z,poo       ; erst dann neues Zeichen ausgeben.
0296    79               ld a,c         ; Zeichen steht vorher in C und kann
0297    D3 CB            out (datcas),a ; dann uebertragen werden
0299    C9               ret


029A                    getri::         ; Einlesen von der Kassette
                                        ; und ausgeben des Binaercodes
                                        ; auf der LED-Anzeige als Kontrolle.
                                        ; dazu werden die Punkte verwendet
029A    3E 7F            ld a,7fh       ; Code fuer Punkt an
029C    D3 01            out (segment),a        ; und fuer alle Segmente ausgeben
029E    CD 0287          call ri        ; A=Datenwert 1=Dunkel
02A1    2F               cpl            ; 1=hell
02A2    D3 00            out (lednr),a  ; binaer ausgeben, um das Datenwort
02A4    2F               cpl            ; zu codieren, dann a wieder restaurieren,
02A5    C9               ret            ; da komplementierte Ausgabe noetig war.


                        ; Weitere Sammlung von Unterprogrammen, die von dem
                        ; eigentlichen Hauptprogramm benoetigt werden.
                        ; I.A. IO unabhaengig.

02A6                    print::         ; hl zeigt auf einen Siebensegmentcode
                                        ; dieser wird nach anzfeld transportiert
```

```
02A6  D5              push de       ; es wird nur HL zerstoert
02A7  C5              push bc
02A8  11 8000         ld de,anzfeld
02AB  01 0008         ld bc,8
02AE  ED B0           ldir
02B0  C1              pop bc
02B1  D1              pop de
02B2  C9              ret

02B3                  clear::
02B3  E5              push hl       ; hl unveraendert lassen
02B4  21 0F9E         ld hl,leer    ; leere Segmente ausgeben
02B7  CD 02A6         call print    ; um Bild zu loeschen
02BA  E1              pop hl
02BB  C9              ret


                      ; es folgen nun Programmteile, die fuer die Einzelschritt-
                      ; Funktion noetig sind.
                      ;

                      ; Laengenbestimmung von Befehlen
                      ;
                      ; hl -> befehl
                      ; b enthaelt nach dem Aufruf die Anzahl der Bytes
                      ; die der Befehl belegt
                      ;
02BC                  length::
02BC  06 00           ld b,0        ;bytezaehler auf 0 stellen
02BE  54              ld d,h
02BF  5D              ld e,l
02C0  7E              ld a,(hl)
02C1  E6 DF           and 0dfh      ;dd,fd - Befehle
02C3  FE DD           cp 0ddh
02C5  CA 0332         jp z,tab5
02C8  7E              ld a,(hl)
02C9  FE CB           cp 0cbh       ;cb - befehle
02CB  CA 032C         jp z,tab3
02CE  FE ED           cp 0edh       ;ed - befehle
02D0  CA 0320         jp z,tab2
02D3  7E              ld a,(hl)
02D4  FE C3           cp 0c3h       ;jmp - befehl
02D6  CA 036E         jp z,b3
02D9  FE CD           cp 0cdh       ;call - befehl
02DB  CA 036E         jp z,b3
02DE  E6 EF           and 0efh
02E0  FE 22           cp 022h
02E2  CA 036E         jp z,b3
02E5  FE 2A           cp 2ah
```

```
02E7  CA 036E           jp z,b3
02EA  E6 CF             and 0cfh
02EC  FE 01             cp 1
02EE  CA 036E           jp z,b3         ;ld xx,nn
02F1  E6 C7             and 0c7h
02F3  FE C2             cp 0c2h         ;jp cond
02F5  CA 036E           jp z,b3
02F8  FE C4             cp 0c4h         ;call cond
02FA  CA 036E           jp z,b3
02FD  7E                ld a,(hl)
02FE  E6 F7             and 0f7h
0300  FE 10             cp 10h          ;djnz , jr
0302  CA 036F           jp z,b2
0305  FE D3             cp 0d3h         ;in , out
0307  CA 036F           jp z,b2
030A  E6 E7             and 0e7h
030C  FE 20             cp 20h          ;jr cond
030E  CA 036F           jp z,b2
0311  E6 C7             and 0c7h
0313  FE 06             cp 6            ;ld n
0315  CA 036F           jp z,b2
0318  FE C6             cp 0c6h         ;op n
031A  CA 036F           jp z,b2
031D  C3 0370           jp b1           ;rest 1 byte
                        ;
0320                    tab2:
0320  23                inc hl
0321  7E                ld a,(hl)
0322  E6 C7             and 0c7h
0324  FE 43             cp 43h          ;ld (nn), xx ld xx,(nn)
0326  CA 036D           jp z,b4
0329  C3 036F           jp b2
032C                    tab3:
032C  C3 036F           jp b2           ;alle 2 bytes
032F                    tab4:
032F  C3 036D           jp b4           ;alle 4 bytes
0332                    tab5:           ;dd,fd
0332  23                inc hl
0333  7E                ld a,(hl)
0334  FE CB             cp 0cbh         ;dd,fd cb
0336  CA 036D           jp z,b4
0339  FE 21             cp 21h
033B  CA 036D           jp z,b4         ;ld ii,nn
033E  E6 FE             and 0feh
0340  FE 34             cp 34h
0342  CA 036E           jp z,b3         ;inc dec (ii)
0345  E6 F8             and 0f8h
0347  FE 70             cp 70h          ;ld (ii),r
0349  CA 036E           jp z,b3
```

```
034C   7E              ld a,(hl)
034D   FE 36           cp 36h          ;auch pseudo
034F   CA 036D         jp z,b4
0352   E6 C7           and 0c7h
0354   FE 06           cp 6
0356   CA 036E         jp z,b3
0359   E6 C7           and 0c7h
035B   FE 02           cp 2            ;ld (nn),ii
035D   CA 036D         jp z,b4
0360   7E              ld a,(hl)
0361   D6 40           sub 40h
0363   E6 87           and 87h
0365   FE 06           cp 6
0367   CA 036E         jp z,b3         ;ld r,(ii)
036A   C3 036F         jp b2
036D                  b4:
036D   04              inc b
036E                  b3:
036E   04              inc b
036F                  b2:
036F   04              inc b
0370                  b1:
0370   04              inc b
0371   EB              ex de,hl        ; alter Pointer nach hl
0372   C9              ret             ; b gueltig, Anzahl bytes 1..4


                      ; Unterprogramme fuer Registerretten,
                      ; denn die Benutzerregister muessen beim Einzelschritt
                      ; nach jedem Schritt festgehalten werden und vor
                      ; jedem Schritt wieder zurueck geladen werden.

0373                  pushall::
0373   ED 43 8021     ld (bcsto),bc
0377   ED 53 8023     ld (desto),de
037B   22 8025        ld (hlsto),hl
037E   F5             push af
037F   C1             pop bc
0380   ED 43 801F     ld (pswlsto),bc
0384   08             ex af,af'
0385   D9             exx
0386   ED 43 8029     ld (bc2sto),bc
038A   ED 53 802B     ld (de2sto),de
038E   22 802D        ld (hl2sto),hl
0391   F5             push af
0392   C1             pop bc
0393   ED 43 8027     ld (psw2sto),bc
0397   ED 57          ld a,i
```

```
0399   32 8035          ld (iregsto),a
039C   ED 5F            ld a,r
039E   32 8036          ld (rregsto),a
03A1   DD 22 802F       ld (ixsto),ix
03A5   FD 22 8031       ld (iysto),iy
03A9   3A 800E          ld a,(anzpoi)   ; Anzeigepointer retten
03AC   32 8010          ld (oldanz),a   ; wenn man ANZEIGE durchlaufen will
03AF   3A 800F          ld a,(anzsys)   ; sonst Stoerungen auf der Anzeige
03B2   32 800E          ld (anzpoi),a   ;
03B5   2A 801D          ld hl,(pcsto)   ; pc Wert
03B8   C9               ret


03B9                    popall::        ; Register zurueckholen
03B9   3A 800E          ld a,(anzpoi)   ; System Anzeigepointer
03BC   32 800F          ld (anzsys),a   ; retten
03BF   3A 8010          ld a,(oldanz)   ; Benutzer zurueckholen
03C2   32 800E          ld (anzpoi),a   ; Anzeigepointer zurueck
03C5   22 801D          ld (pcsto),hl   ;retten hl
03C8   FD 2A 8031       ld iy,(iysto)
03CC   DD 2A 802F       ld ix,(ixsto)
03D0   3A 8035          ld a,(iregsto)
03D3   ED 47            ld i,a
                        ; refresh nicht schreiben
03D5   ED 4B 8027       ld bc,(psw2sto)
03D9   C5               push bc
03DA   F1               pop af
03DB   2A 802D          ld hl,(hl2sto)
03DE   ED 5B 802B       ld de,(de2sto)
03E2   ED 4B 8029       ld bc,(bc2sto)
03E6   D9               exx
03E7   08               ex af,af'
03E8   ED 4B 801F       ld bc,(psw1sto)
03EC   C5               push bc
03ED   F1               pop af
03EE   2A 8025          ld hl,(h1sto)
03F1   ED 5B 8023       ld de,(desto)
03F5   ED 4B 8021       ld bc,(bcsto)
03F9   C9               ret
                        ;
```

```
                        ; Es folgt die Implementierung des Einzelschritt-
                        ; Befehls. Manche Befehle werden einfach in ein
                        ; RAM-Gebiet gebracht und dort ausgefuehrt, andere,
                        ; alle die den Programmzaehler veraendern, werden
                        ; durch Unterprogramme emuliert (EMULIEREN ist
                        ; aehnlich dem Begriff simulieren, wird jedoch
                        ; in diesem Zusammenhang gebraucht, da ein
                        ; Prozessor einen Prozessor nachbildet, in diesem
                        ; Fall sogar sich selbst).

03FA                    befexx::                ;einen NICHT-Sprungbefehl
                                                ;ausfuehren ueber modber
03FA  3E C3             ld a,0c3h               ;jp befehl
03FC  32 803D           ld (modspg),a
03FF  E5                push hl
0400  21 042A           ld hl,modr              ;ruecksprung eintragen
0403  22 803E           ld (modspg+1),hl        ;adresse
0406  E1                pop hl
0407  AF                xor a                  ;loeschen rueckw.
0408  06 04             ld b,4
040A  11 803C           ld de,modber+3          ;4ter Wert
040D                    bef1:
040D  12                 ld (de),a              ;auf 0
040E  1B                 dec de
040F  10 FC             djnz bef1
0411  D5                push de
0412  CD 02BC           call length             ;b=laenge Befehl
0415  D1                pop de
0416                    bef2:
0416  7E                 ld a,(hl)              ;hl -> Befehl
0417  13                 inc de
0418  12                 ld (de),a
0419  23                 inc hl
041A  10 FA             djnz bef2               ;Befehl ablegen
041C  CD 03B9           call popall             ;Register belegen
041F  ED 73 8037        ld (syssp),sp           ;retten stackpointer system
0423  ED 7B 8033        ld sp,(usersp)
0427  C3 8039           jp modber               ;ausfuehren
042A                    modr::                  ;Ruecksprung von mod
042A  ED 73 8033        ld (usersp),sp
042E  ED 7B 8037        ld sp,(syssp)
0432  CD 0373           call pushall            ;alle Register wieder merken
0435  C9                ret                     ;Schritt ausgefuehrt


0436                    rstex::                 ;rst Befehl
0436  7E                ld a,(hl)
0437  23                inc hl
0438  ED 73 8037        ld (syssp),sp
```

```
043C  ED 7B 8033        ld sp,(usersp)
0440  E5                push hl                 ;ablegen Aufrufadresse
0441  ED 73 8033        ld (usersp),sp
0445  ED 7B 8037        ld sp,(syssp)
0449  26 00             ld h,0
044B  E6 38             and 038h
044D  6F                ld l,a                  ;neuer pc-Stand
044E  C9                ret

                        ;

044F                    callex::
044F  23                inc hl
0450  23                inc hl
0451  23                inc hl                  ;ret adr
0452  ED 73 8037        ld (syssp),sp
0456  ED 7B 8033        ld sp,(usersp)
045A  E5                push hl                 ;ret adr dorthin
045B  ED 73 8033        ld (usersp),sp
045F  ED 7B 8037        ld sp,(syssp)
0463  2B                dec hl
0464  7E                ld a,(hl)
0465  2B                dec hl
0466  6E                ld l,(hl)
0467  67                ld h,a                  ;Sprungziel
0468  C9                ret


0469                    pchlex::
0469  2A 8025           ld hl,(hlsto)           ;Sprung
046C  C9                ret

046D                    pcixex::
046D  2A 802F           ld hl,(ixsto)
0470  C9                ret

0471                    pciyex::
0471  2A 8031           ld hl,(iysto)
0474  C9                ret

0475                    retex::
0475  ED 73 8037        ld (syssp),sp
0479  ED 7B 8033        ld sp,(usersp)
047D  E1                pop hl
047E  ED 73 8033        ld (usersp),sp
0482  ED 7B 8037        ld sp,(syssp)
0486  C9                ret
```

```
0487                    jmpex::
0487  23                inc hl
0488  7E                ld a,(hl)
0489  23                inc hl
048A  66                ld h,(hl)
048B  6F                ld l,a          ;Sprung Ausfuehrung
048C  C9                ret

048D                    jrex::
048D  23                inc hl
048E  7E                ld a,(hl)
048F  23                inc hl          ;neuer Befehl
0490  4F                ld c,a
0491  B7                or a            ;Flags setzen
0492  FA 0499           jp m,subjr      ;Sprung zurueck
0495  06 00             ld b,0
0497  09                add hl,bc       ;pc neu = pc + displ
0498  C9                ret
0499                    subjr:
0499  06 FF             ld b,0ffh       ;pc neu = pc +ff displ
049B  09                add hl,bc
049C  C9                ret

049D                    djnzex::
049D  ED 4B 8021        ld bc,(bcsto)
04A1  05                dec b
04A2  ED 43 8021        ld (bcsto),bc
04A6  20 E5             jr nz,jrex      ;ausfuehren Sprung
04A8  23                inc hl
04A9  23                inc hl
04AA  C9                ret             ;sonst weiter

04AB                    callcex::       ;bedingter Unterprogramm-Aufruf
04AB  E5                push hl
04AC  21 044F           ld hl,callex    ;ohne Bedingung
04AF  22 803A           ld (modber+1),hl
04B2  21 04CF           ld hl,notex
04B5  22 803D           ld (modber+4),hl        ;
04B8  3E C3             ld a,0c3h
04BA  32 803C           ld (modber+3),a
04BD  E1                pop hl
04BE  7E                ld a,(hl)       ;daraus Sprungbefehl machen
04BF  E6 38             and 038h
04C1  F6 C2             or 0c2h
04C3  32 8039           ld (modber),a   ;in modbereich
04C6  ED 4B 801F        ld bc,(psw1sto)
04CA  C5                push bc
04CB  F1                pop af          ;Flags gueltig
04CC  C3 8039           jp modber
```

```
04CF                    notex::         ;wenn nicht ausgefuehrt
04CF   23               inc hl
04D0                    not2ex::
04D0   23               inc hl
04D1                    not1ex::
04D1   23               inc hl
04D2   C9               ret             ;fertig


04D3                    jmpcex::
04D3   E5               push hl
04D4   21 0487          ld hl,jmpex     ;ohne Bedingung
04D7   22 803A          ld (modber+1),hl
04DA   21 04CF          ld hl,notex
04DD   22 803D          ld (modber+4),hl
04E0   3E C3            ld a,0c3h
04E2   32 803C          ld (modber+3),a
04E5   E1               pop hl
04E6   7E               ld a,(hl)
04E7   32 8039          ld (modber),a   ;Befehl direkt
04EA   ED 4B 801F       ld bc,(psw1sto)
04EE   C5               push bc
04EF   F1               pop af
04F0   C3 8039          jp modber

04F3                    retcex::
04F3   E5               push hl
04F4   21 0475          ld hl,retex     ;ohne Bedingung
04F7   22 803A          ld (modber+1),hl
04FA   21 04D1          ld hl,not1ex
04FD   22 803D          ld (modber+4),hl        ;
0500   3E C3            ld a,0c3h
0502   32 803C          ld (modber+3),a
0505   E1               pop hl
0506   7E               ld a,(hl)
0507   E6 38            and 038h
0509   F6 C2            or 0c2h         ;in Sprung umwandeln
050B   32 8039          ld (modber),a
050E   ED 4B 801F       ld bc,(psw1sto)
0512   C5               push bc
0513   F1               pop af
0514   C3 8039          jp modber

0517                    jrcex::
0517   E5               push hl
0518   21 048D          ld hl,jrex      ;ohne Bedingung
051B   22 803A          ld (modber+1),hl
051E   21 04D0          ld hl,not2ex
```

```
0521  22 803D           ld (modber+4),hl        ;
0524  3E C3             ld a,0c3h
0526  32 803C           ld (modber+3),a
0529  E1                pop hl
052A  7E                ld a,(hl)
052B  E6 18             and 18h
052D  F6 C2             or 0c2h
052F  32 8039           ld (modber),a   ;als Sprung dorthin
0532  ED 4B 801F        ld bc,(psw1sto)
0536  C5                push bc
0537  F1                pop af
0538  C3 8039           jp modber       ;und ausfuehren

                        ;



                        ; Routine STEP fuehrt einen Schritt aus
                        ; dabei ist hl der aktuelle PC
                        ; er muss also vorher mit dem PCSTO
                        ; geladen werden.
                        ; Nach Aufruf ist in HL der neue
                        ; Programmzaehlerstand

053B                    step::
053B  7E                ld a,(hl)       ;wichtig ruecksetzen nach Step
053C  FE C9             cp 0c9h         ;ret Befehl
053E  CA 0475           jp z,retex
0541  FE CD             cp 0cdh
0543  CA 044F           jp z,callex     ;
0546  FE C3             cp 0c3h
0548  CA 0487           jp z,jmpex
054B  FE 18             cp 18h
054D  CA 048D           jp z,jrex
0550  FE 10             cp 010h
0552  CA 049D           jp z,djnzex
0555  FE E9             cp 0e9h
0557  CA 0469           jp z,pchlex
055A  E6 E7             and 0e7h
055C  FE 20             cp 20h
055E  CA 0517           jp z,jrcex
0561  E6 C7             and 0c7h
0563  FE C2             cp 0c2h
0565  CA 04D3           jp z,jmpcex
0568  FE C4             cp 0c4h
056A  CA 04AB           jp z,callcex    ;
056D  FE C7             cp 0c7h
056F  CA 0436           jp z,rstex
0572  FE C0             cp 0c0h
```

```
0574  CA 04F3          jp z,retcex
                       ;
0577  7E              ld a,(hl)       ;Wert testen
0578  FE ED            cp 0edh         ;Umschaltbefehle
057A  C2 058D          jp nz,con2
057D  23               inc hl
057E  7E               ld a,(hl)
057F  2B               dec hl
0580  FE 4D            cp 4dh
0582  CA 0475          jp z,retex      ;reti
0585  FE 45            cp 45h
0587  CA 0475          jp z,retex      ;retn
058A  C3 03FA          jp befexx       ;STD-Befehl
                       ;
058D                   con2:
058D  7E               ld a,(hl)
058E  E6 DF            and 0dfh
0590  FE DD            cp 0ddh
0592  C2 03FA          jp nz,befexx
0595  23               inc hl
0596  7E               ld a,(hl)
0597  2B               dec hl
0598  FE E9            cp 0e9h
059A  C2 03FA          jp nz,befexx
059D  7E               ld a,(hl)
059E  FE DD            cp 0ddh
05A0  CA 046D          jp z,pcixex
05A3  C3 0471          jp pciyex
                       ;
                       ; Ende step Befehl



                       ;*****************************************
                       ; --- Befehlsimplementierung ---          *
                       ; Alle Programme sind als Unterprogramme  *
                       ; ausgefuehrt, so dass sie auch getrennt  *
                       ; genutzt werden koennen                  *
                       ;*****************************************


                       ; Speicherinhalt anzeigen, modifizieren


05A6                   dospeicher:
05A6  21 0F9E           ld hl,leer
05A9  CD 02A6           call print     ; Leerfeld ausgeben
05AC  2A 801D           ld hl,(pcsto)  ; immer von 0 ausgehen
```

```
05AF  CD 0256          call getadr    ; ADRESSE holen mit Anzeige A=Terminator
05B2  FE 47            cp befex       ; Abbruch moeglich
05B4  C8               ret z
05B5                   sploop:        ; Abbruch erst durch crex
05B5  CD 02B3          call clear     ; erst Bild loeschen
05B8  DD 21 8000       ld ix,anzfeld
05BC  CD 0145          call prthl     ; Adresse linksseitig ausgeben
05BF  DD 21 8006       ld ix,anzfeld+6       ; Rechts Datenwert
05C3  4E               ld c,(hl)      ; direkt aus Speicher holen
05C4  CD 01ED          call getc      ; und manipulieren, moeglich, 1=Wert
05C7  FE 47            cp befex       ; Abbruch
05C9  C8               ret z
05CA  FE 57            cp crex        ;
05CC  20 08            jr nz,sp1l     ; crex erlaubt, beendet Eingabe
05CE  79               ld a,c         ; Wert holen
05CF  77               ld (hl),a      ; ablegen im Speicher
05D0  BE               cp (hl)        ; und pruefen, ob abgelegt
05D1  20 E2            jr nz,sploop   ; wenn nicht, dann wieder zurueck: FEHLER
05D3  23               inc hl         ; naechste Speicherzelle, falls es ok war
05D4  18 DF            jr sploop      ;
05D6                   sp1l:
05D6  FE 5D            cp plusex      ; weiter ohne modifizieren
05D8  20 03            jr nz,sp2l
05DA  23               inc hl         ; naechste Speicherzelle anwaehlen
05DB  18 D8            jr sploop
05DD                   sp2l:
05DD  FE 5E            cp minusex
05DF  20 03            jr nz,sp3l
05E1  2B               dec hl         ; vorherige Speicherzelle anwaehlen
05E2  18 D1            jr sploop
05E4                   sp3l:
05E4  FE 4E            cp optex       ; andere Darstellungsart
05E6  C2 0622          jp nz,spend    ; Ende der Bearbeitung
05E9                   spaloop:       ; zweite Schleife mit Laengentreue
05E9  CD 02B3          call clear     ; Bild erst mal loeschen
05EC  E5               push hl        ; Dann Anzahl der Bytes berechnen
05ED  CD 02BC          call length    ; b=Befehlslaenge 1,2,3,4 moeglich
05F0  DD 21 8000       ld ix,anzfeld
05F4  C5               push bc        ; Laenge merken in B
05F5                   sp10l:
05F5  7E               ld a,(hl)      ; erster Wert
05F6  CD 012A          call prtac
05F9  23               inc hl         ; ix wird automatisch erhoeht
05FA  10 F9            djnz sp10l
05FC  CD 00BB          call holetaste ; a=Code bei CR weiterschalten
05FF  C1               pop bc         ; alte Speicheradresse
0600  E1               pop hl         ; Laenge in B, merken fuer spaeter
0601  FE 5B            cp speichex    ;
0603  28 04            jr z,sp0al
```

```
0605  FE 57            cp crex      ;
0607  20 06            jr nz,sp1al  ; crex erlaubt, beendet Eingabe
0609                   sp0al:
0609  48               ld c,b
060A  06 00            ld b,0
060C  09               add hl,bc    ; + Befehlslaenge
060D  18 DA            jr spaloop   ;
060F                   sp1al:
060F  FE 5D            cp plusex    ; weiter ohne modifizieren
0611  20 03            jr nz,sp2al
0613  23               inc hl       ; naechste Speicherzelle anwaehlen
0614  18 D3            jr spaloop
0616                   sp2al:
0616  FE 5E            cp minusex
0618  20 03            jr nz,sp3al
061A  2B               dec hl       ; vorherige Speicherzelle anwaehlen
061B  18 CC            jr spaloop
061D                   sp3al:
061D  FE 4E            cp optex     ; andere Darstellungsart (wieder zurueck)
061F  CA 05B5          jp z,sploop  ; sonst Fehler
0622                  spend:        ; keine der gueltigen Tasten, dann fertig
0622  22 801D          ld (pcsto),hl  ; letzte Zelle merken als neuer PCstand
0625  22 8018          ld (endadr),hl ; gleich merken als Default fuer CAS,PROMMER
0628  C9              ret


0629                  doprog::       ; Programm starten
0629  2A 801D          ld hl,(pcsto)  ;
062C  CD 0256          call getadr    ; mit altem Defaultwert
062F  22 801D          ld (pcsto),hl  ; ist wieder neuer Wert, CR startet
0632  FE 57            cp crex        ; Bei befex erfolgt auch Abbruch
0634  C0               ret nz         ; sonst Abbruch und keine Ausfuehrung
0635  CD 066D          call setbreak  ; Breakpoints gueltig setzen
0638  ED 73 8037       ld (syssp),sp
063C  ED 7B 8033       ld sp,(usersp) ; auch Benutzerstack laden
0640  E5               push hl        ; auf Userstack legen fuer Start
0641  CD 03B9          call popall    ; hl wird nochmals gerettet, Stack nach unten
                                      ; geaendert, bei spez prog gefaehrlich
0644  C9               ret            ; und Benutzerprogramm starten,
                                      ; da Adresse auf dem Stack liegt

0645                  break::        ; Eingang bei Breakpunkt
                                      ; erfolgt bei RST 6, Code=0f7h
0645  CD 0373          call pushall   ; erst mal Register retten
0648  E1               pop hl         ; Aufruferadresse
0649  2B               dec hl         ; zeigt dann auf Breakpunkt
064A  22 801D          ld (pcsto),hl  ; ist neue Startadresse
064D  ED 73 8033       ld (usersp),sp ; neuer Stackstand Benutzer
                                      ; dann Breakpoints entfernen
```

```
0651  31 807C           ld sp,stack    ; System Stackpointer
0654  CD 0689           call freebreak ; Alle Datenwerte wieder an die
0657  21 0EFA           ld hl,brkmsg   ; Breakpointstellen zurueckschreiben,
065A  CD 02A6           call print     ; dies geschieht in FREEBREAK.
065D  DD 21 8004        ld ix,anzfeld+4
0661  2A 801D           ld hl,(pcsto)
0664  CD 0145           call prthl     ; BRK xxxx ausgeben mit Adresse des PC-Wertes
0667  CD 00BB           call holetaste ; Ausgabe der brk-Adresse
066A  C3 0E0B           jp schleife    ; ins Hauptprogramm zurueck nach Tasteneingabe


066D                    setbreak::     ; alle gueltigen Breakpoints
                                       ; werden gesetzt und die Befehle
                                       ; 0=ungueltiger Breakpoint
                                       ; gerettet, brkzahl gibt 0..n Anzahl an
066D  C5                push bc        ; alle Register beibehalten
066E  D5                push de
066F  E5                push hl
0670  21 8041           ld hl,brk1     ; Zeiger auf erstes Speicherfeld
0673  06 03             ld b,3         ; Anzahl Breakpoints
0675                    set1l:         ; Schleifeneingang
0675  5E                ld e,(hl)      ; LSB Adresse
0676  23                inc hl
0677  56                ld d,(hl)      ; MSB Adresse
0678  23                inc hl         ; zeigt dann auf Befehlsplatz
0679  7A                ld a,d
067A  B3                or e           ; = 0 dann ueberspringen
067B  28 05             jr z,setnext
067D  1A                ld a,(de)      ; alten Wert holen
067E  77                ld (hl),a      ; im Speicherfeld ablegen
067F  3E F7             ld a,0f7h      ; RST1-Befehl dort ablegen
0681  12                ld (de),a      ;
0682                    setnext:       ;
0682  23                inc hl         ; hl zeigt auf naechstes Feld
0683  10 F0             djnz set1l     ; ENDWHILE, bei set1l wieder abfragen
0685  E1                pop hl
0686  D1                pop de
0687  C1                pop bc
0688  C9                ret



0689                    freebreak::    ; alle Breakpoint-Daten werden restauriert
                                       ;
0689  C5                push bc        ; alle Register beibehalten
068A  D5                push de
068B  E5                push hl
068C  21 8041           ld hl,brk1     ; Zeiger auf erstes Speicherfeld
068F  06 03             ld b,3         ; Zaehler
```

```
0691                    frei1:          ; Schleifeneingang WHILE ZAEHLER>0 DO
0691    5E               ld e,(hl)      ; LSB Adresse
0692    23               inc hl
0693    56               ld d,(hl)      ; MSB Adresse
0694    23               inc hl         ; zeigt dann auf Befehlsplatz
0695    7A               ld a,d
0696    B3               or e           ; =0 dann nicht veraendern
0697    28 02            jr z,freskip   ; sonder ueberspringen
0699    7E               ld a,(hl)      ; alten Wert holen
069A    12               ld (de),a      ; im Speicher ablegen und damit Befehl eintragen
069B                    freskip:
069B    23               inc hl         ; hl zeigt auf naechstes Feld
069C    10 F3            djnz frei1     ; ENDWHILE, bei frei1 wieder Abfrage
069E                    fref:
069E    E1               pop hl
069F    D1               pop de
06A0    C1               pop bc
06A1    C9               ret


06A2                    dostep::        ; Einzelschritt ausfuehren
06A2    AF               xor a
06A3    32 8012          ld (ausmode),a ; immer mit Adr-Mode beginnen,
06A6    2A 801D          ld hl,(pcsto)  ; wenn eine Ausgabe erfolgt
06A9    CD 0256          call getadr    ; Startadresse einstellen
06AC    FE 47            cp befex
06AE    C8               ret z          ; Abbruch
06AF    22 801D          ld (pcsto),hl  ; einstellen
06B2    FE 57            cp crex        ; STEP und CR zulaessig
06B4    28 0C            jr z,do2loop   ; erst mal anzeigen
06B6    FE 4D            cp stepex      ; muss dann aber step sein
06B8    C0               ret nz         ; muss cr sein sonst Abbruch
06B9                    doloop::        ; dann nach crex ausfuehren
06B9    2A 801D          ld hl,(pcsto)  ; laden der alten Adresse
06BC    CD 053B          call step      ; einen Schritt ausfuehren nun
06BF    22 801D          ld (pcsto),hl  ; neuer Zaehlerstand
06C2                    do2loop::
                                        ; dann Register oder Speicher anzeigen
06C2    CD 076B          call ausgabe   ; Ausgabeprogramm je nach Mode
06C5    CD 00BB          call holetaste ; nun mit Step weiter, oder Mode
                                        ; einstellen
06C8    FE 47            cp befex       ; Abbruch
06CA    C8               ret z
06CB    FE 4D            cp stepex
06CD    28 EA            jr z,doloop    ; und ausfuehren
06CF    FE 57            cp crex
06D1    28 E6            jr z,doloop    ; beides erlaubt
06D3    FE 4E            cp optex       ; Adresse, Befehl
06D5    20 13            jr nz,do3      ; oder Laengentreue Ausgabe
```

```
06D7  3A 8012          ld a,(ausmode)
06DA  B7               or a          ; auf 0 abfragen
06DB  28 06            jr z,do30     ; wenn ja, dann weiter
06DD  AF               xor a         ; dann umschalten auf ADR-Mode
06DE  32 8012          ld (ausmode),a ; und merken
06E1  18 DF            jr do2loop
06E3                  do30:
06E3  3E 01            ld a,1
06E5  32 8012          ld (ausmode),a ; Befehls-Laengenausgabe
06E8  18 D8            jr do2loop
06EA                  do3:
06EA  FE 0D            cp regex      ; Register default
06EC  20 0D            jr nz,do4
06EE  3E 02            ld a,2        ; Register-Mode
06F0  32 8012          ld (ausmode),a ; einstellen
06F3  3A 8015          ld a,(ausreg) ; alte Einstellung verwenden
06F6  CD 0856          call do1reg   ; komplette Registermodifikation
                                     ; moeglich
06F9  18 C7            jr do2loop    ; mit BEF, oder STEP etc. Abbruch moeglich
06FB                  do4:
06FB  FE 5B            cp speichex   ; Speicher ansehen
06FD  20 1A            jr nz,do5     ; sonst weiter
06FF  CD 02B3          call clear    ; Feld erstmal loeschen
0702  DD 21 8004       ld ix,anzfeld+4 ; zur Unterscheidung
0706  2A 8013          ld hl,(ausadr) ; alte Einstellung
0709  CD 0218          call gethl    ; dann neuen Wert holen
070C  FE 47            cp befex      ; Abbruch moeglich
070E  C8               ret z
070F  22 8013          ld (ausadr),hl ; merken fuer Ausgabe
0712  3E 03            ld a,3        ; Speicher anzeigen
0714  32 8012          ld (ausmode),a ;
0717  18 A9            jr do2loop
0719                  do5:
0719  FE 4B            cp startex    ; Bis Breakpunkt laufen lassen
071B  20 3E            jr nz,do6     ; aber nicht in Echtzeit, sondern
                                     ; mit laufendem Display,
071D                  do5lp:         ; Schleifenbeginn
071D  ED 5B 801D       ld de,(pcsto) ; erst Mal Breakpoints pruefen
0721  2A 8041          ld hl,(brk1)  ; wenn Subtraktion=0, dann erfuellt
0724  AF               xor a         ; wenn hl=0, dann eigentlich kein Breakpoint
0725  ED 52            sbc hl,de     ; jedoch ist dies hier nicht wichtig
0727  7D               ld a,l        ; ist Ergebnis=0, dann Abbruch der Schleife
0728  B4               or h          ; also wenn z.B. Adresse 0 angesprungen wird
0729  CA 06C2          jp z,do2loop  ; und zurueck zur Hauptschleife
072C  2A 8044          ld hl,(brk2)  ; so alle Breakpoints abfragen
072F  AF               xor a         ;
0730  ED 52            sbc hl,de
0732  7D               ld a,l
0733  B4               or h          ; =0 dann beide identisch
```

```
0734  CA 06C2          jp z,do2loop   ; und Ausfuehrung beenden
0737  2A 8047          ld hl,(brk3)   ; letzten Breakpoint auch
073A  AF               xor a
073B  ED 52            sbc hl,de
073D  7D               ld a,l
073E  B4               or h           ; und wieder testen
073F  CA 06C2          jp z,do2loop
0742  CD 076B          call ausgabe   ; Anzeigebereich aktualisieren
0745  CD 0069          call anzeige   ; Anzeige mit je einem Digit aufrufen
0748  FE 47            cp befex       ; nur wenn Taste BEF gedrueckt, Abbruch
074A  C8               ret z          ; sonst weiterschreiten, ca 1ms pro Schritt
074B  FE 4E            cp optex       ; STOP wenn OPT gedrueckt wird
074D  CA 06C2          jp z,do2loop
0750  2A 801D          ld hl,(pcsto)  ; und einen Schritt ausfuehren
0753  CD 053B          call step
0756  22 801D          ld (pcsto),hl  ; neuer Befehlsstand
0759  18 C2            jr do5lp       ; und erneut ausfuehren

075B                   do6:
075B  FE 5E            cp minusex     ; Display ganz ausschalten
075D  20 0B            jr nz,do7
075F  3E 04            ld a,4
0761  32 8012          ld (ausmode),a ; Display bleibt dann dunkel, bzw zeigt nur
                                      ; Anwenderteil an
0764  CD 02B3          call clear     ; noch einmal loeschen
0767  C3 06C2          jp do2loop     ; wichtig fuer Steps durch Displayroutinen

076A                   do7:
076A  C9               ret            ; Ende Abbruch, z.b. mit BEF, - etc.


076B                   ausgabe::      ; in Abh. vom Mode Anzeigen auf dem Display
076B  3A 8012          ld a,(ausmode)
076E  B7               or a
076F  20 16            jr nz,aus1
0771  CD 02B3          call clear     ; Adresse und Befehlsinhalt ausgeben
0774  DD 21 8000       ld ix,anzfeld
0778  2A 801D          ld hl,(pcsto)  ; naechster Befehl
077B  CD 0145          call prthl
077E  DD 21 8006       ld ix,anzfeld+6
0782  7E               ld a,(hl)      ; Inhalt auch ausgeben
0783  CD 012A          call prtac
0786  C9               ret
0787                   aus1:
0787  FE 01            cp 1
0789  20 15            jr nz,aus2
078B  2A 801D          ld hl,(pcsto)  ;
078E  CD 02B3          call clear     ; Bild erst mal loeschen
0791  CD 02BC          call length    ; b=Befehlslaenge 1,2,3,4 moeglich
```

```
0794  DD 21 8000          ld ix,anzfeld
0798                      as10l:
0798  7E                  ld a,(hl)       ; erster Wert
0799  CD 012A             call prtac
079C  23                  inc hl          ; ix wird automatisch erhoeht
079D  10 F9               djnz as10l
079F  C9                  ret
07A0                      aus2:
07A0  FE 02               cp 2            ; Registerinhalt
07A2  20 13               jr nz,aus3
07A4  3A 8015             ld a,(ausreg)   ; Registernr 0..n
07A7  CD 07E2             call regaus     ; Ausgabe des Registers
07AA  4E                  ld c,(hl)       ; lsb Wert
07AB  23                  inc hl
07AC  46                  ld b,(hl)
07AD  69                  ld l,c
07AE  60                  ld h,b
07AF  DD 21 8004          ld ix,anzfeld+4 ; Dorthin Wert
07B3  CD 0145             call prthl
07B6  C9                  ret
07B7                      aus3:
07B7  FE 03               cp 3            ; Speicherzelle
07B9  20 1C               jr nz,aus4      ; sonst Fehler
07BB  CD 02B3             call clear
07BE  DD 21 8000          ld ix,anzfeld
07C2  2A 8013             ld hl,(ausadr)
07C5  CD 0145             call prthl      ; Adresse
07C8  DD 36 00 BF         ld (ix+0),10111111b ; MINUS-Zeichen
07CC  DD 23               inc ix          ; naechste freie Zelle
07CE  7E                  ld a,(hl)
07CF  CD 012A             call prtac      ;  und Inhalt ausgeben
07D2  DD 36 00 BF         ld (ix+0),10111111b ; MINUS-Zeichen
07D6  C9                  ret
07D7                      aus4:
07D7  FE 04               cp 4
07D9  20 01               jr nz,aus5      ; 4= dunkelmode, aber ohne CLEAR
07DB  C9                  ret             ; keine Anzeige wird durchgefuehrt,
                                          ; um Tests mit Displayroutinen
                                          ; durchfuehren zu koennen
07DC                      aus5:
07DC  AF                  xor a
07DD  32 8012             ld (ausmode),a
07E0  18 89               jr ausgabe      ; nochmals mit a=0


07E2                      regaus::        ; a = Registercode 0..anzreg-1
                                          ; hl ist danach die Modifizieradresse
                                          ; Text wird schon ausgegeben
07E2  FE 0D               cp anzreg       ; >=anzreg dann Fehler
```

```
07E4  38 01              jr c,regla     ; < dann ok
07E6  AF                 xor a          ; sonst ruecksetzen
07E7                    regla:
07E7  F5                 push af
07E8  CD 02B3            call clear
07EB  F1                 pop af
07EC  21 0810            ld hl,regtab   ; Tabelle
07EF  4F                 ld c,a
07F0  06 00              ld b,0         ; 5*Index+regtab
07F2  09                 add hl,bc
07F3  09                 add hl,bc
07F4  09                 add hl,bc
07F5  09                 add hl,bc
07F6  09                 add hl,bc      ;
07F7  4E                 ld c,(hl)      ; Adresse LSB
07F8  23                 inc hl         ; Laden der Registerpaaradresse
07F9  46                 ld b,(hl)      ; Adresse MSB
07FA  DD 21 8000         ld ix,anzfeld  ; Adresse Ziel fuer den Text
07FE  23                 inc hl
07FF  7E                 ld a,(hl)      ; Name des Registers
0800  DD 77 00           ld (ix+0),a    ; ins Anzeigefeld
0803  23                 inc hl         ; uebertragen
0804  7E                 ld a,(hl)
0805  DD 77 01           ld (ix+1),a
0808  23                 inc hl
0809  7E                 ld a,(hl)
080A  DD 77 02           ld (ix+2),a
080D  69                 ld l,c
080E  60                 ld h,b         ; hl ist nun die Adresse
080F  C9                 ret            ; des Registerpaars
0810                    regtab::        ; Registertabelle adr,segmentcode
                                        ; Anzahl = anzreg
0810  801F               defw pswlsto
0812  88 8E FF           defb 10001000b,10001110b,11111111b
0815  8021               defw bcsto
0817  83 C6 FF           defb 10000011b,11000110b,11111111b
081A  8023               defw desto
081C  A1 86 FF           defb 10100001b,10000110b,11111111b
081F  8025               defw hlsto
0821  89 C7 FF           defb 10001001b,11000111b,11111111b
0824  8033               defw usersp
0826  92 8C FF           defb 10010010b,10001100b,11111111b
0829  802F               defw ixsto
082B  F9 89 FF           defb 11111001b,10001001b,11111111b
082E  8031               defw iysto
0830  F9 8D FF           defb 11111001b,10001101b,11111111b
0833  8027               defw psw2sto
0835  88 8E DF           defb 10001000b,10001110b,11011111b
0838  8029               defw bc2sto
```

```
083A  83 C6 DF          defb 10000011b,11000110b,11011111b
083D  802B              defw de2sto
083F  A1 86 DF          defb 10100001b,10000110b,11011111b
0842  802D              defw hl2sto
0844  B9 C7 DF          defb 10001001b,11000111b,11011111b
0847  8035              defw iregsto
0849  AF F9 FF          defb 10101111b,11111001b,11111111b
084C  801D              defw pcsto
084E  8C C6 FF          defb 10001100b,11000110b,11111111b

000D                    anzreg equ ($-regtab)/5



0851                    doreg::         ; Register anzeigen und modifizieren
                                        ;
0851  3E 00              ld a,0         ; Start bei Register 0
0853  32 8015            ld (ausreg),a  ;
0856                    doireg::        ; GLOBALER Einsprung mit altem Registersatz
0856  CD 07E2            call regaus    ; ausgeben auf dem Bildschirm
0859  4E                 ld c,(hl)      ; LSB
085A  23                 inc hl
085B  46                 ld b,(hl)      ; MSB
085C  E5                 push hl
085D  69                 ld l,c
085E  60                 ld h,b         ; Wert bearbeiten
085F  DD 21 8004         ld ix,anzfeld+4        ; modifizieren
0863  CD 0218            call gethl
0866  FE 47              cp befex       ; Abbruch moeglich, Z-Flag setzen
0868  4D                 ld c,l
0869  44                 ld b,h         ; und zurueckspeichern
086A  E1                 pop hl
086B  C8                 ret z          ; Z-Flag ist noch gueltig
086C  70                 ld (hl),b
086D  2B                 dec hl
086E  71                 ld (hl),c      ; an alten Platz
086F  FE 5E              cp minusex
0871  20 12              jr nz,do20r
0873  3A 8015            ld a,(ausreg)
0876  3D                 dec a
0877  32 8015            ld (ausreg),a
087A  B7                 or a
087B  F2 0856            jp p,doireg    ;>=0 ist ok
087E  3E 0C              ld a,anzreg-1  ; da Bereich 0..anzreg-1
0880  32 8015            ld (ausreg),a
0883  18 D1              jr doireg      ; auf max-1 stellen

0885                    do20r:
0885  FE 57              cp crex        ; CR arbeitet wie PLUS
```

```
0887  28 03           jr z,do21r
0889  FE 5D           cp plusex      ; Wenn PLUS erneut gedrueckt wird, so
088B  C0              ret nz         ; wird das naechste Register angewaehlt
088C                  do21r:
088C  3A 8015         ld a,(ausreg)  ; sonst immer das zuletzt verwendete
088F  3C              inc a
0890  32 8015         ld (ausreg),a  ; neuer Index
0893  FE 0D           cp anzreg      ; muss < als ANZREG sein
0895  38 BF           jr c,do1reg    ; mehr als es Register gibt
0897  AF              xor a          ; dann wieder bei erstem Register
0898  32 8015         ld (ausreg),a  ; anfangen
089B  18 B9           jr do1reg


089D                  dobreak::      ; Breakpoints setzen, dies sind
                                     ; Haltepunkte, bei denen die
                                     ; Ausfuehrung von Programmen
                                     ; unterbrochen wird, damit
                                     ; lassen sich also Programme gut
                                     ; testen
089D  06 01           ld b,1         ; Startwert fuer Index
089F  21 8041         ld hl,brk1     ; Break-Tabelle
08A2                  do1blp:        ; Hauptschleife
08A2  C5              push bc
08A3  CD 02B3         call clear     ; Loeschen der Anzeige zuerst
08A6  DD 21 8000      ld ix,anzfeld
08AA  C1              pop bc
08AB  78              ld a,b         ; Index ausgeben
08AC  C5              push bc
08AD  CD 012A         call prtac
08B0  DD 21 8004      ld ix,anzfeld+4
08B4  5E              ld e,(hl)      ; lsb Adresse
08B5  23              inc hl
08B6  56              ld d,(hl)      ; msb Adresse
08B7  2B              dec hl         ; wieder zurueck
08B8  EB              ex de,hl       ; der Zwischenspeicher
08B9  D5              push de
08BA  CD 0218         call gethl     ; alten Wert modifizieren
08BD  D1              pop de
08BE  C1              pop bc
08BF  EB              ex de,hl       ;
08C0  FE 47           cp befex
08C2  C8              ret z          ; Abbruch
08C3  73              ld (hl),e      ; lsb ablegen
08C4  23              inc hl
08C5  72              ld (hl),d      ; dann msb
08C6  23              inc hl
08C7  23              inc hl         ; und zurueck
08C8  04              inc b          ;1,2,3 zulaessig
```

```
08C9  78              ld a,b
08CA  FE 04           cp 4          ;
08CC  20 D4           jr nz,doiblp  ; nach 3 Mal Abbruch
08CE  C9              ret


08CF                  doios::       ; IO setzen
08CF  CD 02B3          call clear   ; erst Mal Anzeige loeschen
08D2  DD 21 8000       ld ix,anzfeld ; Eingabe der Adresse
08D6  0E 00            ld c,0       ; ein Byte lesen
08D8  CD 01ED          call getc    ; Abschlusszeichen in A
08DB  FE 47            cp befex
08DD  C8               ret z        ; Abbruch wenn BEF-Taste gedrueckt wird
08DE  1E 00            ld e,0       ; Data=0 ist Defaultwert
08E0                  dolios:       ; Wiederhole
08E0  C5               push bc      ; sonst Wert in C merken
08E1  DD 21 8006       ld ix,anzfeld+6      ; dann Datenwert holen
08E5  4B               ld c,e       ; Default Daten-Wert
08E6  CD 01ED          call getc    ; neuen Wert einlesen
08E9  59               ld e,c       ; e=Datenwert
08EA  C1               pop bc       ; c=Adresse
08EB  FE 47            cp befex
08ED  C8               ret z        ; Abbruch bei BEF-Eingabe
08EE  ED 59            out (c),e    ; und Datenwert an Peripherie ausgeben
08F0  FE 57            cp crex      ; nochmals wiederholen mit gleicher Adresse
08F2  28 EC            jr z,dolios  ; Wenn IOS gedrueckt wird, e=alter Datenwert
08F4  C9               ret          ; sonst Ende der Routine


08F5                  doiol::       ; lesen von IO
08F5  CD 02B3          call clear   ; Anzeigefeld loeschen
08F8  DD 21 8000       ld ix,anzfeld ; und Adresse holen
08FC  0E 00            ld c,0
08FE  16 00            ld d,0       ; Sedezimaler Anzeigemode
0900  CD 01ED          call getc    ; ist ein Byte in C
0903  FE 47            cp befex
0905  C8               ret z        ; Abbruch bei BEF-Taste
0906                  dolpiol:      ; Schleife fuer Einzelausgabe
0906  ED 78            in a,(c)     ; Datenwert holen
0908  CD 0937          call iodataus ; Ausgaberoutine, Datenwert in E zusaetzlich
090B                  doOlp:        ; Befehl eingeben
090B  CD 00BB          call holetaste ; nun Abbruch, oder Dauer, oder Formatumschaltung
                                     ; oder einfache Wiederholung
090E  FE 47            cp befex
0910  C8               ret z        ; Abbruch
0911  FE 57            cp crex      ; einfache Wiederholung mit gleicher Adresse
0913  28 F1            jr z,dolpiol
0915  FE 4B            cp startex   ; Dauerfunktion
0917  20 11            jr nz,doaOiol
```

```
0919                    dob0iol:        ; Schleifeneingang
0919    ED 78            in a,(c)       ; Datenwert holen
091B    CD 0937          call iodataus  ; einlesen und ausgeben
091E    CD 0069          call anzeige   ; und auch anzeigen
0921    FE 47            cp befex       ; Abbruch
0923    C8               ret z          ; Achtung, startex bleibt zunaechst gedrueckt
0924    FE 57            cp crex        ; bei CR-Eingabe Abbruch
0926    28 E3            jr z,do0lp
0928    18 EF            jr dob0iol     ; wiederholen, wenn keine Taste da, sonst
092A                    doa0iol:
092A    FE 4E            cp optex       ; Optionsumschaltung
092C    C0               ret nz         ; sonst Abbruch
092D    7A               ld a,d         ; 0,1
092E    EE 01            xor 1
0930    57               ld d,a
0931    7B               ld a,e         ; Datenwert holen
0932    CD 0937          call iodataus  ; aktualisieren
0935    18 D4            jr do0lp       ; mit neuem Mode zurueck


0937                    iodataus::      ; Akku=IO-Datenwert
0937    5F               ld e,a         ; merken in E, dort bleibt er fuer Aufrufer
0938    7A               ld a,d         ; d ist Umschaltmerker 0=hex(sedezimal) 1=binaer
0939    B7               or a
093A    20 14            jr nz,doliol
093C    CD 02B3           call clear    ; Anzeigefeld loeschen
093F    DD 21 8000        ld ix,anzfeld
0943    79                ld a,c
0944    CD 012A           call prtac    ; Adresse ausgeben und Datenwert
0947    DD 21 8006        ld ix,anzfeld+6 ; Datenfeld ganz rechts
094B    7B                ld a,e        ; und Datenwert holen
094C    CD 012A           call prtac    ; und in sedezimaler Form ausgeben
094F    C9                ret
0950                    doliol:         ; Binaerausgabe, nur Datenwert
0950    CD 02B3          call clear     ; Anzeige loeschen zuerst
0953    7B               ld a,e         ; Datenwert holen
0954    DD 21 8000       ld ix,anzfeld  ; Ziel
0958    CD 014E          call prtbin    ; Ausgabe binaer
095B    C9               ret


095C                    dofuell::       ; Speicherbereiche mit einem
                                        ; Wert fuellen
095C    21 0000          ld hl,0        ; Default Wert
095F    CD 0266          call getvon    ; Startadresse
0962    FE 47            cp befex       ; Abbruch
0964    C8               ret z
0965    E5               push hl        ; merken
0966    21 0000          ld hl,0        ; Default
```

```
0969  CD 0272          call getbis    ;
096C  D1               pop de         ; hl=bis de=von
096D  FE 47            cp befex       ; Abbruch auch hier moeglich
096F  C8               ret z
0970  E5               push hl
0971  21 0F22          ld hl,datamsg  ; nun Datenwert holen
0974  CD 02A6          call print
0977  E1               pop hl
0978  DD 21 8006       ld ix,anzfeld+6
097C  0E 00            ld c,0         ; Default Wert
097E  CD 01ED          call getc      ; ins c-Register
0981  EB               ex de,hl       ; hl=von de=bis
0982                   dolfuel:       ; und ausfuehren, Stop wenn Speicherfehler
0982  79               ld a,c         ; Datenwert holen
0983  77               ld (hl),a      ; Abspeichern
0984  BE               cp (hl)        ; und pruefen ob dort angekommen
0985  20 0B            jr nz,doferr   ; Fehler aufgetreten
0987  E5               push hl
0988  AF               xor a
0989  ED 52            sbc hl,de      ; test ob ENDE=AKTUELL
098B  7C               ld a,h         ; wenn ja, dann
098C  B5               or l           ; Zero-Flag setzen.
098D  E1               pop hl         ; alte Werte wieder zurueck, Flag bleibt
098E  C8               ret z          ; Abbruch wenn Endadresse erreicht.
098F  23               inc hl         ; sonst naechste Adresse anwaehlen
0990  18 F0            jr dolfuel     ; und alles wiederholen
0992                   doferr:
0992  E5               push hl
0993  21 0F2A          ld hl,errmsg   ; Fehler im Speicher
0996  CD 02A6          call print     ; z.B. kein RAM da.
0999  DD 21 8004       ld ix,anzfeld+4
099D  E1               pop hl         ; Fehleradresse ausgeben
099E  CD 0145          call prthl
09A1  CD 00BB          call holetaste ; warten auf Eingabe,
09A4  C9               ret            ; dann erst zurueck


09A5                   domve::        ; Speicherbereiche verschieben
09A5  21 0000          ld hl,0        ; Default Adressen
09A8  CD 0266          call getvon    ; hl=Adresse "von"
09AB  FE 47            cp befex       ; Abbruch bei Taste BEF
09AD  C8               ret z
09AE  E5               push hl        ; "von"-Adresse merken
09AF  CD 0272          call getbis    ; Endadresse, alte adr verwenden
09B2  D1               pop de         ; "von" in de
09B3  FE 47            cp befex       ; Abbruch
09B5  C8               ret z          ; moeglich
09B6  E5               push hl        ; "bis" merken
09B7  CD 026C          call getnach   ; Ziel-Adresse, alte adr verwenden
```

```
09BA   C1              pop bc          ; hl=nach de=von bc=bis
09BB   FE 47           cp befex
09BD   C8              ret z           ; und Abbruch moeglich
09BE   E5              push hl
09BF   AF              xor a
09C0   ED 52           sbc hl,de       ; untersuchen ob nach>=von
09C2   E1              pop hl          ; nur Flags setzen, aber nicht Minus abfragen
09C3   DA 09D1         jp c,useup      ; ldir verwendbar, wenn nach<von
                                       ; sonst lddr, um Datenverlust zu vermeiden
09C6   CD 09D8         call setupld    ; berechnen hl=von, de=nach, bc=0..n Anzahl
09C9   09              add hl,bc       ; da rueckwaerts transportiert wird,
09CA   EB              ex de,hl        ; muss hier noch die Laenge addiert werden
09CB   09              add hl,bc
09CC   EB              ex de,hl        ; hl=Endadresse, de=Zielende, bc=0..n Anzahl
09CD   03              inc bc          ; Anzahl=1..n+1
09CE   ED B8           lddr            ; und transportieren
09D0   C9              ret

09D1                   useup:          ; mit ldir  schieben
09D1   CD 09D8         call setupld    ; berechnen hl=von, de=nach, bc=0..n Anzahl
09D4   03              inc bc          ; Anzahl =1..n+1
09D5   ED B0           ldir            ; Ausfuehren ohne Speichervergleich
09D7   C9              ret

09D8                   setupld:        ; umrechnen
09D8   E5              push hl         ; Ziel
09D9   60              ld h,b          ; hl wird "bis"
09DA   69              ld l,c
09DB   AF              xor a
09DC   ED 52           sbc hl,de       ; bis-von = 0..n Anzahl-1
09DE   EB              ex de,hl        ; hl=VON-Adresse
09DF   4B              ld c,e
09E0   42              ld b,d          ; Anzahl
09E1   D1              pop de          ; HL=VON , DE=NACH(Ziel) BC=Anzahl 0..n
09E2   C9              ret



                       ;
                       ; Kassettenformat
                       ;
                       ; FF FF FF FF FF 00 : startadrh startadrl endadrh endadrl
                       ; data ..... data  checkh checkl
                       ;

09E3                   dospe::         ; Speichern auf Kassette mit CAS-Baugruppe
09E3   2A 8016         ld hl,(startadr) ; Default Wert verwenden
09E6   CD 0266         call getvon     ; hl=VON-Adresse
09E9   22 8016         ld (startadr),hl       ; ist auch Startadresse
```

```
09EC   FE 47            cp befex       ; Abbruch moeglich
09EE   C8               ret z          ; mit Taste BEF
09EF   2A 8018          ld hl,(endadr) ; auch Default Wert verwenden
09F2   CD 0272          call getbis    ; hl=BIS-Adresse
09F5   22 8018          ld (endadr),hl ; bis dorthin, incl., abspeichern
09F8   FE 47            cp befex       ; auch hier noch Abbruch moeglich
09FA   C8               ret z          ; mit der Taste BEF
09FB   06 14            ld b,20        ; erst Mal einen Vorspann ausgeben,
09FD                   do1spe:
09FD   0E FF            ld c,0ffh      ; um CAS bei der Wiedergabe
09FF   CD 0290          call poo       ; zu synchronisieren
0A02   10 F9            djnz do1spe    ;
0A04   0E 00            ld c,0         ; dann eine Kennung,
0A06   CD 0290          call poo       ; um eine Abfrage zu ermoeglichen
0A09   0E 3A            ld c,':'       ; und nochmals eine Zeichen,
0A0B   CD 0290          call poo       ; damit Format wie GRUNDZ80-Format aussieht
0A0E   ED 5B 8016       ld de,(startadr) ; de=Startadresse
0A12   21 0000          ld hl,0        ; Pruefsumme in HL bilden
0A15   19               add hl,de      ; inclusive Adressteile
0A16   4A               ld c,d         ; MSB Startadresse
0A17   CD 0290          call poo
0A1A   4B               ld c,e         ; LSB Startadresse
0A1B   CD 0290          call poo
0A1E   ED 5B 8018       ld de,(endadr) ; dann auch mit Endadresse
0A22   19               add hl,de      ; zuerst Pruefsumme bilden,
0A23   4A               ld c,d         ; dann erst MSB ausgeben
0A24   CD 0290          call poo
0A27   4B               ld c,e         ; und dann LSB
0A28   CD 0290          call poo
0A2B   ED 5B 8016       ld de,(startadr) ; nun Eingang zur Schleife de=Start
0A2F                   do2spe:
0A2F   1A               ld a,(de)      ; Datenwert laden
0A30   4F               ld c,a
0A31   06 00            ld b,0         ; zuerst zur Pruefsumme addieren
0A33   09               add hl,bc
0A34   CD 0290          call poo       ; und dann auch ausgeben
0A37   E5               push hl        ; Pruefsumme retten
0A38   2A 8018          ld hl,(endadr) ; Test ob Endadresse erreicht ist
0A3B   AF               xor a
0A3C   ED 52            sbc hl,de
0A3E   7C               ld a,h         ; end-start
0A3F   B5               or l           ; =0 dann Ende erreicht
0A40   E1               pop hl         ; Flags bleiben erhalten nach POP
0A41   28 03            jr z,do3spe    ; ja=0, dann fertig
0A43   13               inc de         ; sonst Startadresse erhoehen
0A44   18 E9            jr do2spe      ; und naechstes Byte ausgeben
0A46                   do3spe:        ; jetzt noch Pruefsumme ausgeben
0A46   4C               ld c,h         ; erst MSB der Pruefsumme
0A47   CD 0290          call poo
```

```
0A4A   4D              ld c,l          ; und dann LSB
0A4B   CD 0290         call poo
0A4E   C9              ret             ; Aufzeichnung beendet


0A4F                   doprf::         ; Nur Prueflesen
0A4F   3E 01           ld a,1          ; Test, ob Daten identisch sind
0A51   32 801C         ld (merkcas),a
0A54   18 04           jr lad1         ;

0A56                   dolad::         ; Laden von der Kassette
0A56   AF              xor a
0A57   32 801C         ld (merkcas),a ; laden

0A5A                   lad1::          ; Einsprung
0A5A   CD 029A         call getri      ; erst muss FF erkannt sein
0A5D   FE FF           cp 0ffh
0A5F   20 F9           jr nz,lad1      ; dann folgt 00
0A61                   lad0:
0A61   CD 029A         call getri      ; ff ff ff 00 :   Uebergang 00 : interessant
0A64   FE FF           cp 0ffh         ; wenn FF dann warten
0A66   28 F9           jr z,lad0       ; auf 0
0A68   FE 00           cp 0            ; wenn aber keine 0 folgt, dann
0A6A   20 EE           jr nz,lad1      ; von Anfang an nochmal
0A6C   CD 029A         call getri      ; und dann :
0A6F   FE 3A           cp ':'          ; sonst zurueck zum Anfang
0A71   20 E7           jr nz,lad1      ; andernfalls ist Kennung
                                       ; gueltig
0A73   21 0000         ld hl,0         ; Pruefsumme auf 0 setzen
0A76   CD 029A         call getri      ; Startadresse MSB holen
0A79   57              ld d,a
0A7A   CD 029A         call getri      ; und LSB
0A7D   5F              ld e,a
0A7E   ED 53 B016      ld (startadr),de ; und Startadresse merken
0A82   19              add hl,de       ; und Pruefsumme bilden
0A83   D5              push de         ; Startadresse merken
0A84   CD 029A         call getri      ; dann Endadresse holen
0A87   57              ld d,a          ; erst MSB
0A88   CD 029A         call getri
0A8B   5F              ld e,a          ; dann LSB
0A8C   ED 53 B018      ld (endadr),de ; ebenso Endadresse
0A90   19              add hl,de       ; und auch in die Pruefsumme
0A91   D1              pop de          ; jetzt Daten einlesen
0A92                   do1lad:         ; in einer Schleife
0A92   CD 029A         call getri      ; Byte holen
0A95   4F              ld c,a          ; und merken
0A96   06 00           ld b,0          ; zuerst Pruefsumme berechnen
0A98   09              add hl,bc
0A99   3A 801C         ld a,(merkcas) ; pruefen oder laden
```

```
0A9C   B7              or a
0A9D   20 02           jr nz,do2lad
0A9F   79              ld a,c
0AA0   12              ld (de),a       ; bei laden zuvor abspeichern
0AA1                   do2lad:
0AA1   1A              ld a,(de)       ; und dann Test, ob angekommen
0AA2   B9              cp c
0AA3   20 2F           jr nz,errprf
0AA5   E5              push hl         ; Test, ob Ende erreicht
0AA6   2A 8018         ld hl,(endadr)
0AA9   AF              xor a           ; dazu end-aktuell bilden
0AAA   ED 52           sbc hl,de
0AAC   7D              ld a,l
0AAD   B4              or h            ; =0, dann fertig
0AAE   E1              pop hl          ; Z-Flag bleibt erhalten
0AAF   28 03           jr z,do3lad     ; Ende dann
0AB1   13              inc de          ; sonst naechste Adresse nehmen
0AB2   18 DE           jr do1lad       ; und wiederholen, von vorne
0AB4                   do3lad:         ; fertig
0AB4   CD 029A         call getri      ; msb Pruefsumme
0AB7   BC              cp h            ; muss uebereinstimmen,
0AB8   20 2C           jr nz,errsum    ; sonst Pruefsummenfehler
0ABA   CD 029A         call getri      ; und auch LSB
0ABD   BD              cp l
0ABE   20 26           jr nz,errsum    ; ebenfalls
0AC0   21 0F32         ld hl,stmsg
0AC3   CD 02A6         call print
0AC6   DD 21 8004      ld ix,anzfeld+4
0ACA   2A 8016         ld hl,(startadr)
0ACD   CD 0145         call prthl      ; Ausgabe ST= Startadresse
0AD0   CD 00BB         call holetaste ; und dann noch auf Tastendruck warten
0AD3   C9              ret             ; Fertig, geladen


0AD4                   errprf:         ; de=adresse, Vergleichsfehler
0AD4   21 0F3A         ld hl,prfmsg    ; wird auch vom PROMMER verwendet
0AD7   CD 02A6         call print
0ADA   EB              ex de,hl        ; hl=Fehleradresse
0ADB   DD 21 8004      ld ix,anzfeld+4
0ADF   CD 0145         call prthl
0AE2   CD 00BB         call holetaste ; und warten bis Eingabe da
0AE5   C9              ret


0AE6                   errsum:
0AE6   21 0F42         ld hl,chkmsg    ; Pruefsummenfehler
0AE9   CD 02A6         call print      ; CHECKSUM ERROR
0AEC   CD 00BB         call holetaste ;
0AEF   C9              ret
```

```
                        ;
                        ; Eprom Programmierer
                        ; zusammen mit der PROMMER-Baugruppe
                        ;


0080                    promd   equ     80h     ; Datenport bidirektional
0081                    proma1  equ     81h     ; lsb Adresse beim schreiben
0082                    proma2  equ     82h     ; msb Adresse

                        ;
                        ; proma2
                        ;
                        ;7      6       5       4       3       2       1       0
                        ;enable led     trigger a12     a11     a10     19      a8
                        ;
                        ; beim Lesen aus dem Eprom
                        ; 0     1       0       anlegen an Bit 7..5
                        ;
                        ; beim Schreiben in das Eprom
                        ; 1     0       0       erster Schritt
                        ; 1     0       1       Trigger ausloesen
                        ; 1     0       0       am Schluss
                        ;
                        ; proma1  lesen
                        ;
                        ; x     x       x       x       x       x       x       busy
                        ; busy=1 dann wird gerade programmiert, sonst=0
                        ;
                        ; proma2 schreiben ist A7..A0
                        ;
                        ; promd lesen,schreiben ist D7..D0
                        ;

0AF0                    prominit::      ; in Lese-Stellung bringen
0AF0    3E 40            ld a,01000000b ; LED ausschalten
0AF2    D3 82            out (proma2),a
0AF4    C9               ret

0AF5                    doprl::         ; Eprom-Inhalt in den Speicher
                                        ; laden
0AF5    3E 40            ld a,01000000b ; LED ausschalten, Lese-Mode einstellen
0AF7    D3 82            out (proma2),a ;
0AF9    21 0000          ld hl,0        ; 0=Standart VON-Adresse
0AFC    CD 0266          call getvon    ; "von" ist die Adresse im Eprom
0AFF    FE 47            cp befex       ; Abbruch moeglich
0B01    C8               ret z
0B02    22 8016          ld (startadr),hl
0B05    21 07FF          ld hl,07ffh    ; 2K Laenge als Default nehmen
```

```
0B08  CD 0272          call getbis    ; ist 0..7ffh im Eprom als Quelle
0B0B  FE 47            cp befex
0B0D  C8               ret z          ; Abbruch moeglich
0B0E  22 8018          ld (endadr),hl ; Endadresse
0B11  2A 801D          ld hl,(pcsto)  ; Zieladresse im RAM waehlen, default
0B14  CD 026C          call getnach   ;
0B17  FE 47            cp befex       ; Abbruch moeglich
0B19  C8               ret z
0B1A  22 801A          ld (zieladr),hl  ; ist Ziel im RAM
0B1D  2A 8016          ld hl,(startadr) ; Start-Adresse
0B20  ED 5B 8018       ld de,(endadr)   ; Ende-Adresse
0B24  DD 2A 801A       ld ix,(zieladr)  ; Ziel-Adresse
0B28                   doiprl:          ; Wiederhole, bis alles gelesen
0B28  7D               ld a,l           ; LSB Adresse einstellen
0B29  D3 81            out (proma1),a ; und ablegen
0B2B  7C               ld a,h           ; dann MSB und Befehlscode
0B2C  E6 1F            and 00011111b  ;
0B2E  F6 40            or  01000000b  ; Lese-Befehl
0B30  D3 82            out (proma2),a ; ausgeben
0B32  DB 80            in a,(promd)   ; Daten vom Eprom einlesen
0B34  DD 77 00         ld (ix+0),a    ; und im Speicher ablegen
0B37  DD BE 00         cp (ix+0)      ; und pruefen, ob erfolgreich
0B3A  20 24            jr nz,errlprf  ;
0B3C  E5               push hl
0B3D  AF               xor a
0B3E  ED 52            sbc hl,de
0B40  7C               ld a,h
0B41  B5               or l           ; pruefen, ob endadr=startadr
0B42  E1               pop hl         ; Z-Flag bleibt erhalten
0B43  28 05            jr z,fprl      ; =0, dann lesen beendet
0B45  DD 23            inc ix         ; Ziel+1
0B47  23               inc hl         ; Start+1
0B48  18 DE            jr doiprl      ; und weiter wiederholen
0B4A                   fprl:
0B4A  2A 801A          ld hl,(zieladr) ; Startadresse defaulten
0B4D  ED 4B 8016       ld bc,(startadr) ; und alte adr merken
0B51  22 8016          ld (startadr),hl ; ist Ram-Zieladresse
0B54  ED 5B 801B       ld de,(endadr) ; + Endadresse ist neue
0B58  19               add hl,de      ;
0B59  AF               xor a
0B5A  ED 42            sbc hl,bc      ; -Anfang im Eprom ergibt Laenge
0B5C  22 8018          ld (endadr),hl ; Endadresse, z.B. fuer CAS-Schreiben
0B5F  C9               ret


0B60                   errlprf::      ; Speichervergleichsfehler
0B60  DD E5             push ix       ; in errprf wird die
0B62  D1                pop de        ; Fehleradresse in DE verlangt
0B63  C3 0AD4           jp errprf     ; weiter wie bei CAS
```

```
0B66                    doprp::         ; Eprom programmieren

0B66    3E 40            ld a,01000000b ; Prommer auf "lesen" stellen zunaechst
0B68    D3 82            out (proma2),a ; und dann Parameter einlesen
0B6A    2A 8016          ld hl,(startadr)       ; Start im RAM
0B6D    CD 0266          call getvon    ; neu einstellen
0B70    FE 47            cp befex
0B72    C8               ret z          ; Abbruch moeglich
0B73    22 8016          ld (startadr),hl
0B76    2A 8018          ld hl,(endadr)
0B79    CD 0272          call getbis    ; Ende im RAM-Speicher
0B7C    FE 47            cp befex       ; Abbruch moeglich
0B7E    C8               ret z
0B7F    22 8018          ld (endadr),hl
0B82    21 0000          ld hl,0        ; Default Ziel=0
0B85    CD 026C          call getnach   ; Ziel im Eprom
0B88    FE 47            cp befex       ; Abbruch moeglich
0B8A    C8               ret z
0B8B    22 801A          ld (zieladr),hl ; auch merken, dann starten
0B8E    21 0F4A          ld hl,begmsg   ; CR=Beginn
0B91    CD 02A6          call print
0B94    CD 00BB          call holetaste
0B97    FE 57            cp crex
0B99    C0               ret nz         ; nur wenn CR eingegeben, erfolgt Start
0B9A    CD 02B3          call clear     ; Anzeigefeld loeschen
0B9D    2A 8016          ld hl,(startadr) ; von
0BA0    ED 5B 8018       ld de,(endadr)   ; bis
0BA4    ED 4B 801A       ld bc,(zieladr)  ; nach
0BA8                    prlp1:          ; Hauptschleife  1
0BA8    3E 80            ld a,10000000b ; Tri-State einschalten und Progspg
0BAA    D3 82            out (proma2),a ;
0BAC    7E               ld a,(hl)      ; Datenwert ausgeben an Port
0BAD    D3 80            out (promd),a  ;
0BAF    79               ld a,c         ; lsb - Adresse
0BB0    D3 81            out (promal),a ; einstellen
0BB2    78               ld a,b         ; msb - Adresse
0BB3    E6 1F            and 00011111b  ; Maske Adr.
0BB5    F6 80            or 10000000b   ; TRI
0BB7    D3 82            out (proma2),a ; zuerst
0BB9    F6 20            or 00100000b   ; Programmierpuls
0BBB    D3 82            out (proma2),a ; triggern
0BBD    E6 DF            and 11011111b  ; und neutral
0BBF    E5               push hl        ; nun warten bis programmiert
0BC0    D5               push de        ; und anzeigen der aktuellen
0BC1    C5               push bc        ; Adresse
0BC2    DD 21 8000       ld ix,anzfeld  ; Adresse ausgeben
0BC6    CD 0145          call prthl     ;
0BC9    7E               ld a,(hl)      ; und Datenwert dazu
0BCA    DD 21 8006       ld ix,anzfeld+6
```

```
0BCE  CD 012A          call prtac     ;
0BD1                   prlp01:        ; Warteschleife
0BD1  CD 0069           call anzeige  ; 1ms und Ausgabe auf Segment
0BD4  DB 81             in a,(proma1) ; Busy abfragen
0BD6  E6 01             and 1         ; =1, dann wird noch programmiert
0BD8  20 F7             jr nz,prlp01  ; also warten bis 0 geworden
                                      ; dann mindestens 10ms warten,
                                      ; bis Monoflop wieder bereit ist,
                                      ; da C sich umladen muss 80% Zyklus
0BDA  C1                pop bc
0BDB  C5                push bc       ;
0BDC  78                ld a,b        ; Adresse, MCS
0BDD  E6 1F             and 00011111b ; im Lese-Mode mit LED=an
0BDF  D3 82             out (proma2),a ; fuer Prueflesen einstellen
0BE1  06 0A             ld b,10       ;
0BE3                    prlp02:       ; Die Anzeigeroutine dient gleichzeitig
0BE3  CD 0069           call anzeige  ; als Zeitschleife,
0BE6  10 FB             djnz prlp02
0BE8  C1                pop bc
0BE9  D1                pop de
0BEA  E1                pop hl
0BEB  DB 80             in a,(promd)  ; Laden
0BED  BE                cp (hl)       ; muss gleich sein, sonst Fehler
0BEE  C2 0C34           jp nz,err2prf ; FEHLER VERGLEICH
0BF1  E5                push hl
0BF2  AF                xor a
0BF3  ED 52             sbc hl,de
0BF5  7D                ld a,l        ; Start=Ende dann stopp
0BF6  B4                or h          ; dazu auf 0 pruefen
0BF7  E1                pop hl
0BF8  28 05             jr z,prg1fin  ; und Endebehandlung
0BFA  23                inc hl        ; Quelle+1
0BFB  03                inc bc        ; Ziel+1
0BFC  C3 0BA8           jp prlp1


0BFF                   prg1fin:       ; nun nochmal alles vergleichen
0BFF  3E 40             ld a,01000000b ; LED ausschalten beim Lese-Vorgang
0C01  D3 82             out (proma2),a
0C03  2A 8016           ld hl,(startadr)      ;
0C06  ED 5B 8018        ld de,(endadr)
0C0A  ED 4B 801A        ld bc,(zieladr)       ; Parameter wieder laden
0C0E                   prglp4:
0C0E  79                ld a,c        ; LSB Adresse einstellen
0C0F  D3 81             out (proma1),a
0C11  78                ld a,b        ; MSB Adresse
0C12  E6 1F             and 00011111b
0C14  F6 40             or 01000000b  ; mit LED an
0C16  D3 82             out (proma2),a
```

```
0C18  DB 80           in a,(promd)   ; Datenport lesen
0C1A  BE              cp (hl)        ; und Vergleich mit Quelle durchfuehren
0C1B  20 17           jr nz,err2prf  ; wenn nicht identisch, Fehler VERGLEICH
0C1D  E5              push hl
0C1E  AF              xor a          ; Nun auf Endadresse pruefen
0C1F  ED 52           sbc hl,de
0C21  7C              ld a,h
0C22  B5              or l           ; =0 dann fertig
0C23  E1              pop hl         ; dann endadr=startadr
0C24  28 04           jr z,prgfin
0C26  23              inc hl         ; dann naechste Adresse einstellen
0C27  03              inc bc
0C28  18 E4           jr prglp4
0C2A                  prgfin:
0C2A  21 0F52         ld hl,burnmsg  ; ok uebertragen
0C2D  CD 02A6         call print
0C30  CD 00BB         call holetaste ; dann erst zurueck
0C33  C9              ret


0C34                  err2prf:       ; wie CAS, VERGLEICHSFEHLER
0C34  E5              push hl
0C35  D1              pop de         ; de=Adresse
0C36  3E 40           ld a,01000000b,; LED abstellen und Lese-Mode
0C38  D3 82           out (proma2),a
0C3A  C3 0AD4         jp errprf      ; und ausgeben


                      ;
                      ; Ende Prom Programmierer
                      ;


0C3D                  dovgl::        ; Speicherbereiche vergleichen
0C3D  2A 8016         ld hl,(startadr)
0C40  CD 0266         call getvon    ; Startadresse holen
0C43  FE 47           cp befex       ; Abbruch moeglich
0C45  C8              ret z
0C46  22 8016         ld (startadr),hl
0C49  2A 8018         ld hl,(endadr) ; Default alter Wert
0C4C  CD 0272         call getbis    ; Bis Endeadresse
0C4F  FE 47           cp befex
0C51  C8              ret z
0C52  22 8018         ld (endadr),hl
0C55  2A 801A         ld hl,(zieladr) ; Als Eingangswert verwenden
0C58  CD 0278         call getmit    ; Vergleich "MIT"
0C5B  FE 47           cp befex       ; Abbruch ?
0C5D  C8              ret z
```

```
0C5E   22 801A            ld (zieladr),hl
0C61   2A 8016            ld hl,(startadr) ; nun durchfuehren
0C64   ED 5B 8018         ld de,(endadr)   ; Ende
0C68   ED 4B 801A         ld bc,(zieladr)  ; Vergleichsadresse
0C6C                      do1vgl:
0C6C   0A                 ld a,(bc)
0C6D   BE                 cp (hl)          ; wenn nicht gleich, dann Adresse melden
0C6E   20 0C              jr nz,do2vgl
0C70   E5                 push hl
0C71   AF                 xor a            ; nun pruefen, ob Ende erreicht
0C72   ED 52              sbc hl,de        ; end=aktuell
0C74   7C                 ld a,h
0C75   B5                 or l             ; Z-Flag setzen
0C76   E1                 pop hl           ; bleibt erhalten
0C77   C8                 ret z            ; Ende wenn Ja, ohne Meldung ok
0C78   23                 inc hl           ; Start+1
0C79   03                 inc bc           ; Mit+1
0C7A   18 F0              jr do1vgl


0C7C                      do2vgl:
0C7C   E5                 push hl
0C7D   D1                 pop de
0C7E   C3 0AD4            jp errprf        ; wie in CAS PRF xxxx melden



                          ;
                          ; Programme zum Einstellen der Hardware.
                          ; PROMMER und CAS benoetigen einen Abgleich.
                          ; Da dabei Pulsbreiten und Perioden gemessen werden
                          ; muessen, ist ein Messgeraet noetig.
                          ; Doch der Computer kann auch ein Messgeraet sein,
                          ; und daher sind in HEXMON gleich die noetigen
                          ; Programme eingebaut.

0C81                      domes1::         ; Eprom-Programmierer abgleichen
0C81   21 0000            ld hl,0          ; Messwert Start
0C84   3E 80              ld a,10000000b ; erstes Bit gibt Monoflop frei
0C86   D3 82              out (proma2),a ; LED an
0C88   3E A0              ld a,10100000b ; Triggern des Monoflops
0C8A   D3 82              out (proma2),a ; und damit den Messvorgang starten (11)
0C8C   3E 80              ld a,10000000b ; (7)
0C8E   D3 82              out (proma2),a ; und wieder zurueckschalten (11)
0C90                      do2mes1:         ; Zaehlschleife
0C90   23                 inc hl           ; 6  TK
0C91   13                 inc de           ; 6  TK fuer den Zeitangleich
0C92   DB 81              in a,(proma1)  ; 11 TK 1=Busy
0C94   E6 01              and 1            ; 7  TK
```

```
0C96   C2 0C90          jp nz,do2mes1  ; 10 TK Messchleife
                                       ; Summe=40 Zyklen
                                       ; bei 4MHz somit 10 Mikrosekunden
                                       ; Startwert 29TK Ungenauigkeit
0C99   CD 0CA3          call ausms     ; ausgeben ms-Wert
0C9C   30 E3            jr nc,domes1   ; zurueck, wenn nicht abgebrochen
0C9E   3E 40            ld a,01000000b ; LED wieder an
0CA0   D3 82            out (proma2),a ; Ende durch BEF-Taste
0CA2   C9               ret
0CA3                    ausms:         ; Ausgabe hl in MS
0CA3   CD 02B3          call clear     ; erstmal loeschen
0CA6   CB 7C            bit 7,h        ;
0CA8   20 24            jr nz,do6mes   ; <0 dann FEHLER, Abbruch
0CAA                    do3mes:        ; Ausgabe
0CAA   E5               push hl        ; Messwert
0CAB   21 0F5A          ld hl,msmsg    ; xxx.00 MS ausgeben
0CAE   CD 02A6          call print
0CB1   DD 21 8004       ld ix,anzfeld+4 ; xxx.xx
0CB5   E1               pop hl         ; Messwert ausgeben
0CB6   CD 0162          call prtdez    ; dann
0CB9   DD 21 8002       ld ix,anzfeld+2 ; Komma ausgeben
0CBD   DD CB 00 BE      res 7,(ix+0)   ;
0CC1   06 C8            ld b,200       ; mind 200ms Wert anzeigen,
0CC3                    do5mes:        ; um Flackern zu minimieren
0CC3   CD 0069          call anzeige
0CC6   FE 47            cp befex       ; Abbruch
0CC8   28 04            jr z,do6mes    ; ENDE
0CCA   10 F7            djnz do5mes
0CCC   AF               xor a
0CCD   C9               ret
0CCE                    do6mes:
0CCE   37               scf            ; mit Abbruch
0CCF   C9               ret



                        ;
                        ; bei den folgenden Messprogrammen
                        ; muss eine Leitung von Bit 7 Port 0
                        ; an die Messtelle gelegt werden
                        ;

0CD0                    domes2::       ; Periodendauer messen (CAS)
0CD0   21 0002          ld hl,2        ; Messwert auf 2 Mikrosekunden stellen,
                                       ; da konstanter Fehler =10*3.25 ys von DJNZ
                                       ;  -6.25ys fuer Starterkennung
                                       ; also 26.25ys bei 10 Schleifen,
                                       ; wegen 10 facher Genauigkeit
                                       ; ergibt sich ein Fehler von ca. 2 Einheiten
```

```
0CD3   06 0A              ld b,10         ; zehnmal durchfuehren
0CD5                      do20mes2:       ; Warten ___----____ Uebergang
0CD5   DB 00              in a,(tastin)   ;
0CD7   07                 rlca            ; in Carry
0CD8   D2 0CD5            jp nc,do20mes2  ; bis Carry da , dann erst ----___
0CDB                      do21mes2:
0CDB   DB 00              in a,(tastin)   ; 11 TK * zaehlt aber nicht dazu
0CDD   07                 rlca            ; 4  TK
0CDE   DA 0CDB            jp c,do21mes2   ; 10 TK  / Summe = 25 Zyklen
0CE1                      do2mes2:        ; Zaehlschleife
0CE1   23                 inc hl          ; 6  TK
0CE2   13                 inc de          ; 6  TK fuer den Zeitangleich
0CE3   DB 00              in a,(tastin)   ; 11 TK 1=Busy
0CE5   E6 80              and 80h         ; 7  TK
0CE7   CA 0CE1            jp z,do2mes2    ; 10 TK Messchleife ____-----
                                          ; Summe=40 Zyklen
                                          ; bei 4MHz somit 10 Mikrosekunden
0CEA                      do3mes2:        ; Zaehlschleife
0CEA   23                 inc hl          ; 6  TK
0CEB   13                 inc de          ; 6  TK fuer den Zeitangleich, b bleibt
0CEC   DB 00              in a,(tastin)   ; 11 TK 1=Busy
0CEE   E6 80              and 80h         ; 7  TK
0CF0   C2 0CEA            jp nz,do3mes2   ; 10 TK Messchleife
                                          ; Summe=40 Zyklen
                                          ; bei 4MHz somit 10 Mikrosekunden
                                          ; ---_____
0CF3   10 EC              djnz do2mes2    ; schnell wieder umgekehrt, pruefen 13TK
0CF5   CD 0D1D            call ausys      ; und anzeigen in Mikrosekunden
0CF8   30 D6              jr nc,domes2    ; dann wiederholen, bis BEF-Taste gedrueckt
0CFA   C9                 ret             ; Ende durch BEF-Taste



0CFB                      domes3::        ; 0-Puls-Dauer messen (CAS)
                                          ; hl liefert Ergebnis in Mikrosekunden ab
0CFB   21 0000            ld hl,0         ; Fehler in dieser Routine kleiner,
                                          ; da do21mes3 auch aufgerufen wird
0CFE   06 0A              ld b,10         ; zehnmal messen, um Genauigkeit
                                          ; zu erhoehen und Wert zu mitteln
0D00                      do20mes3:       ; Warten ___----____ Uebergang
0D00   DB 00              in a,(tastin)   ;
0D02   07                 rlca            ; in Carry
0D03   D2 0D00            jp nc,do20mes3  ; warten bis auf Carry, dann erst ----___
0D06                      do21mes3:
0D06   DB 00              in a,(tastin)   ; 11 TK * zaehlt aber nicht dazu
0D08   07                 rlca            ; 4  TK
0D09   DA 0D06            jp c,do21mes3   ; 10 TK  / Summe = 25 Zyklen
0D0C                      do2mes3:        ; Zaehlschleife
0D0C   23                 inc hl          ; 6  TK
```

```
0D0D   13              inc de          ; 6  TK fuer den Zeitangleich
0D0E   DB 00           in a,(tastin)   ; 11 TK 1=Busy
0D10   E6 80           and 80h         ; 7  TK
0D12   CA 0D0C         jp z,do2mes3    ; 10 TK Messchleife
                                       ; Summe=40 Zyklen
                                       ; bei 4MHz somit Mikrosekunden
0D15   10 EF           djnz do21mes3   ; ___--  erfuellt 13 TK  *
0D17   CD 0D1D         call ausys      ; Mikrosekunden Ausgabe
0D1A   30 DF           jr nc,domes3    ; weiter
0D1C   C9              ret             ; fertig


0D1D                   ausys::         ; hl=Wert in Mikrosekunden
0D1D   CD 02B3          call clear
0D20   CB 7C            bit 7,h        ; =0 dann Fehler
0D22   C2 0CCE          jp nz,do6mes   ; carry Fehlerausgange
0D25   E5               push hl
0D26   21 0F62          ld hl,ysmsg    ;
0D29   CD 02A6          call print     ; xxxxx Mikrosekunden (ys wird )
0D2C   DD 21 8004       ld ix,anzfeld+4 ;          (auf der Anzeige ausgegeben)
0D30   E1               pop hl
0D31   CD 0162          call prtdez
0D34   06 64            ld b,100       ; 100ms Ausgabe
0D36                   auslys:
0D36   CD 0069          call anzeige
0D39   FE 47            cp befex       ; Abbruch moeglich
0D3B   CA 0CCE          jp z,do6mes    ; dort CARRY setzen
0D3E   10 F6            djnz auslys    ; bis 100ms
0D40   AF               xor a          ; Ende Ausgabeteil
0D41   C9               ret



                       ;
                       ; Umrechnungshilfe
                       ;


0D42                   doconv::        ; Zahlensysteme umrechnen

0D42   CD 02B3          call clear
0D45   21 0000          ld hl,0        ; Default
0D48   DD 21 8004       ld ix,anzfeld+4
0D4C   CD 0218          call gethl     ; HEX->DEZ
0D4F   FE 47            cp befex
0D51   C8               ret z          ; Abbruchmoeglichkeit
0D52                   do10conv:
0D52   E5               push hl        ; hl merken
0D53   21 0F6A          ld hl,dezmsg
```

```
0D56  CD 02A6          call print
0D59  E1               pop hl          ; fuer Ausgabe
0D5A  DD 21 8007       ld ix,anzfeld+7 ; auf letztes Feld
0D5E  E5               push hl
0D5F  CD 0162          call prtdez
0D62  E1               pop hl          ; alter Datenwert
0D63  CD 00BB          call holetaste
0D66  FE 47            cp befex
0D68  C8               ret z
0D69  FE 4E            cp optex
0D6B  20 D5            jr nz,doconv    ; nochmals im gleichen Mode
0D6D  18 10            jr do2conv      ; alte Zahl ausgeben

0D6F                   do1conv:
0D6F  CD 02B3          call clear      ; loeschen des Anzeigefeldes
0D72  DD 21 8007       ld ix,anzfeld+7 ; auf letztes Feld
0D76  21 0000          ld hl,0         ; Default
0D79  CD 01C2          call getdez     ; DEZ->HEX
0D7C  FE 47            cp befex
0D7E  C8               ret z

0D7F                   do2conv:
0D7F  E5               push hl         ; Wert merken
0D80  21 0F72          ld hl,sedmsg
0D83  CD 02A6          call print
0D86  E1               pop hl          ; fuer Ausgabe
0D87  DD 21 8004       ld ix,anzfeld+4
0D8B  CD 0145          call prthl
0D8E  CD 00BB          call holetaste
0D91  FE 47            cp befex
0D93  C8               ret z
0D94  FE 4E            cp optex
0D96  20 D7            jr nz,do1conv   ; nochmals
0D98  18 B8            jr do10conv     ; anderer Mode



                       ; Unterprogramm fuer INIT

0D9A                   lastmem::       ; Sucht Speicherende und testet.
                                       ; Adresse der letzten verfuegbaren Zelle
                                       ; wird in HL gelegt
0D9A  21 807D          ld hl,last      ; Suche beginnt bei erster freien Zelle
0D9D                   such1:          ; zerstoert dabei den Inhalt nicht,
0D9D  46               ld b,(hl)       ; da immer der alte Wert zurueck geschrieben wird
0D9E  36 55            ld (hl),055h    ; Testmuster
0DA0  23               inc hl
0DA1  4E               ld c,(hl)       ; auch merken
```

```
0DA2  36 AA           ld (hl),0aah   ; versetzt ablegen, sonst C-Effekt (Kapazitiver
                                     ; Speicher)
0DA4  2B              dec hl
0DA5  3E 55           ld a,055h      ; der Datenwert muss auch dort
0DA7  BE              cp (hl)        ; angekommen sein
0DA8  20 1B           jr nz,suche    ; Am Ende des Speichers stoppt das Programm
0DAA  70              ld (hl),b      ; erster Wert zurueck, denn alte Speicherinhalte
0DAB  78              ld a,b         ; sollen erhalten bleiben,
0DAC  BE              cp (hl)        ; und immer wieder einen Vergleich durchfuehren
0DAD  20 16           jr nz,suche    ; damit wird das Ende zuverlaessig erkannt, aber
0DAF  23              inc hl         ; auch defekte Speicher
0DB0  3E AA           ld a,0aah
0DB2  BE              cp (hl)        ; auch Ende-Teil
0DB3  20 10           jr nz,suche
0DB5  71              ld (hl),c      ; alter Wert zurueck
0DB6  79              ld a,c
0DB7  BE              cp (hl)        ; auch hier nochmals testen,
0DB8  20 0B           jr nz,suche    ; ob Wert zurueckgeschrieben werden kann
0DBA  7C              ld a,h
0DBB  FE FF           cp 0ffh
0DBD  20 DE           jr nz,suchl
0DBF  7D              ld a,l
0DC0  FE FF           cp 0ffh        ; max ffffh., denn dort ist der Z80-
0DC2  20 D9           jr nz,suchl    ; Adressraum zu Ende.
0DC4  23              inc hl         ; auf 0, wenn letzte Zelle erreicht ist.
0DC5                 suche:          ; hl zeigt auf letzte Zelle,
0DC5  2B              dec hl         ; die noch schreibfaehig war
0DC6  C9              ret


0DC7                 init::          ; Bei Start des Eproms
                                     ; werden alle Speicherzellen
                                     ; des HEXMON-Programmes hier
                                     ; definiert
0DC7  AF              xor a
0DC8  32 8012         ld (ausmode),a ; Adr Befehl in STEP anzeigen
0DCB  32 8015         ld (ausreg),a  ; PSW ist Startregister
0DCE  32 8040         ld (brkzahl),a ; Breakpoints loeschen
0DD1  32 8011         ld (doton),a   ; Punkte aus
0DD4  21 0000         ld hl,0        ; und initialisieren
0DD7  22 8041         ld (brk1),hl
0DDA  22 8044         ld (brk2),hl
0DDD  22 8047         ld (brk3),hl
0DE0  21 8100         ld hl,8100h    ; Programstart default, hier einzige Stelle
0DE3  22 801D         ld (pcsto),hl  ; wo er gesetzt wird,
0DE6  22 8013         ld (ausadr),hl ; Default Adresse
0DE9  22 8016         ld (startadr),hl
0DEC  22 801A         ld (zieladr),hl
0DEF  22 8018         ld (endadr),hl
```

```
                                        ; Userstack ans Speicherende
0DF2    CD 0D9A                 call lastmem    ; hl-> letzte Speicherzelle
0DF5    22 8033                 ld (usersp),hl  ; dort Anwenderstack hinlegen
0DF8    CD 027E                 call casinit    ; Kassettenschnittstelle
0DFB    CD 0AF0                 call prominit   ; Eprom-Programmierer neutral stellen
0DFE    C9                      ret



                                ;****************************************
                                ;*      Hauptprogramm                   *
                                ;****************************************

0DFF                            start::         ; Start des HEXMON-Programms.
0DFF    31 807C                 ld sp,stack     ; Stackpointer ans Ende vom Speicher
0E02    21 0EDA                 ld hl,hallo     ; Hallo Meldung ausgeben nach Start
0E05    CD 02A6                 call print      ; uebertragen in anzfeld
0E08    CD 0DC7                 call init       ; Voreinstellungen durchfuehren
0E0B                            schleife::      ; Eingabeschleife
0E0B    CD 00BB                 call holetaste  ; Nun auf einen Befehl warten
0E0E    FE 5B                   cp speichex     ; Speicher modifizieren
0E10    20 06                   jr nz,schl1
0E12    CD 05A6                  call dospeicher
0E15    C3 0ED1                 jp finex
0E18                            schl1:
0E18    FE 4B                   cp startex      ; Programm starten
0E1A    20 06                   jr nz,schl2
0E1C    CD 0629                 call doprog     ; Programmstart durchfuehren
0E1F    C3 0ED1                 jp finex        ; kehrt i.A. nicht hier zurueck
0E22                            schl2:
0E22    FE 4D                   cp stepex       ; Programm im Einzelschritt durchlaufen
0E24    20 06                   jr nz,schl3
0E26    CD 06A2                 call dostep     ; und ausfuehren
0E29    C3 0ED1                 jp finex
0E2C                            schl3:
0E2C    FE 0D                   cp regex        ; Registerinhalte modifizieren
0E2E    20 06                   jr nz,schl4
0E30    CD 0851                 call doreg      ; Register Routine
0E33    C3 0ED1                 jp finex
0E36                            schl4:
0E36    FE 1B                   cp iolex        ; io lesen
0E38    20 06                   jr nz,schl5
0E3A    CD 08F5                  call doiol
0E3D    C3 0ED1                 jp finex
0E40                            schl5:
0E40    FE 0B                   cp iosex        ; io setzen
0E42    20 06                   jr nz,schl6
0E44    CD 08CF                 call doios
```

```
0E47  C3 0ED1          jp finex
0E4A                  schl6:
0E4A  FE 3D            cp ladex      ; Daten/Programm laden von Kassette
0E4C  20 06            jr nz,schl7
0E4E  CD 0A56          call dolad
0E51  C3 0ED1          jp finex
0E54                  schl7:
0E54  FE 2D            cp speex      ; Daten/Programm speichern auf Kassette
0E56  20 06            jr nz,schl8
0E58  CD 09E3          call dospe    ;
0E5B  C3 0ED1          jp finex
0E5E                  schl8:
0E5E  FE 1D            cp prfex      ; Daten/Programm pruefen mit Kassetteninhalt
0E60  20 06            jr nz,schl9
0E62  CD 0A4F          call doprf
0E65  C3 0ED1          jp finex
0E68                  schl9:
0E68  FE 0E            cp mveex      ; Bereiche verschieben
0E6A  20 06            jr nz,schl10
0E6C  CD 09A5           call domve   ;
0E6F  C3 0ED1          jp finex
0E72                  schl10:
0E72  FE 1E            cp brkex      ; Haltepunkte (breakpoints) setzen
0E74  20 06            jr nz,schl11
0E76  CD 089D          call dobreak  ; Ausfuehrung
0E79  C3 0ED1          jp finex
0E7C                  schl11:
0E7C  FE 2B            cp prpex      ; Eprom programmieren
0E7E  20 06            jr nz,schl12
0E80  CD 0B66          call doprp
0E83  C3 0ED1          jp finex
0E86                  schl12:
0E86  FE 3B            cp prlex      ; Eprom laden
0E88  20 06            jr nz,schl13
0E8A  CD 0AF5          call doprl
0E8D  C3 0ED1          jp finex
0E90                  schl13:
0E90  FE 2E            cp fillex     ; Speicherbereiche fuellen
0E92  20 06            jr nz,schl14
0E94  CD 095C          call dofuell  ; Fuellen aufrufen
0E97  C3 0ED1          jp finex
0E9A                  schl14:
0E9A  FE 3E            cp vglex      ; Bereiche vergleichen
0E9C  20 06            jr nz,schl15
0E9E  CD 0C3D          call dovgl
0EA1  C3 0ED1          jp finex
0EA4                  schl15:
0EA4  FE 07            cp meslex     ; Eprom-Puls abgleichen
```

```
0EA6  20 06           jr nz,schl16
0EA8  CD 0C81         call domes1
0EAB  C3 0ED1         jp finex
0EAE                  schl16:
0EAE  FE 17           cp mes2ex      ; Periodendauer messen (CAS)
0EB0  20 06           jr nz,schl17   ; ueber Port
0EB2  CD 0CD0         call domes2
0EB5  C3 0ED1         jp finex
0EB8                  schl17:
0EB8  FE 27           cp mes3ex      ; 0-Puls-Dauer messen (CAS)
0EBA  20 06           jr nz,schl18
0EBC  CD 0CFB         call domes3
0EBF  C3 0ED1         jp finex
0EC2                  schl18:
0EC2  FE 37           cp convex      ; Umrechnungshilfe
0EC4  20 06           jr nz,schl19
0EC6  CD 0D42         call doconv
0EC9  C3 0ED1         jp finex
0ECC                  schl19:
0ECC  21 0EEA         ld hl,Fehler   ; Fehlermeldung ausgeben
0ECF  18 03           jr finex1

0ED1                  finex:
0ED1  21 0EE2         ld hl,okmsg
0ED4                  finex1:
0ED4  CD 02A6         call print
0ED7  C3 0E0B         jp schleife    ; mit Meldung


                      ;**************************
                      ;* Texte und Meldungen    *
                      ;**************************


                      ;
                      ; Bitnummernzuordnung der Siebensegmentanzeige
                      ;
                      ;         0
                      ;       *******
                      ;    5 *     * 1
                      ;      *  6  *
                      ;       *******
                      ;    4 *     * 2
                      ;      *     *
                      ;       *******
                      ;         3     *7
                      ;
                      ;     76543210      1=Segment ist dunkel, 0=Segment ist hell
                      ;
```

```
0EDA                    hallo::
0EDA    89               defb 10001001b ; HALLO-1.1
0EDB    88               defb 10001000b
0EDC    C7               defb 11000111b
0EDD    C7               defb 11000111b
0EDE    C0               defb 11000000b
0EDF    BF               defb 10111111b
0EE0    79               defb 01111001b ;       1.
0EE1    F9               defb 11111001b ;         1

0EE2                    okmsg::
0EE2    BF               defb 10111111b ; -BEF-
0EE3    83               defb 10000011b
0EE4    86               defb 10000110b
0EE5    8E               defb 10001110b
0EE6    BF               defb 10111111b
0EE7    FF               defb 11111111b
0EE8    FF               defb 11111111b
0EE9    FF               defb 11111111b


0EEA                    fehler::
0EEA    BF               defb 10111111b ; --------
0EEB    BF               defb 10111111b
0EEC    BF               defb 10111111b
0EED    BF               defb 10111111b
0EEE    BF               defb 10111111b
0EEF    BF               defb 10111111b
0EF0    BF               defb 10111111b
0EF1    BF               defb 10111111b


0EF2                    adrmsg::        ; ADR
0EF2    88               defb 10001000b
0EF3    A1               defb 10100001b
0EF4    AF               defb 10101111b
0EF5    FF               defb 11111111b
0EF6    FF FF FF FF      defb 255,255,255,255

0EFA                    brkmsg::        ; BRE
0EFA    83               defb 10000011b
0EFB    AF               defb 10101111b
0EFC    86               defb 10000110b
0EFD    FF               defb 11111111b
0EFE    FF FF FF FF      defb 255,255,255,255

0F02                    vonmsg::
0F02    C1               defb 11000001b ; VON
0F03    C0               defb 11000000b
```

```
0F04   C8                defb 11001000b
0F05   FF                defb 11111111b
0F06   FF FF FF FF       defb 255,255,255,255

0F0A                     bismsg::
0F0A   83                defb 10000011b ; BIS
0F0B   F9                defb 11111001b
0F0C   92                defb 10010010b
0F0D   FF                defb 11111111b
0F0E   FF FF FF FF       defb 255,255,255,255

0F12                     nachmsg::       ; NAC
0F12   C8                defb 11001000b
0F13   88                defb 10001000b
0F14   C6                defb 11000110b
0F15   FF                defb 11111111b
0F16   FF FF FF FF       defb 255,255,255,255

0F1A                     mitmsg::        ; MIT
0F1A   AA                defb 10101010b
0F1B   CF                defb 11001111b
0F1C   87                defb 10000111b
0F1D   FF                defb 11111111b
0F1E   FF FF FF FF       defb 255,255,255,255

0F22                     datamsg::       ; DTA
0F22   A1                defb 10100001b
0F23   87                defb 10000111b
0F24   88                defb 10001000b
0F25   FF                defb 11111111b
0F26   FF FF FF FF       defb 255,255,255,255

0F2A                     errmsg::        ; ERR
0F2A   86                defb 10000110b
0F2B   AF                defb 10101111b
0F2C   AF                defb 10101111b
0F2D   FF                defb 11111111b
0F2E   FF FF FF FF       defb 255,255,255,255

0F32                     stmsg::         ; ST=
0F32   92                defb 10010010b
0F33   87                defb 10000111b
0F34   B7                defb 10110111b
0F35   FF                defb 11111111b
0F36   FF FF FF FF       defb 255,255,255,255

0F3A                     prfmsg::        ; PRF
0F3A   8C                defb 10001100b
0F3B   AF                defb 10101111b
```

```
0F3C   8E                defb 10001110b
0F3D   FF                defb 11111111b
0F3E   FF FF FF FF       defb 255,255,255,255

0F42                     chkmsg::        ; CHE
0F42   C6                defb 11000110b
0F43   89                defb 10001001b
0F44   86                defb 10000110b
0F45   FF                defb 11111111b
0F46   FF FF FF FF       defb 255,255,255,255

0F4A                     begmsg::        ; CR
0F4A   C6                defb 11000110b
0F4B   AF                defb 10101111b
0F4C   FF                defb 11111111b
0F4D   FF                defb 11111111b
0F4E   FF FF FF FF       defb 255,255,255,255

0F52                     burnmsg::       ; BURNED
0F52   83                defb 10000011b
0F53   E3                defb 11100011b
0F54   AF                defb 10101111b
0F55   AB                defb 10101011b
0F56   86                defb 10000110b
0F57   A1                defb 10100001b
0F58   FF FF             defb 255,255

0F5A                     msmsg::         ;    .00 MS
0F5A   FF                defb 11111111b
0F5B   FF                defb 11111111b
0F5C   7F                defb 01111111b
0F5D   C0                defb 11000000b
0F5E   C0                defb 11000000b
0F5F   FF                defb 11111111b
0F60   AA                defb 10101010b
0F61   92                defb 10010010b

0F62                     ysmsg::         ;        YS
0F62   FF FF FF FF       defb 255,255,255,255,255,255
0F66   FF FF
0F68   8D                defb 10001101b
0F69   92                defb 10010010b

0F6A                     dezmsg::        ; D
0F6A   A1                defb 10100001b
0F6B   FF FF FF FF       defb 255,255,255,255,255,255,255
0F6F   FF FF FF
```

```
0F72                    sedmsg::        ; S
0F72    92               defb 10010010b
0F73    FF FF FF FF      defb 255,255,255,255,255,255,255
0F77    FF FF FF

0F7A                    copyr:
0F7A    28 43 29 20      defb '(C) 1984 Muenchen Rolf-Dieter Klein
0F7E    31 39 38 34
0F82    20 4D 75 65
0F86    6E 63 68 65
0F8A    6E 20 52 6F
0F8E    6C 66 2D 44
0F92    69 65 74 65
0F96    72 20 4B 6C
0F9A    65 69 6E 20

0F9E                    leer::          ; Leerfeld
0F9E    FF FF FF FF      defb 255,255,255,255,255,255,255,255
0FA2    FF FF FF FF

                        ;****************************************
                        ;* Ram Speicher - Dort sind alle Variablen*
                        ;* des Programms untergebracht.          *
                        ;* Er beginnt bei der SBCII auf          *
                        ;* Adresse   8000h                       *
                        ;****************************************
                               org 8000h       ; RAM ab Adresse 8000h

                        ; Die nachfolgenden Speicherzellen koennen auch
                        ; von Anwenderprogrammen verwendet werden.
                        ; Im ANZFELD sind die einzelnen Segmentcodes
                        ; der Anzeige gespeichert. Das erste Byte ist
                        ; dabei die linke Siebensegment-Anzeige.

8000    00 00 00 00     anzfeld::       defb 0,0,0,0,0,0,0,0    ; 0..7
8004    00 00 00 00

                        ; Die nachfolgenden Speicherzellen koennen
                        ; einen Sprungbefehl aufnehmen, damit kann
                        ; man auch Interruptprogramme konstruieren

8008    00 00 00        nmiloc::        defb 0,0,0      ; Zur Aufnahme eines JP-Befehls
800B    00 00 00        intloc::        defb 0,0,0      ; Zur Aufnahme eines JP-Befehls
```

```
                        ; Die weiteren Speicherzellen sollten vom Anwenderprogramm
                        ; nicht verwendet werden

800E  00                anzpoi::  defb 0      ; Ein Index in das Feld ANZFELD
800F  00                anzsys::  defb 0      ; Merker fuer dostep, alter Wert vom System
8010  00                oldanz::  defb 0      ; Merker fuer dostep
8011  00                doton::   defb 0      ; Merker Punkt anschalten.

8012  00                ausmode:: defb 0      ; fuer STEP
8013  0000              ausadr::  defw 0      ; Adresse fuer Speicher-Ausgabe in STEP
8015  00                ausreg::  defb 0      ; Registernr. fuer Register-Ausgabe in STEP

8016  0000              startadr:: defw 0     ; Merker fuer CAS,PROMMER
8018  0000              endadr::  defw 0      ;
801A  0000              zieladr:: defw 0      ; fuer Prommer
801C  00                merkcas:: defb 0      ; Merker ob pruefen oder laden

                        ; die nachfolgenden Zellen sind fuer den Einzelschritt
                        ; reserviert.
801D  0000              pcsto::       defw 0  ; aktueller Programmzaehlerstand
801F  0000              psw1sto::     defw 0  ; Programmstatuswort
8021  0000              bcsto::       defw 0  ; Registerpaar BC
8023  0000              desto::       defw 0  ; Registerpaar DE
8025  0000              h1sto::       defw 0  ; Registerpaar HL
8027  0000              psw2sto::     defw 0  ; zweiter Registersatz Statuswort
8029  0000              bc2sto::      defw 0  ; Registerpaar BC
802B  0000              de2sto::      defw 0  ; Registerpaar DE
802D  0000              h12sto::      defw 0  ; Registerpaar HL
802F  0000              ixsto::       defw 0  ; Registerpaar IX
8031  0000              iysto::       defw 0  ; Registerpaar IY
8033  0000              usersp::      defw 0  ; Stackpointer Anwender
8035  00                iregsto::     defb 0  ; Interruptvektor, danach rregsto
8036  00                rregsto::     defb 0  ; Refreshregister

8037  0000              syssp::       defw 0  ; Systemstackpointer

                        ; Bereich fuer die Ausfuehrung von
                        ; Befehlen beim Einzelschritt.
                        ; Sie werden dazu an diese Stelle
                        ; geschrieben.

8039  00 00 00 00       modber::      defb 0,0,0,0    ; Befehl
803D  00 00 00          modspg::      defb 0,0,0      ; Ruecksprung

                        ; Ende des Bereichs fuer den Einzelschritt

8040  00                brkzahl::     defb 0          ; -- z.Z. nicht verwendet
8041  0000              brk1::        defw 0          ; Adresse 1. Breakpoint
```

```
8043  00                        defb 0          ; Wert des geretteten Befehls 1
8044  0000              brk2::  defw 0          ; Adresse 2. Breakpoint
8046  00                        defb 0          ; Befehl 2
8047  0000              brk3::  defw 0          ; Adresse 3. Breakpoint
8049  00                        defb 0          ; Befehl 3

804A                            defs 50         ; Reserve fuer Stack
807C                    stack:: defs 1

807D  00                last::  defb 0  ; Der Speicher danach ist
                                        ; frei fuer den Anwender,
                                        ; z.B. fuer Programmeingabe
```


Macros:

Symbols:

0EF2I	ADRMSG	0069I	ANZEIGE	8000I	ANZFELD
0081	ANZLP	0092	ANZLP1	800EI	ANZPOI
000D	ANZREG	800FI	ANZSYS	0798	AS10L
07B7	AUS1	0D36	AUS1YS	07A0	AUS2
07B7	AUS3	07D7	AUS4	07DC	AUS5
8013I	AUSADR	076BI	AUSGABE	8012I	AUSMODE
0CA3	AUSMS	8015I	AUSREG	0D1DI	AUSYS
0370	B1	036F	B2	036E	B3
036D	B4	0000	BASIS	8029I	BC2STO
8021I	BCSTO	040D	BEF1	0416	BEF2
0047	BEFEX	03FAI	BEFEXX	0F4AI	BEGMSG
0F0AI	BISMSG	0645I	BREAK	8041I	BRK1
8044I	BRK2	8047I	BRK3	001E	BRKEX
0EFAI	BRKMSG	8040I	BRKZAHL	0F52I	BURNMSG
04ABI	CALLCEX	044FI	CALLEX	027EI	CASINIT
0F42I	CHKMSG	02B3I	CLEAR	00CA	CMDCAS
058D	CON2	0037	CONVEX	0F7A	COPYR
0057	CREX	0F22I	DATAMSG	00CB	DATCAS
802BI	DE2STO	8023I	DESTO	0F6AI	DEZMSG
049DI	DJNZEX	090B	DOOLP	0D52	DO10CONV

08A2	DO1BLP	0D6F	DO1CONV	0982	DO1FUEL
0950	DO1IOL	08E0	DO1IOS	0A92	DO1LAD
0B28	DO1PRL	0856I	DO1REG	09FD	DO1SPE
0C6C	DO1VGL	0CD5	DO20MES2	0D00	DO20MES3
0885	DO20R	0CDB	DO21MES2	0D06	DO21MES3
088C	DO21R	0D7F	DO2CONV	0AA1	DO2LAD
06C2I	DO2LOOP	0C90	DO2MES1	0CE1	DO2MES2
0D0C	DO2MES3	0A2F	DO2SPE	0C7C	DO2VGL
06EA	DO3	06E3	DO30	0AB4	DO3LAD
0CAA	DO3MES	0CEA	DO3MES2	0A46	DO3SPE
06FB	DO4	0719	DO5	071D	DO5LP
0CC3	DO5MES	075B	DO6	0CCE	DO6MES
076A.	DO7	092A	DOA0IOL	0919	DOB0IOL
089DI	DOBREAK	0D42I	DOCONV	0992	DOFERR
095CI	DOFUELL	08F5I	DOIOL	08CFI	DOIOS
0A56I	DOLAD	06B9I	DOLOOP	0906	DOLPIOL
0C81I	DOMES1	0CD0I	DOMES2	0CFBI	DOMES3
09A5I	DOMVE	0A4FI	DOPRF	0AF5I	DOPRL
0629I	DOPROG	0B66I	DOPRP	0851I	DOREG
09E3I	DOSPE	05A6	DOSPEICHER	06A2I	DOSTEP
8011I	DOTON	0C3DI	DOVGL	8018I	ENDADR
0B60I	ERR1PRF	0C34	ERR2PRF	0F2AI	ERRMSG
0AD4	ERRPRF	0AE6	ERRSUM	0EEAI	FEHLER
002E	FILLEX	0ED1	FINEX	0ED4	FINEX1
0B4A	FPRL	0691	FRE1L	0689I	FREEBREAK
069E	FREF	069B	FRESKIP	0242	GETAA
025A	GETADDR	0256I	GETADR	0272	GETBIS
01EDI	GETC	019B	GETD1	01AD	GETD2
01C2I	GETDEZ	01DA	GETF1HL	022B	GETFHL
0209	GETFL	0218I	GETHL	01F5	GETL
0278	GETMIT	026C	GETNACH	029AI	GETRI
00A8I	GETTAST	0266	GETVON	0EDAI	HALLO
802DI	HL2STO	8025I	HLSTO	00C6	HOL1
00BB	HOL10	00BD	HOL11	00BBI	HOLETASTE
0DC7I	INIT	800BI	INTLOC	0937I	IODATAUS
001B	IOLEX	000B	IOSEX	8035I	IREGSTO
802FI	IXSTO	8031I	IYSTO	04D3I	JMPCEX
0487I	JMPEX	0517I	JRCEX	048DI	JREX
0A61	LAD0	0A5AI	LAD1	003D	LADEX
807DI	LAST	0D9AI	LASTMEM	0000	LEDNR
0F9EI	LEER	02BCI	LENGTH	801CI	MERKCAS
0007	MES1EX	0017	MES2EX	0027	MES3EX
005E	MINUSEX	0F1AI	MITMSG	8039I	MODBER
042AI	MODR	803DI	MODSPG	0F5AI	MSMSG
000E	MVEEX	0F12I	NACHMSG	8008I	NMILOC
04D1I	NOT1EX	04D0I	NOT2EX	04CFI	NOTEX
00E6I	NUMTAB	0EE2I	OKMSG	8010I	OLDANZ
004E	OPTEX	0469I	PCHLEX	046DI	PCIXEX
0471I	PCIYEX	801DI	PCSTO	005D	PLUSEX

0290I	POD	03B9I	POPALL	001D	PRFEX
0F3AI	PRFMSG	0BFF	PRG1FIN	0C2A	PRGFIN
0C0E	PRGLP4	02A6I	PRINT	003B	PRLEX
0BD1	PRLP01	0BE3	PRLP02	0BA8	PRLP1
0081	PROMA1	0082	PROMA2	0080	PROMD
0AF0I	PROMINIT	002B	PRPEX	0151	PRT1BILP
0172	PRT1DEZ	015C	PRT2BI	0175	PRT2DEZ
0183	PRT3DEZ	012AI	PRTAC	014EI	PRTBIN
0162I	PRTDEZ	0145I	PRTHL	B01FI	PSW1STD
B027I	PSW2STD	0373I	PUSHALL	07E7	REG1A
07E2I	REGAUS	000D	REGEX	0B10I	REGTAB
04F3I	RETCEX	0475I	RETEX	0287I	RI
B036I	RREGSTD	0436I	RSTEX	0E18	SCHL1
0E72	SCHL10	0E7C	SCHL11	0E86	SCHL12
0E90	SCHL13	0E9A	SCHL14	0EA4	SCHL15
0EAE	SCHL16	0EB8	SCHL17	0EC2	SCHL18
0ECC	SCHL19	0E22	SCHL2	0E2C	SCHL3
0E36	SCHL4	0E40	SCHL5	0E4A	SCHL6
0E54	SCHL7	0E5E	SCHL8	0E68	SCHL9
0E0BI	SCHLEIFE	0F72I	SEDMSG	0001	SEGMENT
011AI	SEGTAB	0675	SET1L	066DI	SETBREAK
0682	SETNEXT	09D8	SETUPLD	0609	SP0AL
05F5	SP10L	060F	SP1AL	05D6	SP1L
0616	SP2AL	05DD	SP2L	061D	SP3AL
05E4	SP3L	05E9	SPALOOP	002D	SPEEX
005B	SPEICHEX	0622	SPEND	05B5	SPLOOP
B07CI	STACK	0DFFI	START	B016I	STARTADR
004B	STARTEX	053BI	STEP	004D	STEPEX
0F32I	STMSG	0499	SUBJR	0DC5	SUCHE
0D9D	SUCHL	B037I	SYSSP	0320	TAB2
032C	TAB3	032F	TAB4	0332	TAB5
0000	TASTIN	00D5	TO1NUM	00DF	TO2NUM
00E3	TO3NUM	00CEI	TONUM	0106I	TOSEG
B033I	USERSP	09D1	USEUP	003E	VGLEX
0F02I	VONMSG	0F62I	YSMSG	801AI	ZIELADR

No Fatal error(s)

```
rom  abs                                                    checksum
0000  C3 FF 0D C3 87 02 C3 90 02 C3 69 00 C3 BB 00 C3      += 07DD
0010  CE 00 C3 06 01 C3 A6 02 C3 2A 01 C3 45 01 C3 4E      += 060B
0020  01 C3 62 01 C3 ED 01 C3 18 02 C3 C2 01 C3 3B 05      += 063E
0030  C3 45 06 C3 B3 02 00 00 C3 0B 80 C3 BC 02 C3 9A      += 06B2
0040  0D C3 9D 0D C3 9A 02 5F 00 00 00 00 00 00 00 00      += 0338
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      += 0000
0060  00 00 00 00 00 00 C3 08 80 E5 D5 C5 3A 0E 80 3C      += 04CE
0070  E6 07 32 0E 80 21 00 80 5F 4F 06 00 09 0E 7F 3C      += 03D4
0080  47 CB 01 10 FC 3E FF D3 00 7E D3 01 79 D3 00 01      += 06CE
0090  A6 00 0B 79 B0 C2 92 00 DB 00 E6 0F FE 0F 20 08      += 0633
00A0  3E FF D3 00 C1 D1 E1 C9 CB 03 CB 03 CB 03 CB 03      += 0884
00B0  B3 F5 3E FF D3 00 F1 C1 D1 E1 C9 06 08 CD 69 00      += 0929
00C0  FE FF 20 F7 10 F7 CD 69 00 FE FF 28 F9 C9 C5 E5      += 0AE2
00D0  21 E6 00 06 10 BE 28 07 23 23 10 F9 37 18 04 23      += 03CF
00E0  7E 37 3F E1 C1 C9 0E 00 1E 01 2E 02 3E 03 0D 04      += 040E
00F0  1D 05 2D 06 3D 07 0B 08 1B 09 2B 0A 3B 0B 07 0C      += 015E
0100  17 0D 27 0E 37 0F C5 E5 21 1A 01 E6 0F 4F 06 00      += 03CF
0110  09 4E 3A 11 80 2F A1 E1 C1 C9 C0 F9 A4 B0 99 92      += 0895
0120  82 F8 80 98 88 83 C6 A1 86 8E F5 E6 F0 0F 0F 0F      += 0910
0130  0F CD 06 01 DD 77 00 DD 23 F1 E6 0F CD 06 01 DD      += 06CE
0140  77 00 DD 23 C9 7C CD 2A 01 7D CD 2A 01 C9 C5 06      += 06BD
0150  08 DD 36 00 C0 07 30 04 DD 36 00 F9 DD 23 10 F1      += 0623
0160  C1 C9 0E 00 CB 7C 28 0A 0E 01 E5 D1 21 00 00 AF      += 05A6
0170  ED 52 06 10 AF CB 25 CB 14 CB 17 FE 0A 38 04 D6      += 06CF
0180  0A CB C5 10 F0 CD 06 01 DD 77 00 DD 2B 7C B5 20      += 071B
0190  E1 79 B7 C8 DD 36 00 BF DD 2B C9 CD BB 00 FE 5E      += 0960
01A0  20 0B D5 EB 21 00 00 AF ED 52 D1 AF C9 CD CE 00      += 07DE
01B0  D8 D5 C5 E5 D1 29 29 19 29 4F 06 00 09 C1 D1 37      += 06E3
01C0  3F C9 3E 80 32 11 80 DD E5 E5 CD B3 02 CD 62 01      += 07E2
01D0  E1 DD E1 CD 9B 01 38 02 18 E8 F5 AF 32 11 80 CD      += 0876
01E0  B3 02 DD E5 E5 CD 62 01 E1 DD E1 F1 C9 E5 69 CD      += 0B00
01F0  F5 01 4D E1 C9 3E 80 32 11 80 DD E5 7D CD 2A 01      += 07A5
0200  DD E1 CD 42 02 38 02 18 EC F5 AF 32 11 80 DD CB      += 081C
0210  00 FE DD CB 01 FE F1 C9 3E 80 32 11 80 DD E5 CD      += 096F
0220  45 01 DD E1 CD 42 02 38 02 18 ED F5 AF 32 11 80      += 06BB
0230  DD CB 00 FE DD CB 01 FE DD CB 02 FE DD CB 03 FE      += 0A9E
0240  F1 C9 CD BB 00 CD CE 00 D8 C5 29 29 29 29 4F 06      += 0773
0250  00 09 C1 37 3F C9 E5 21 F2 0E CD A6 02 E1 DD 21      += 0763
0260  04 80 CD 18 02 C9 E5 21 02 0F 18 EE E5 21 12 0F      += 0578
0270  18 E8 E5 21 0A 0F 18 E2 E5 21 1A 0F 18 DC 3E 53      += 05CD
0280  D3 CA 3E 50 D3 CA C9 DB CA E6 01 28 FA DB CB C9      += 0AAE
0290  DB CA E6 02 28 FA 79 D3 CB C9 3E 7F D3 01 CD 87      += 0974
02A0  02 2F D3 00 2F C9 D5 C5 11 00 80 01 08 00 ED B0      += 05CD
02B0  C1 D1 C9 E5 21 9E 0F CD A6 02 E1 C9 06 00 54 5D      += 07E4
02C0  7E E6 DF FE DD CA 32 03 7E FE CB CA 2C 03 FE ED      += 0A48
02D0  CA 20 03 7E FE C3 CA 6E 03 FE CD CA 6E 03 E6 EF      += 0942
02E0  FE 22 CA 6E 03 FE 2A CA 6E 03 E6 CF FE 01 CA 6E      += 08AA
02F0  03 E6 C7 FE C2 CA 6E 03 FE C4 CA 6E 03 7E E6 F7      += 0A03
```

rom abs																checksum
0300	FE	10	CA	6F	03	FE	D3	CA	6F	03	E6	E7	FE	20	CA	6F	+= 097B
0310	03	E6	C7	FE	06	CA	6F	03	FE	C6	CA	6F	03	C3	70	03	+= 0826
0320	23	7E	E6	C7	FE	43	CA	6D	03	C3	6F	03	C3	6F	03	C3	+= 07F6
0330	6D	03	23	7E	FE	CB	CA	6D	03	FE	21	CA	6D	03	E6	FE	+= 0851
0340	FE	34	CA	6E	03	E6	F8	FE	70	CA	6E	03	7E	FE	36	CA	+= 0970
0350	6D	03	E6	C7	FE	06	CA	6E	03	E6	C7	FE	02	CA	6D	03	+= 0843
0360	7E	D6	40	E6	87	FE	06	CA	6E	03	C3	6F	03	04	04	04	+= 0681
0370	04	EB	C9	ED	43	21	80	ED	53	23	80	22	25	80	F5	C1	+= 07E9
0380	ED	43	1F	80	08	D9	ED	43	29	80	ED	53	2B	80	22	2D	+= 06C3
0390	80	F5	C1	ED	43	27	80	ED	57	32	35	80	ED	5F	32	36	+= 07EC
03A0	80	DD	22	2F	80	FD	22	31	80	3A	0E	80	32	10	80	3A	+= 05C2
03B0	0F	80	32	0E	80	2A	1D	80	C9	3A	0E	80	32	0F	80	3A	+= 04A2
03C0	10	80	32	0E	80	22	1D	80	FD	2A	31	80	DD	2A	2F	80	+= 059D
03D0	3A	35	80	ED	47	ED	4B	27	80	C5	F1	2A	2D	80	ED	5B	+= 07D7
03E0	2B	80	ED	4B	29	80	D9	08	ED	4B	1F	80	C5	F1	2A	25	+= 0749
03F0	80	ED	5B	23	80	ED	4B	21	80	C9	3E	C3	32	3D	80	E5	+= 07E2
0400	21	2A	04	22	3E	80	E1	AF	06	04	11	3C	80	12	1B	10	+= 03D3
0410	FC	D5	CD	BC	02	D1	7E	13	12	23	10	FA	CD	B9	03	ED	+= 0873
0420	73	37	80	ED	7B	33	80	C3	39	80	ED	73	33	80	ED	7B	+= 083C
0430	37	80	CD	73	03	C9	7E	23	ED	73	37	80	ED	7B	33	80	+= 0796
0440	E5	ED	73	33	80	ED	7B	37	80	26	00	E6	38	6F	C9	23	+= 07B6
0450	23	23	ED	73	37	80	ED	7B	33	80	E5	ED	73	33	80	ED	+= 085D
0460	7B	37	80	2B	7E	2B	6E	67	C9	2A	25	80	C9	2A	2F	80	+= 0615
0470	C9	2A	31	80	C9	ED	73	37	80	ED	7B	33	80	E1	ED	73	+= 08E0
0480	33	80	ED	7B	37	80	C9	23	7E	23	66	6F	C9	23	7E	23	+= 06C1
0490	4F	B7	FA	99	04	06	00	09	C9	06	FF	09	C9	ED	4B	21	+= 06A5
04A0	80	05	ED	43	21	80	20	E5	23	23	C9	E5	21	4F	04	22	+= 05E5
04B0	3A	80	21	CF	04	22	3D	80	3E	C3	32	3C	80	E1	7E	E6	+= 06C1
04C0	38	F6	C2	32	39	80	ED	4B	1F	80	C5	F1	C3	39	80	23	+= 0807
04D0	23	23	C9	E5	21	87	04	22	3A	80	21	CF	04	22	3D	80	+= 054F
04E0	3E	C3	32	3C	80	E1	7E	32	39	80	ED	4B	1F	80	C5	F1	+= 07C6
04F0	C3	39	80	E5	21	75	04	22	3A	80	21	D1	04	22	3D	80	+= 05AC
0500	3E	C3	32	3C	80	E1	7E	E6	38	F6	C2	32	39	80	ED	4B	+= 0847
0510	1F	80	C5	F1	C3	39	80	E5	21	8D	04	22	3A	80	21	D0	+= 0735
0520	04	22	3D	80	3E	C3	32	3C	80	E1	7E	E6	18	F6	C2	32	+= 0719
0530	39	80	ED	4B	1F	80	C5	F1	C3	39	80	7E	FE	C9	CA	75	+= 0946
0540	04	FE	CD	CA	4F	04	FE	C3	CA	87	04	FE	18	CA	8D	04	+= 0873
0550	FE	10	CA	9D	04	FE	E9	CA	69	04	E6	E7	FE	20	CA	17	+= 0963
0560	05	E6	C7	FE	C2	CA	D3	04	FE	C4	CA	AB	04	FE	C7	CA	+= 0ADD
0570	36	04	FE	C0	CA	F3	04	7E	FE	ED	C2	8D	05	23	7E	2B	+= 0842
0580	FE	4D	CA	75	04	FE	45	CA	75	04	C3	FA	03	7E	E6	DF	+= 0917
0590	FE	DD	C2	FA	03	23	7E	2B	FE	E9	C2	FA	03	7E	FE	DD	+= 0A65
05A0	CA	6D	04	C3	71	04	21	9E	0F	CD	A6	02	2A	1D	80	CD	+= 064A
05B0	56	02	FE	47	C8	CD	B3	02	DD	21	00	80	CD	45	01	DD	+= 0755
05C0	21	06	80	4E	CD	ED	01	FE	47	C8	FE	57	20	08	79	77	+= 072A
05D0	BE	20	E2	23	18	DF	FE	5D	20	03	23	18	D8	FE	5E	20	+= 06E7
05E0	03	2B	18	D1	FE	4E	C2	22	06	CD	B3	02	E5	CD	BC	02	+= 073F
05F0	DD	21	00	80	C5	7E	CD	2A	01	23	10	F9	CD	BB	00	C1	+= 072E

```
------------- Seite  3  ------- File HEXMON 1.1a -----------
rom  abs                                                  checksum
0600  E1 FE 5B 28 04 FE 57 20 06 48 06 00 09 18 DA FE      += 0628
0610  5D 20 03 23 18 D3 FE 5E 20 03 2B 18 CC FE 4E CA      += 0632
0620  B5 05 22 1D 80 22 18 80 C9 2A 1D 80 CD 56 02 22      += 050A
0630  1D 80 FE 57 C0 CD 6D 06 ED 73 37 80 ED 7B 33 80      += 0824
0640  E5 CD B9 03 C9 CD 73 03 E1 2B 22 1D 80 ED 73 33      += 07D8
0650  80 31 7C 80 CD B9 06 21 FA 0E CD A6 02 DD 21 04      += 06A9
0660  80 2A 1D 80 CD 45 01 CD BB 00 C3 0B 0E C5 D5 E5      += 073D
0670  21 41 80 06 03 5E 23 56 23 7A B3 28 05 1A 77 3E      += 040E
0680  F7 12 23 10 F0 E1 D1 C1 C9 C5 D5 E5 21 41 80 06      += 08CF
0690  03 5E 23 56 23 7A B3 28 02 7E 12 23 10 F3 E1 D1      += 05BC
06A0  C1 C9 AF 32 12 80 2A 1D 80 CD 56 02 FE 47 C8 22      += 0718
06B0  1D 80 FE 57 28 0C FE 4D C0 2A 1D 80 CD 3B 05 22      += 0627
06C0  1D 80 CD 6B 07 CD BB 00 FE 47 C8 FE 4D 28 EA FE      += 08CC
06D0  57 28 E6 FE 4E 20 13 3A 12 80 B7 28 06 AF 32 12      += 0588
06E0  80 18 DF 3E 01 32 12 80 18 D8 FE 0D 20 0D 3E 02      += 04E2
06F0  32 12 80 3A 15 80 CD 56 08 18 C7 FE 5B 20 1A CD      += 05FD
0700  B3 02 DD 21 04 80 2A 13 80 CD 18 02 FE 47 C8 22      += 060A
0710  13 80 3E 03 32 12 80 18 A9 FE 4B 20 3E ED 5B 1D      += 0565
0720  80 2A 41 80 AF ED 52 7D B4 CA C2 06 2A 44 80 AF      += 07B9
0730  ED 52 7D B4 CA C2 06 2A 47 80 AF ED 52 7D B4 CA      += 08DC
0740  C2 06 CD 6B 07 CD 69 00 FE 47 C8 FE 4E CA C2 06      += 0828
0750  2A 1D 80 CD 3B 05 22 1D 80 18 C2 FE 5E 20 0B 3E      += 0532
0760  04 32 12 80 CD B3 02 C3 C2 06 C9 3A 12 80 B7 20      += 0641
0770  16 CD B3 02 DD 21 00 80 2A 1D 80 CD 45 01 DD 21      += 05EE
0780  06 80 7E CD 2A 01 C9 FE 01 20 15 2A 1D 80 CD B3      += 0640
0790  02 CD BC 02 DD 21 00 80 7E CD 2A 01 23 10 F9 C9      += 0676
07A0  FE 02 20 13 3A 15 80 CD E2 07 4E 23 46 69 60 DD      += 0615
07B0  21 04 80 CD 45 01 C9 FE 03 20 1C CD B3 02 DD 21      += 063E
07C0  00 80 2A 13 80 CD 45 01 DD 36 00 BF DD 23 7E CD      += 066D
07D0  2A 01 DD 36 00 BF C9 FE 04 20 01 C9 AF 32 12 80      += 0625
07E0  18 89 FE 0D 38 01 AF F5 CD B3 02 F1 21 10 08 4F      += 0684
07F0  06 00 09 09 09 09 09 4E 23 46 DD 21 00 80 23 7E      += 0309
0800  DD 77 00 23 7E DD 77 01 23 7E DD 77 02 69 60 C9      += 06D3
0810  1F 80 88 8E FF 21 80 83 C6 FF 23 80 A1 86 FF 25      += 088B
0820  80 89 C7 FF 33 80 92 8C FF 2F 80 F9 89 FF 31 80      += 0980
0830  F9 8D FF 27 80 88 8E DF 29 80 83 C6 DF 2B 80 A1      += 093E
0840  86 DF 2D 80 89 C7 DF 35 80 AF F9 FF 1D 80 8C C6      += 098C
0850  FF 3E 00 32 15 80 CD E2 07 4E 23 46 E5 69 60 DD      += 06FC
0860  21 04 80 CD 18 02 FE 47 4D 44 E1 C8 70 2B 71 FE      += 0715
0870  5E 20 12 3A 15 80 3D 32 15 80 B7 F2 56 08 3E 0C      += 04B4
0880  32 15 80 18 D1 FE 57 28 03 FE 5D C0 3A 15 80 3C      += 0656
0890  32 15 80 FE 0D 38 BF AF 32 15 80 18 B9 06 01 21      += 0538
08A0  41 80 C5 CD B3 02 DD 21 00 80 C1 78 C5 CD 2A 01      += 077C
08B0  DD 21 04 80 5E 23 56 2B EB D5 CD 18 02 D1 C1 EB      += 07A8
08C0  FE 47 C8 73 23 72 23 23 04 78 FE 04 20 D4 C9 CD      += 0763
08D0  B3 02 DD 21 00 80 0E 00 CD ED 01 FE 47 C8 1E 00      += 0627
08E0  C5 DD 21 06 80 4B CD ED 01 59 C1 FE 47 C8 ED 59      += 08BC
08F0  FE 57 28 EC C9 CD B3 02 DD 21 00 80 0E 00 16 00      += 0656
```

```
rom  abs                                                    checksum
0900  CD ED 01 FE 47 C8 ED 78 CD 37 09 CD BB 00 FE 47        += 0907
0910  C8 FE 57 28 F1 FE 4B 20 11 ED 78 CD 37 09 CD 69        += 0858
0920  00 FE 47 C8 FE 57 28 E3 18 EF FE 4E C0 7A EE 01        += 08E9
0930  57 7B CD 37 09 18 D4 5F 7A B7 20 14 CD B3 02 DD        += 06EE
0940  21 00 80 79 CD 2A 01 DD 21 06 80 7B CD 2A 01 C9        += 05D2
0950  CD B3 02 7B DD 21 00 80 CD 4E 01 C9 21 00 00 CD        += 064E
0960  66 02 FE 47 C8 E5 21 00 00 CD 72 02 D1 FE 47 C8        += 079A
0970  E5 21 22 0F CD A6 02 E1 DD 21 06 80 0E 00 CD ED        += 06D9
0980  01 EB 79 77 BE 20 0B E5 AF ED 52 7C B5 E1 C8 23        += 0895
0990  18 F0 E5 21 2A 0F CD A6 02 DD 21 04 80 E1 CD 45        += 0731
09A0  01 CD BB 00 C9 21 00 00 CD 66 02 FE 47 C8 E5 CD        += 0767
09B0  72 02 D1 FE 47 C8 E5 CD 6C 02 C1 FE 47 C8 E5 AF        += 09D4
09C0  ED 52 E1 DA D1 09 CD D8 09 09 EB 09 EB 03 ED B8        += 0912
09D0  C9 CD D8 09 03 ED B0 C9 E5 60 69 AF ED 52 EB 4B        += 09B2
09E0  42 D1 C9 2A 16 80 CD 66 02 22 16 80 FE 47 C8 2A        += 06C0
09F0  18 80 CD 72 02 22 18 80 FE 47 C8 06 14 0E FF CD        += 0694
0A00  90 02 10 F9 0E 00 CD 90 02 0E 3A CD 90 02 ED 5B        += 05F7
0A10  16 80 21 00 00 19 4A CD 90 02 4B CD 90 02 ED 5B        += 056B
0A20  18 80 19 4A CD 90 02 4B CD 90 02 ED 5B 16 80 1A        += 05FC
0A30  4F 06 00 09 CD 90 02 E5 2A 18 80 AF ED 52 7C B5        += 0683
0A40  E1 28 03 13 18 E9 4C CD 90 02 4D CD 90 02 C9 3E        += 067E
0A50  01 32 1C 80 18 04 AF 32 1C 80 CD 9A 02 FE FF 20        += 05EE
0A60  F9 CD 9A 02 FE FF 28 F9 FE 00 20 EE CD 9A 02 FE        += 09F3
0A70  3A 20 E7 21 00 00 CD 9A 02 57 CD 9A 02 5F ED 53        += 062A
0A80  16 80 19 D5 CD 9A 02 57 CD 9A 02 5F ED 53 18 80        += 06E4
0A90  19 D1 CD 9A 02 4F 06 00 09 3A 1C 80 B7 20 02 79        += 04D9
0AA0  12 1A B9 20 2F E5 2A 18 80 AF ED 52 7D B4 E1 28        += 0703
0AB0  03 13 18 DE CD 9A 02 BC 20 2C CD 9A 02 BD 20 26        += 05E9
0AC0  21 32 0F CD A6 02 DD 21 04 80 2A 16 80 CD 45 01        += 052C
0AD0  CD BB 00 C9 21 3A 0F CD A6 02 EB DD 21 04 80 CD        += 076A
0AE0  45 01 CD BB 00 C9 21 42 0F CD A6 02 CD BB 00 C9        += 06CF
0AF0  3E 40 D3 82 C9 3E 40 D3 82 21 00 00 CD 66 02 FE        += 06C3
0B00  47 C8 22 16 80 21 FF 07 CD 72 02 FE 47 C8 22 18        += 0676
0B10  80 2A 1D 80 CD 6C 02 FE 47 C8 22 1A 80 2A 16 80        += 060B
0B20  ED 5B 18 80 DD 2A 1A 80 7D D3 81 7C E6 1F F6 40        += 0809
0B30  D3 82 DB 80 DD 77 00 DD BE 00 20 24 E5 AF ED 52        += 08B6
0B40  7C B5 E1 28 05 DD 23 23 18 DE 2A 1A 80 ED 4B 16        += 066A
0B50  80 22 16 80 ED 5B 18 80 19 AF ED 42 22 18 80 C9        += 0692
0B60  DD E5 D1 C3 D4 0A 3E 40 D3 82 2A 16 80 CD 66 02        += 07FC
0B70  FE 47 C8 22 16 80 2A 18 80 CD 72 02 FE 47 C8 22        += 06F7
0B80  18 80 21 00 00 CD 6C 02 FE 47 C8 22 1A 80 21 4A        += 0528
0B90  0F CD A6 02 CD BB 00 FE 57 C0 CD B3 02 2A 16 80        += 0763
0BA0  ED 5B 18 80 ED 4B 1A 80 3E 80 D3 82 7E D3 80 79        += 080F
0BB0  D3 81 78 E6 1F F6 80 D3 82 F6 20 D3 82 E6 DF E5        += 0AB1
0BC0  D5 C5 DD 21 00 80 CD 45 01 7E DD 21 06 80 CD 2A        += 0724
0BD0  01 CD 69 00 DB 81 E6 01 20 F7 C1 C5 78 E6 1F D3        += 0867
0BE0  82 06 0A CD 69 00 10 FB C1 D1 E1 DB 80 BE C2 34        += 0855
0BF0  0C E5 AF ED 52 7D B4 E1 28 05 23 03 C3 A8 0B 3E        += 06F8
```

```
rom  abs                                                    checksum
0C00  40 D3 82 2A 16 80 ED 5B 18 80 ED 4B 1A 80 79 D3        += 0753
0C10  81 78 E6 1F F6 40 D3 82 DB 80 BE 20 17 E5 AF ED        += 095A
0C20  52 7C B5 E1 28 04 23 03 18 E4 21 52 0F CD A6 02        += 05A9
0C30  CD BB 00 C9 E5 D1 3E 40 D3 82 C3 D4 0A 2A 16 80        += 083B
0C40  CD 66 02 FE 47 C8 22 16 80 2A 18 80 CD 72 02 FE        += 06FB
0C50  47 C8 22 18 80 2A 1A 80 CD 78 02 FE 47 C8 22 1A        += 061D
0C60  80 2A 16 80 ED 5B 18 80 ED 4B 1A 80 0A BE 20 0C        += 05E6
0C70  E5 AF ED 52 7C B5 E1 C8 23 03 18 F0 E5 D1 C3 D4        += 0A28
0C80  0A 21 00 00 3E 80 D3 82 3E A0 D3 82 3E 80 D3 82        += 0684
0C90  23 13 DB 81 E6 01 C2 90 0C CD A3 0C 30 E3 3E 40        += 06E4
0CA0  D3 82 C9 CD B3 02 CB 7C 20 24 E5 21 5A 0F CD A6        += 080D
0CB0  02 DD 21 04 80 E1 CD 62 01 DD 21 02 80 DD CB 00        += 06BD
0CC0  BE 06 C8 CD 69 00 FE 47 28 04 10 F7 AF C9 37 C9        += 07B2
0CD0  21 02 00 06 0A DB 00 07 D2 D5 0C DB 00 07 DA DB        += 055F
0CE0  0C 23 13 DB 00 E6 80 CA E1 0C 23 13 DB 00 E6 80        += 06B1
0CF0  C2 EA 0C 10 EC CD 1D 0D 30 D6 C9 21 00 00 06 0A        += 05AB
0D00  DB 00 07 D2 00 0D DB 00 07 DA 06 0D 23 13 DB 00        += 04A1
0D10  E6 80 CA 0C 0D 10 EF CD 1D 0D 30 DF C9 CD B3 02        += 0799
0D20  CB 7C C2 CE 0C E5 21 62 0F CD A6 02 DD 21 04 80        += 0751
0D30  E1 CD 62 01 06 64 CD 69 00 FE 47 CA CE 0C 10 F6        += 07A0
0D40  AF C9 CD B3 02 21 00 00 DD 21 04 80 CD 18 02 FE        += 0682
0D50  47 C8 E5 21 6A 0F CD A6 02 E1 DD 21 07 80 E5 CD        += 081B
0D60  62 01 E1 CD BB 00 FE 47 C8 FE 4E 20 D5 18 10 CD        += 080F
0D70  B3 02 DD 21 07 80 21 00 00 CD C2 01 FE 47 C8 E5        += 06DD
0D80  21 72 0F CD A6 02 E1 DD 21 04 80 CD 45 01 CD BB        += 0715
0D90  00 FE 47 C8 FE 4E 20 D7 18 B8 21 7D 80 46 36 55        += 070F
0DA0  23 4E 36 AA 2B 3E 55 BE 20 1B 70 78 BE 20 16 23        += 0507
0DB0  3E AA BE 20 10 71 79 BE 20 0B 7C FE FF 20 DE 7D        += 079D
0DC0  FE FF 20 D9 23 2B C9 AF 32 12 80 32 15 80 32 40        += 06B9
0DD0  80 32 11 80 21 00 00 22 41 80 22 44 80 22 47 80        += 0416
0DE0  21 00 81 22 1D 80 22 13 80 22 16 80 22 1A 80 22        += 03AC
0DF0  18 80 CD 9A 0D 22 33 80 CD 7E 02 CD F0 0A C9 31        += 06EF
0E00  7C 80 21 DA 0E CD A6 02 CD C7 0D CD BB 00 FE 5B        += 07FC
0E10  20 06 CD A6 05 C3 D1 0E FE 4B 20 06 CD 29 06 C3        += 066E
0E20  D1 0E FE 4D 20 06 CD A2 06 C3 D1 0E FE 0D 20 06        += 0698
0E30  CD 51 08 C3 D1 0E FE 1B 20 06 CD F5 08 C3 D1 0E        += 0773
0E40  FE 0B 20 06 CD CF 08 C3 D1 0E FE 3D 20 06 CD 56        += 06F9
0E50  0A C3 D1 0E FE 2D 20 06 CD E3 09 C3 D1 0E FE 1D        += 0773
0E60  20 06 CD 4F 0A C3 D1 0E FE 0E 20 06 CD A5 09 C3        += 065E
0E70  D1 0E FE 1E 20 06 CD 9D 08 C3 D1 0E FE 2B 20 06        += 0684
0E80  CD 66 0B C3 D1 0E FE 3B 20 06 CD F5 0A C3 D1 0E        += 07AD
0E90  FE 2E 20 06 CD 5C 09 C3 D1 0E FE 3E 20 06 CD 3D        += 0692
0EA0  0C C3 D1 0E FE 07 20 06 CD 81 0C C3 D1 0E FE 17        += 06EA
0EB0  20 06 CD D0 0C C3 D1 0E FE 27 20 06 CD FB 0C C3        += 0753
0EC0  D1 0E FE 37 20 06 CD 42 0D C3 D1 0E 21 EA 0E 18        += 0629
0ED0  03 21 E2 0E CD A6 02 C3 0B 0E 89 88 C7 C7 C0 BF        += 0783
0EE0  79 F9 BF 83 86 8E BF FF FF FF BF BF BF BF BF BF        += 0BFE
0EF0  BF BF 88 A1 AF FF FF FF FF FF 83 AF 86 FF FF FF        += 0D06
```

```
-------------- Seite  6  ------- File HEXMON 1.1a --+---------
rom  abs                                                checksum
0F00  FF FF C1 C0 C8 FF FF FF FF FF 83 F9 92 FF FF FF    += 0E4D
0F10  FF FF C8 88 C6 FF FF FF FF FF AA CF 87 FF FF FF    += 0E0C
0F20  FF FF A1 87 88 FF FF FF FF FF 86 AF AF FF FF FF    += 0D8A
0F30  FF FF 92 87 B7 FF FF FF FF FF 8C AF 8E FF FF FF    += 0D8F
0F40  FF FF C6 89 86 FF FF FF FF FF C6 AF FF FF FF FF    += 0E3F
0F50  FF FF 83 E3 AF AB 86 A1 FF FF FF FF 7F C0 C0 FF    += 0CDF
0F60  AA 92 FF FF FF FF FF FF 8D 92 A1 FF FF FF FF FF    += 0DF1
0F70  FF FF 92 FF FF FF FF FF FF FF 28 43 29 20 31 39    += 0AA7
0F80  38 34 20 4D 75 65 6E 63 68 65 6E 20 52 6F 6C 66    += 0572
0F90  2D 44 69 65 74 65 72 20 4B 6C 65 69 6E 20 FF FF    += 06BB
0FA0  FF FF FF FF FF FF 00 00 00 00 00 00 00 00 00 00    += 05FA
0FB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    += 0000
0FC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    += 0000
0FD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    += 0000
0FE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    += 0000
0FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    += 0000
```

G Bezugsquellenverzeichnis

Alle Bausätze der beschriebenen Baugruppen (einschl. HEXMON-Eprom) können bei der folgenden Firma bezogen werden:

Graf Elektronik Systeme GmbH

Magnusstraße 13

Postfach 1610

8968 Kempten

Telefon: 0831/6211

Unterlagen, Listings etc. zum NDR-Klein-Computer können auch im Franzis-Verlag beim Software-Service angefordert werden; eine Übersicht liegt bereit.
(Franzis-Verlag GmbH, Karlstraße 37- 41, 8000 München 2)

H Literaturverzeichnis

[1] ZILOG Z80-Datenbuch, Vertrieb durch Kontron, München.

[2] Z80-Assembler Handbuch. Vertrieb durch Kontron, München.

[3] Leventhal, Lance A. Z80 Assembly Language Programming, erschienen bei OSBORNE/McGRAW-HILL, Berkeley California.

[4] Klein, Rolf-Dieter. Mikrocomputersysteme. Franzis-Verlag

[5] Klein, Rolf-Dieter. Mikrocomputer Hard- und Softwarepraxis. Franzis-Verlag.

[6] Klein, Michael u. Klein, Rolf-Dieter. Z80-Applikationsbuch. Franzis-Verlag.

[7] Klein, Rolf-Dieter. Mikrocomputer selbstgebaut und programmiert.

[8] Klein, Rolf-Dieter. Basic-Interpreter. Franzis-Verlag.

[9] Klein, Rolf-Dieter. Was ist PASCAL. Franzis-Verlag.

Stichwortverzeichnis

M

N

O

P

R

S

T

U

V

Z

Klein
Mit HEXMON Programme entwickeln

Rolf-Dieter Klein

HEXMON ist eine Maschinensprache, zu der ein Computer-System gehört. Beides hat der Autor selbst entwickelt, genau so wie den NDR-Klein-Computer. HEXMON ist gewissermaßen eine Miniatur-Ausgabe.
Aus nur fünf Karten besteht die Hardware. Sie ist leicht aufzubauen, denn ihre Beschreibung ist folgerichtig und durchsichtig, auch für Anfänger. Wer will, kann sie sich fertig kaufen. Das ist vielleicht der direkteste und der billigste Weg.
Die Software, in einem EPROM geladen, ist denkbar kompakt, durchdacht und komfortabel. In acht Versuchen wird zunächst die Bedienung des HEXMON-Systems geübt. Danach erfolgt die Erläuterung der 24 HEXMON-Befehle sowie ihre Arbeitsweise. 23 fertige Unterprogramme dienen sowohl dem Komfort als auch der Einfachheit des ganzen Systems. – Übrigens: Ein HEXMON-Listing ist im Anhang komplett wiedergegeben.
Was hat der Benutzer von dem neuartigen HEXMON-System, das er sich nach diesem Buch rasch und problemlos aufbauen kann?

1. Das HEXMON-System ist preiswert.
2. Für die Investition wird viel Hardware geboten.
3. Die Kommunikation Mensch-Computer-Mensch über eine Hexadezimal-Tastatur und Siebensegmentanzeige ist für Lernzwecke hervorragend geeignet.
4. Die Software ist so einfach, daß sie auf Anhieb begriffen wird.
5. Die eigenen Z-80-Programme sind mit HEXMON einfach zu erstellen.
6. Mit HEXMON werden Programme erstellt, die auf der gleichen Hardware minus einer Karte als Steuerrechner laufen können.
7. Anpassungsprobleme sind vom Tisch.
8. Der Steuerrechner kann z.B. für Alarmanlage, Modellbahn, Roboter, Wärmeregulierung, Musik usw. verwendet werden.

ISBN 3-7723-7831-5

Franzis'