mikroelektronik

Computer

Hinweise Zur Anwendung des Lerncomputers LC 80

Applikation

PDF-Version: Volker Pohlers, 2004 incl. Berichtingungsblatt

Hinweise zur Anwendung des

Lerncomputers LC -80

Autor: Dipl.-Ing. Gunther Zielosko

HALLO, USER OF LC-80!

Ein Einstieg in die Mikroprozessortechnik für Besitzer
und Benutzer des Lerncomputers LC-80 des
veb mikroelektronik "karl marx" erfurt - Stammbetrieb

veb mikroelektronik >karl marx< erfurt
stammbetrieb
DDR — 5010 Erfurt, Rudolfstraße 47 Telefo 5 80 Telext 061 306

Vorwort

Die vorliegende Broschüre wendet sich an alle, die ohne umfassende Spezialkenntnisse auf den Gebieten Elektronik, Digitaltechnik, Mikroprozessortechnik und -programmierung den Einstieg in diese Bereiche wagen wollen.
Dabei beschränkt sich das angebotene Material keinesfalls taut reine Programmierungsprobleme, sondern kombiniert in den meisten Fällen die Anwendungsbeispiele so, daß auch Raum für "konventionelles" Basteln bleibt. Gerade diese reizvolle Verbindung von Hardware und Software für Anfänger erfordert eine neue Art der Information.
übliche Informationsquellen sind meist zu umfangreich, zu kompliziert, zu theoretisch, zu speziell und in einer Sprache geschrieben, die zwar exakt wissenschaftlich ist, aber dem Anfänger oft nicht weiterhilft. Zudem ist das Schriftmaterial allein nicht geeignet, ein kreatives Auseinandersetzen mit dem Problem zu ermöglichen.
Dem besonderen Zweck der vorliegenden Broschüre gemäß wurden Darstellungsweisen gewählt, die unkompliziert dem jeweiligen Problem angepaßt wurden.
Mit dem Vorliegen des Lerncomputers LC-80 wurde die Möglichkeit geschaffen, daß sich auch "Nichtfachleute" im privaten Bereich mit diesen neuen Techniken anfreunden können. Dies gilt im gleichen Maße für Schüler, die in Arbeitsgemeinschaften oder in den Stationen Junger Techniker Zugang zur Mikrorechnertechnik suchen.
Das Vorhandensein des einfachen Rechners LC-80, der Wille und der Wunsch nach mehr Wiesen auf diesem Gebiet sowie etwas Zeit reichen zunächst aus, um die ersten Schritte in die Welt der Computer zu tun.
Wenn das geschafft ist, ist der Anfänger meist genügend motiviert und in der Lage, die weiterführende Literatur zu verstehen. Diese Broschüre soll versuchen, dieses Ziel zu erreichen.

Inhaltsverzeichnis

0. Einleitung

Wir haben uns vorgenommen, mit Hilfe des Lerncomputers LC-80 die Grundlagen der Mikrorechnertechnik und -programmierung zu erlernen. Das soll zunächst "spielend" vor sich gehen, d. h. wir werden einfache Experimente mit unserem Lerncomputer anstellen und dabei allerlei nützliche Dinge lernen. Wichtige Voraussetzungen dafür sind das Vorhandensein eines solchen Rechners, ein dazugehöriges Netzteil, eia Kassettengerät, die Bedienungsanleitung LC-80 sowie eine Befehlsliste für den U 880 D, möglichst in kommentierter Form [4]. Aber auch im LC-80 - Handbuch finden wir eine Befehlsliste.
Damit wir gleich anfangen können, lesen wir uns zunächst einmal die Bedienungsanleitung des LC-80, Kapitel 3 - Inbetriebnahme und Programmeingabe, durch. Danach schließen wir unseren Rechner genauso an wie dort beschrieben [5]. Wenn der LC-80 ordnungsgemäß funktioniert - er spielt normalerweise eine Melodie vor und begrüßt seinen Benutzer mit einem Text - steht in der Anzeige sein Name "LC-80".
Für die weitere Arbeit schauen wir uns jetzt die Darstellung der Bedienelemente und Anschlußstellen an (Kapitel 1.2 auf S. 8 der Bedienungsanleitung). In den folgenden Abschnitten wollen wir unsere Bedienungselemente wie Tastatur und Anzeige kennenlernen und den Umgang mit dem Rechner üben.

1. Die Tastatur

Eine der wichtigsten Tasten ist RES (Rücksetzen). Diese Taste dient zur Herstellung des Grundzustandes des Rechners, was durch die Anzeige "LC-80" angezeigt wird. Drücken wir sie, und es kann losgehen.

Wir wollen unseren Rechner programmieren, d. h. ihn veranlassen, etwas zu tun. Unser Rechner bekommt seine Anweisungen durch das Programm, welches er schrittweise abarbeitet. Diese Schritte geben wir ihm einzeln vor. Die einzelnen abzuarbeitenden Schritte müssen wir in den Arbeitsspeicher "eintragen", der sich im sog. RAM-Bereich befindet (sh. a. Abschnitt 8). Dort sind Speicherzellen vorhanden, die unsere Befehle aufnehmen können. Wie bei jeder Nachrichtenübertragung sind zwei Angaben erforderlich, einmal

- die Adresse (Wohin soll unsere Nachricht gehen?)
- die Nachricht selbst (in der Rechnersprache als Daten bezeichnet).

Auf unserem Rechner finden wir zwei Tasten, die genau diese Beschriftung haben:

- ADR Adresseneingabe
- DAT Dateneingabe

Nachdem wir RES betätigt haben, versuchen wir's nun mit ADR . Die Anzeige "LC-80" verschwindet und auf dem Display (Anzeigefeld) erscheint:

2.0.0.0.X X — X soll ab jetzt heißen, daß hier irgendwelche, für uns nicht wichtige Werte stehen.

Was heißt das nun?
Das Display ist 6stellig, d. h. es können maximal 6 Symbole (Ziffern, Buchstaben usw.) dargestellt werden. Die ersten 4 Stellen sind in unserem Fall die Adresse, in die unser erster Befehl eingetragen werden könnte.
Wir stellen fest, daß dies die Adresse 2000 ist, warum nicht 0000? In den Speicherplätzen von 0000 ... 2000 befinden sich Informationen, die vom Hersteller unseres Rechners "für immer" dort eingetragen wurden, um seine Arbeitsfähigkeit zu gewährleisten. Sie stehen uns also nicht zur Verfügung.
Wir merken uns, daß alle unsere Programme in Zukunft zunächst bei der Adresse 2000 beginnen.

Zurück zu unserer Anzeige.
Es wird also die Adresse 2000 angezeigt. Die vier Punkte deuten an, daß jetzt die Adresse verändert werden kann (z. b. durch Betätigen der Zifferntasten). Wir wollen das aber nicht tun, sondern uns mit der Adresse 2000 begnügen, in die wir ja Informationen, also Daten, eintragen wollen.
Dazu müssen wir die Taste DAT betätigen. Der Rechner quittiert dies, indem die Punkte der vier Adressenstellen in der Anzeige verschwinden; dafür leuchten jetzt zwei Punkte in den letzten beiden Stellen. Dies sind die Anzeigen für unsere Daten. Das Aufleuchten der Punkte zeigt an, daß jetzt "Daten" eingetragen werden können. Wie funktioniert das?
Unser Rechner ist sehr einfach aufgebaut, aber sehr leistungsfähig. Durch den einfachen Aufbau bedingt "versteht" er nur Zahlen. Das heißt, prinzipiell zeigt er nur Zahlen an und man kann auch nur Zahlen eingeben. Allerdings unterscheiden sich diese Zahlen von unseren bisher gewohnten Zahlen im Dezimalsystem. Im folgenden wollen wir dies kurz behandeln.

Das Dezimalsystem ist auf den Ziffern 0 ... 9 aufgebaut, alle Zahlen bestehen aus diesen Ziffern. Ein weiteres Zahlensystem ist das Binär- oder Dualsystem, das, wie der Name schon sagt, nur aus zwei Ziffern besteht - 0 und 1. Während Dezimalzahlen aus der Summe von Potenzen zur Basis 10 bestehen, ist das beim Dualsystem die Basis 2. Ein Beispiel:
Die Dezimalzahl 723 setzt sich wie folgt zusammen:

3	"Einer"	3×10^0 =	3
+ 2	"Zehner"	2×10^1 =	20
+ 7	"Hunderter"	7×10^2 =	700
			723

Genauso ist es im Binärsystem, nur eben alles zur Basis 2. Auch hier ein Beispiel:
Die Zahl 5 im Dezimalsystem stellt sich im Binärsystem so dar:

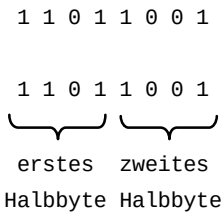
```
  1 "Einer"       1 x 2^0 =   1
+ 0 "Zweier"      0 x 2^1 =   0
+ 1 "Vierer"      1 x 2^2 =   4
+ 0 "Achter"      0 x 2^3 =   0    zusammengefaßt: 0101
                              5
```

Im Folgenden wollen wir einmal in Tabellenform Dezimalzahlen und Binärzahlen in ansteigender Folge nebeneinanderschreiben, und zwar zuerst dezimal, dann binär und daneben "hexadezimal". Das hört sich "verhext" an, ist aber gar nicht so schwer, wenn man erst einmal den Zusammenhang begriffen hat:

Dezimal	Binär	Hexadezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Was fällt uns auf?
Zuerst, daß das Hexadezimalsystem neben Ziffern auch Buchstaben enthält.
Zweitens, daß es genau 16 (mit der "0") Binärzahlen mit vier Stellen gibt. Alle Möglichkeiten sind damit ausgeschöpft .

Und das ist genau der Grund, weshalb man das Hexadezimalsystem erfunden hat. Das reine Dezimalsystem mit einer Stelle schöpft eben nicht alle Möglichkeiten des Binärsystems mit 4 Stellen aus, das Hexadezimalsystem tut das, denn es hat 16 Ziffern.
Unser Rechner arbeitet intern mit 8 Binärstellen, jede dieser Stellen heißt in der Rechnersprache "Bit". Jedes dieser Bits kann den Wert "0" oder "1" haben. 8 Bit sind zusammen ein "Wort", in der Rechnersprache "Byte".
Ein Byte mit 8 Bit wird in zwei Halbbytes zu je 4 Bit aufgeteilt, die dann jeweils im Hexacode zwei Hexa-Zeichen ergeben.
Beispiel:



Im Hexacode nach unserer Tabelle bestimmen wir dafür:

D 9

Dieses Spiel können wir zur Übung wiederholen:

- Wie stellt man im Hexacode folgende Binärzahlen dar?

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

- Wie sehen die 8Bit-Worte (Bytes) aus, die durch folgende Hexa-Zahlen gekennzeichnet werden?

F 7
3 E
2 C
D E

Na, alles klar?
Wir verstehen das nun - aber unser Rechner?
Schauen wir uns wieder unser Tastenfeld an. Tatsächlich hat es neben den weißen Zifferntasten von 0 bis 9 noch weiße Buchstabentasten A, B, C, D, E und F, wie geschaffen für uns Hexa-Spezialisten. Also los!
Unser Rechner wartet noch geduldig auf unsere Dateneingabe in die Adresse 2000, die beiden Punkte der Datenanzeige leuchten. Wir drücken erstmal

[1]

Er hat es verstanden und zeigt die "1" in der letzten Stelle an. Geht das auch mit den Hexa-Buchstaben? Also

[C]

Er zeigt jetzt " 1 C " an, wir sehen also, daß das zuerst eingegebene Halbbyte links, das zweite rechts steht, genau in der eingegebenen Reihenfolge. Zur Übung "rechnen" wir noch einmal schnell aus, was 1 C im Binärsystem heißt!

Hier noch ein wichtiger Hinweis !!
Es kann vorkommen, daß in einem System oder Programm Hexadezimalzahlen und Dezimalzahlen gleichzeitig verwendet werden. Dabei kann es zu Verwechslungen kommen:

- Die Zahl 10 im Hexadezimalsystem ist, wie wir wissen, eigentlich die "16".
- Die Zahl 10 im Dezimalsystem ist ein ganz anderer Wert. Deshalb hat man festgelegt, daß Hexadezimalzahlen grundsätzlich mit einem großen H am Ende abgeschlossen werden, also z. B.:

1 2 4 0 H

In unserem Heft wird auf diese Darstellungsweise verzichtet, wir meinen immer Hexadezimalzahlen - auch unser LC-80 ist nur Hexa-Zahlen "gewöhnt".

Unser Computer zeigt jetzt 6 Stellen an, wir wiesen nun schon, was das heißt:

[2 0 0 0 1.C.]

Auf der Adresse 2000 steht die Information 1 C. Leider ist das noch kein Programm, das besteht aus viel mehr Befehlen. Wie tragen wir nun diese ein?

Auf unserem Rechner gibt es zwei Tasten

[+] und [-]

Diese dienen nicht, wie man vielleicht vermuten könnte, zum Addieren oder Subtrahieren, sondern werden z. B. zur Veränderung der Adresse nach oben bzw. unten benutzt.

Wir drücken

[+]

und schon wird die Adresse 2001 in den ersten vier Stellen des Displays angezeigt.

Die Punkte in den beiden Datenstellen zeigen, daß wiederum Daten eingetragen werden können. Wenn vorher schon welche dort stehen, werden sie durch unsere Hexa-Tasten einfach überschrieben, z. B. mit

[D] und [F]

Durch Betätigen von

[+]

erreichen wir die nächste Adresse, in die wir die Daten eintragen können.

Jedesmal, wenn wir [+] und [-] drücken, verändern wir die Adresse, in die Daten eingeschrieben werden können. Die in die vorherige Adresse eingegebenen Daten werden dadurch "gültig", d. h. jetzt hat sie der Rechner gespeichert.

Wir merken uns, daß unser Rechner für jede Adresse zwei Hexa-Eingaben braucht, die mit der Betätigung von + oder - abgeschlossen werden.

Das wollen wir jetzt einmal mit einem kleinen Programm üben:

Taste	Anzeige	Beschreibung
RES	LC-80	Rücksetzen in den Anfangszustand
ADR	2.0.0.0.X X	Setzen auf Anfangsadresse 2000, Daten sind noch von "früher" drin
DAT	2 0 0 0 X.X.	Punkte zeigen an, daß Daten eingegeben werden können
C	2 0 0 0 0.C.	"C" wird eingetragen
D	2 0 0 0 C.D.	"D" wird eingetragen
+	2 0 0 1 X.X.	Adresse wird um 1 erhöht, Rechner wartet auf neue Daten
E	2 0 0 1 0.E.	Eintragen "E"
A	2 0 0 1 E.A.	Eintragen "A"
+	2 0 0 2 X.X.	Adresse erhöhen
0	2 0 0 2 0.0.	Eintragen "0"
4	2 0 0 2 0.4.	Eintragen "4"
+	2 0 0 3 X.X.	Adresse erhöhen
7	2 0 0 3 0.7.	Eintragen "7"
6	2 0 0 3 7.6.	Eintragen "6"
+	2 0 0 4 X.X.	Adresse erhöhen

Unser kleines Programm ist fertig, später werden wir das so schreiben:

Adresse	Daten
2 0 0 0	C D
2 0 0 1	E A
2 0 0 2	0 4 *
2 0 0 3	7 6
2 0 0 4	X X

* Wenn dieses Sternchen auftaucht, muß überprüft werden, ob unser LC-80 mit zwei ROMS U 505 D oder mit einem EPROM U 2716 C ausgerüstet ist (sh. a. Bedienungsanleitung 5.58).

Und was haben wir nun davon?
Unser kurzes Programm ist nun im Speicher. Wenn es funktionieren soll, müssen wir folgendes tun:

1. RES Rücksetzen
2. ADR Anfangsadresse aufrufen
3. EX Execute-Taste drücken.

Nachdem wir uns von unserer Überraschung erholt haben, wollen wir uns merken:
Bevor unser Rechner ein Programm abarbeitet, muß er auf dessen Anfangsadresse eingestellt werden.
Das kann natürlich auch eine andere als 2000 sein, wenn unser Programm dort beginnen sollte (was wir allein festlegen). Dann muß nach ADR diese Startadresse eingegeben werden. Da hierbei die vier Punkte der Adressenanzeige leuchten, kann wieder über die Hexa-Tasten die gewünschte Adresse eingegeben werden, wie wir das von der Dateneingabe gewohnt sind.

Nun wieder zurück zu unserem Programm. Diese lustige Musik ist natürlich nur mit einem Trick so einfach zu programmieren. Wir haben nämlich die komplette Anfangsmusik der Einschaltroutine übernommen. Unser Programm macht dabei folgendes: In der Adresse 2000 steht der Befehl "CD", ein sehr leistungsfähiger Befehl, der dem Rechner den Auftrag gibt, zu der nach CD eingetragenen Adresse 0 4 E A zu springen. Wir erkennen, daß CD der eigentliche Befehl ist (auf Adresse 2000), während die auf den Adressen 2001 und 2002 stehenden Daten eine Zieladresse bezeichnen.
Dabei fällt auf, daß diese Zieladresse (4stellig!) in zwei Etappen gespeichert wird und zwar in umgekehrter Reihenfolge:

Bei Adresse 0 4 E A wird also zunächst E A und danach 0 4 abgespeichert. Auf diesem Speicherplatz beginnt das Anfangsmusikprogramm "Popcorn", welches nun vom Rechner aufgerufen und folgerichtig abgearbeitet wird.
Damit ist die Wirkung des Befehls CD aber noch nicht erschöpft. Bevor 'gesprungen" wurde, hat sich der Rechner die Adresse "gemerkt", bei der er das normale Programm verlassen hatte, nämlich 2002. Nach Abarbeitung des Musikprogrammes springt er wieder dorthin zurück und macht bei 2003 einfach weiter. Der dort stehende Befehl 7 6 in unserem Beispiel ist der Befehl "Halt", d. h. die Programmabarbeitung wird dort angehalten, was durch die rote Leuchtdiode angezeigt wird. Unser Programm läßt sich wieder mit RES , ADR , EX starten.

Nun wollen wir lernen, bestehende Programme zu ändern oder dort Fehler zu beseitigen. Nehmen wir an, wir wollten unser Lied nicht nur einmal spielen lassen, sondern sozusagen im Dauerbetrieb. Was ist zu tun?
Zunächst funktionierte der Teil bis zum Spielen der Anfangsmelodie recht gut, den wollen wir also so lassen (bis Adresse 2002). In die Adressen 2002 bis 2005 wollen wir neue Befehle eintragen. Wir drücken also RES , ADR , + , + , + ,
C , 3 , + , 0 , 0 , + , 2 , 0

Adresse	Daten	Kommentar
2000	C D	Springe auf Adresse
2001	E A	0 4 E A und (danach auf 2003)
2002	0 4	
2003	C 3	Springe auf Adresse
2004	0 0	2000 (und mache dort
2005	2 0	weiter)
2006	X X	

RES , ADR , EX

Es geht los - und hört nicht mehr auf. Warum?
Wir haben zwei Sprungbefehle benutzt, der eine (C D) bewirkt einen Sprung zu einem sogenannten Unterprogramm (Lied) und wieder zurück, der andere einen Sprung zum Anfangspunkt (Adresse 2000) unseres Programmes. Damit ist eine Schleife geschlossen, der Rechner macht unentwegt Musik - bis wir RES drücken und den Urzustand wieder herstellen. Das Programm bleibt trotzdem erhalten, wie uns das Drücken von ADR und EX beweist.
Diese Programmänderung haben wir durch schrittweises Erhöhen der Adresse (+) erreicht. Insbesondere bei längeren Programmen ist das mühsam. Einfacher geht es dann durch gezielte Eingabe der zu ändernden Adresse.
Wir wollen das üben:

RES
ADR

Der Rechner zeigt 2.0.0.0.X X an.
Die Punkte signalisieren, daß Adressen eingegeben werden können. Durch Drücken unserer Hexa-Tasten geben wir jetzt die zu ändernde Adresse ein:

2 0 0 3

Jetzt wollen wir ändern, d. h. neue Daten eintragen, dazu müssen wir

DAT

drücken. Die Punkte im Datendisplay zeigen, daß der Rechner dazu bereit ist. Wir geben ein

7 6

also den Haltbefehl wie im ersten Beispiel. Nach

RES ADR und EX

spielt unser Rechner das Lied wieder nur einmal.

Wir beherrschen jetzt schon fast das ganze Tastenfeld, nur NMI, ST und LD fehlen noch. Zunächst nur soviel:

NMI dient zur Unterbrechung von Programmen

ST dient zum Speichern von Programmen auf Kassetten und

LD zum Übernehmen von Kassettenprogrammen in den Rechner.

Die letzten beiden Funktionen werden im übernächsten Kapitel behandelt.

2. Wir programmieren ein Lied

Unser Rechner spielt beim Einschalten ein Lied. Das ist immer dasselbe, weil es von den Konstrukteuren des LC-80 so festgelegt wurde.

Wir wollen jetzt ein eigenes Lied "schreiben". Wie wäre es mit

"Horch, was kommt von draußen rein....." ?

Es wäre für uns schwierig, mit unseren jetzigen Kenntnissen so etwas "von Grund auf" zu programmieren. Glücklicherweise haben die Konstrukteure des LC-80 auch hier schon Erleichterungen vorgesehen. Schon bei unserem ersten Programmbeispiel haben wir ein sog. Unterprogramm benutzt, in das wir einfach hineingesprungen sind und uns damit viel Arbeit gespart haben. Für unser Lied brauchen wir das Unterprogramm "MUSIK", das auf der Adresse 04EE* beginnt. Damit ist es möglich, sehr einfach eine Reihe von Tönen mit Tonhöhe und -länge zu programmieren. Jeder Ton wird auf zwei Speicheradressen geschrieben und besteht aus 2 Bytes:

1. Byte = Tonhöhe (erlaubt 00 ... 1F)
2. Byte = Tonlänge (erlaubt 00 ... FF).

Eine Pause wird ebenfalls auf zwei Speicheradressen geschrieben und besteht auch aus zwei Bytes:

1. Byte = 20
2. Byte = Pausenlänge (wie bei Tönen).

Wenn unser Musikstück enden soll, steht am Programmende "80", wenn es ständig wiederholt werden soll "40".

Im folgenden finden wir eine Zusammenstellung von Noten, deren Lage auf der Klaviertastatur und den dazugehörigen Code für den LC-80. Ebenfalls eine Tabelle zeigt uns die Codierung der Notenlängen für ausgewählte Tonlängen.

Abschließend unser Lied in Notenform. Wer halbwegs Schulwissen auf musikalischem Gebiet hat, kann nun Musik machen.

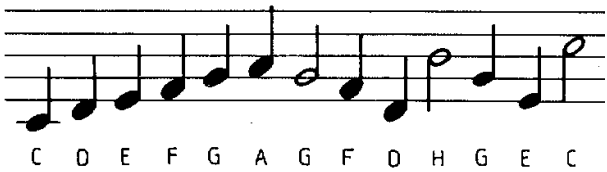
Tonlängen (Beispiel)

Code	Länge	
0 4	1/8 Note	**e**
0 8	1/4 Note	**q**
1 0	1/2 Note	**h**
2 0	1/1 Note	**o**

20 - Pause
40 - Wiederholung
80 – Ende

Frequenztabelle

Code	Ton	Klaviertastatur
	A	
1 F	Gis	
1 E	G	
1 D	Fis	
1 C	F	
1 B	E	
1 A	Dis	
1 9	D	
1 8	Cis	
1 7	C	
1 6	H	
1 5	Ais	
1 4	A	
1 3	Gis	
1 2	G	
1 1	Fis	
1 0	F	
0 F	E	
0 E	Dis	
0 D	D	
0 C	Cis	
0 B	C	
0 A	H	
0 9	Ais	
0 8	A	
0 7	Gis	
0 6	G	
0 5	Fis	
0 4	F	
0 3	E	
0 2	Dis	
0 1	D	
0 0	Cis	
	C	



Nun zum eigentlichen Programm.
Den ersten Teil schreiben wir zunächst einmal auf, auch wenn wir ihn noch nicht verstehen. In Zukunft schreiben wir dabei zusammengehörige Daten in eine Reihe, z. B. den Befehl und die zugehörige Zieladresse.
Beim Eingeben müssen wir beachten, daß trotzdem nach jeweils zwei Zeichen [+] gedrückt wird. Außerdem dürfen wir nie vergessen, den Rechner zuerst in den Grundzustand [RES] zu versetzen, [ADR] aufzurufen, [DAT] zu drücken und dann erst Daten einzugeben!

Adresse	Daten	Erläuterungen
2 0 0 0	FD 21 10 20	1. Programmteil
2 0 0 4 ♦	CD EE 04 *	
2 0 1 0	0B 08	1. Ton "C"
2 0 1 2	0D 08	"D"
2 0 1 4	OF 08	"E"
2 0 1 6	10 08	"F"
2 0 1 8	12 08	"G"
2 0 1 A	14 08	"A"
2 0 1 C	12 10	"G"
2 0 1 E	10 08	"F"
2 0 2 0	OD 08	"D"
2 0 2 2	16 10	"H"
2 0 2 4	12 08	"G"
2 0 2 6	OF 08	"E"
2 0 2 8	17 10	"C"
2 0 2 A	80	Liedende
2 0 2 B	FF	

♦ Die Lücke zwischen den Adressen 2 0 0 6 und 2 0 1 0 stört uns zur Zeit noch nicht.

* ROM-Bestückung beachten ! (S. 58 der Bedienungsanleitung)

Wie immer starten wir unser Programm mit

RES ADR EX

Na, funktioniert es?
Wenn ja, können wir jetzt beliebige Melodien programmieren. Viel Spaß!
Gefällt uns eine unserer Melodien besonders gut, können wir sie ja zur Übung auf Kassette speichern. Dazu lesen wir uns das Kapitel 3 erst einmal gründlich durch. Sollte uns ein Ton oder eine Tonlänge einmal "falsch" vorkommen, "reparieren" wir unser Programm selbständig, indem wir

RES ADR

drücken und danach die Adresse eintragen, deren Inhalt zu ändern ist. Danach

DAT

drücken und jetzt die neuen Daten eintragen. Nach

RES ADR EX

können wir uns davon überzeugen, ob wir mit unserer Änderung Glück hatten.

3. Der Kassettenanschluß

Wir haben im vorigen Kapitel die Grundlagen der Bedienung unseres LC-80 kennengelernt und schon erste kleine Programme geschrieben. Dabei sind wir zunächst nicht weiter gekommen als bis zur Adresse 2 0 2 B. Unser Rechner bietet in der Grundausstattung den scheinbar riesigen Speicherbereich

bis zur Adresse 2 3 F F , das sind 1024 Byte (davon sind allerdings die letzten 66 Byte nicht von uns benutzbar, sie werden für interne Aufgaben im Rechner benötigt). Das reicht uns zunächst, außerdem wollen ca. 1000 Befehle erst einmal eingegeben werden - eine mühsame Arbeit! Sie erfordert erhebliche Konzentration, um fehlerfrei durchgeführt zu werden. Hat man ein solch umfangreiches Programm erst einmal geschrieben und funktioniert es dann auch noch, wäre es sehr schade, wenn es verloren ginge. Das ist aber der springende Punkt.
Wenn unser Rechner ausgeschaltet wird, sind alle von uns eingeschriebenen Programme verloren!
Das liegt daran, daß unsere Speicherbausteine (RAMS) natürlich nur dann funktionieren, wenn die Stromversorgung gewährleistet ist. Aber auch ohne diese Eigenart unseres Rechners wären wir nicht sehr glücklich, da wir ja schon morgen wieder neue Programme "erfinden" wollen, die dann die jetzt besetzten Speicherplätze belegen würden. Aber unser Rechner bietet auch dafür eine elegante Lösung. Mit einem ganz normalen Kassettenrecorder oder Tonbandgerät können wir unsere Programme "retten" und für spätere Zeiten aufheben.
Wie funktioniert das?
Zunächst einmal zum Anschluß des Kassettenrecorders.
Wir benötigen ein normales Überspielkabel (alle Anschlüsse sind "über Kreuz" miteinander verbunden: also 1 des einen Steckers mit 3 des anderen Steckers, 3 mit 1, 4 mit 5, 5 mit 4 und 2 mit 2) und ein Kassettengerät, das für Aufnahme und Wiedergabe geeignet ist. Dessen Aufnahme- und Wiedergabebuchse wird mit der Diodenbuchse des Rechners über das Überspielkabel verbunden. Jetzt den Kassettenrecorder einschalten - fertig!

Wir schreiben jetzt unser erstes Programm wieder in den Rechner ein:

Adresse	Daten
2 0 0 0	CD
2 0 0 1	EA
2 0 0 2	04
2 0 0 3	76
2 0 0 4	XX

RES ADR EX

Geht's noch ?

Dieses Programm wollen wir uns aufheben. Dazu drücken wir:

RES ST → STORE (englisch: speichern)

Unser Rechner zeigt an:

X.X.X.X.- F

Das F kommt von FILE, d. h. in der Rechnersprache soviel wie "Name". Unser Programm soll also einen Namen bekommen, wir wollen es A001 nennen. Das sagen wir dem Rechner mit:

A 0 0 1

und er zeigt das an:

A.0.0.1.- F

Wir drücken nun:

+

und es erscheint:

X.X.X.X.- S

S heißt Startadresse, d. h. wir müssen dort die erste Adresse unseres Programmes (2000) eingeben:

2 0 0 0

Das Display zeigt jetzt:

2.0.0.0.- S

Wir betätigen jetzt wieder

+

und es erscheint:

X.X.X.X.- E

E bedeutet Endadresse, die bei uns 2004 ist (bei Adresse 2003 sind noch Daten!), also

2 0 0 4

Der Rechner zeigt jetzt an:

2.0.0.4.- E

Damit sind wir zunächst fertig. Jetzt muß das Kassettengerät auf Aufnahme geschaltet werden, wir müssen dafür natürlich eine Kassette eingelegt haben. Wenn das alles geht, muß noch der Aufnahmepegel eingestellt werden, falls wir kein Automatikgerät haben. Schließlich beachten wir noch, daß schon Magnetband am Tonkopf vorbeiläuft und nicht etwa das Vorspannband.
Alles fertig?
Wir drücken jetzt

EX

Es ertönt ein Pfeifton von einigen Sekunden und danach eine recht unmelodische Musik - das sind unsere Daten, die der Rechner in Töne umsetzt und an das Kassettengerät übergibt. Nach Beendigung der Übertragung erscheint auf der vorher dunklen Anzeige

2 0 0 4 F.F.

Das ist unsere Endadresse.
Jetzt können wir unser Bandgerät stoppen, zurücklaufen lassen und auf Wiedergabe schalten. Wir wollen uns die Sache einmal anhören. Wenn die vorher gehörten Tonfolgen auf dem Band sind, ist soweit alles in Ordnung. Wir lassen das Band nun wieder in Ausgangsstellung zurückspulen.
Nun wird es spannend, denn jetzt wollen wir wissen, ob unser Rechner sich auch von unserer Kassette "programmieren" läßt. Dazu lassen wir ihn zunächst alles wieder "vergessen", indem wir ihn für ca. 10 sec ausschalten (Netzstecker). Nach dem Einschalten spielt er die Anfangsmelodie und zeigt den Begrüßungstext, das ist das Zeichen, daß alles im RAM gelöscht ist.

Jetzt drücken wir:

LD

Der Rechner zeigt

X.X.X.X.- F

an, d. h. er wartet auf einen FILE-Namen, in unserem Fall war das A001.
Das geben wir nun ein:

A 0 0 1

und in der Anzeige steht jetzt:

A.0.0.1.- F

Danach drücken wir die Ausführungstaste

EX

Jetzt starten wir unser Kassettengerät in Wiedergabefunktion (nachdem wir vorher zurückgespult hatten!).

Die unmelodische Tonfolge muß jetzt sowohl aus dem Kassettenlautsprecher als auch aus dem Lautsprecher unseres LC-80 kommen.
Wenn der Rechner alles versteht, erscheinen jetzt in der Anzeige folgende Angaben nacheinander:

. . . . . .	Der Rechner "hört" irgendetwas.
A.0.0.1.- F	Der FILE-Name wurde gefunden.
' ' ' ' ' '	Jetzt werden die Daten "gelesen".
2 0 0 4 F.F.	Die Endadresse 2004 wurde erreicht.

Wenn alles so funktioniert, steht unser Programm jetzt wieder im Speicher, wir können das mit

RES ADR EX

ausprobieren, die Anfangsmelodie wird einmal gespielt. Wurden Daten verfälscht, erscheint in der Anzeige "ERROR" (Irrtum) und wir müssen den Fehler suchen.

Was haben wir gelernt?
Wir können unsere eigenen Programme nun auf Kassette speichern, dort unbegrenzte Zeit aufheben und bei Bedarf wieder in unseren Rechner laden. Beim Übernehmen auf Kassette machen wir folgendes:

ST

FILE-Namen eingeben

+

Start-Adresse eingeben

+

Endadresse eingeben (Das soll in Zukunft immer eine Adresse weiter sein als die des letzten Befehls)
Bandggerät auf Aufnahme schalten und ggf. starten

EX

Anzeige wird dunkel.
Wenn die Anzeige die Endadresse anzeigt, Bandgerät ausschalten.

Wenn wir ein auf Kassette gespeichertes Programm in den Rechner übernehmen wollen, läuft das so ab:

LD

FILE-Namen eingeben

EX

Bandgerät auf Wiedergabe schalten und ggf. starten.
Wenn die Anzeige die Endadresse anzeigt, Bandgerät ausschalten.

Sind später mehrere Programme auf der Kassette, ist der FILE-Name ein wichtiges Hilfsmittel, das von uns gewünschte Programm in den Rechner zu laden. Es werden zwar alle empfangenen FILE-Namen vom Rechner kurz angezeigt, wenn es aber nicht der "richtige" ist, erscheinen auf dem Display 6 Striche

- - - - - -

und er sucht weiter.
Eine eigene Kontrollautomatik überprüft, ob alle Daten des gesuchten Programmes richtig übernommen wurden, wenn nicht, wird

E R R O R

angezeigt.

Die Übertragung eines Programmes von 1000 Bytes benötigt ca. 90 sec, danach haben wir in einer Kassette schon einen "Riesenspeicher" zur Verfügung,. mit dem wir unter Umständen Hunderte Programme aufheben können.

4. Die Alarmsirene

Nicht nur Lieder, sondern auch andere akustische "Äußerungen" lassen sich dem LC-80 entlocken. Im Unterprogramm "SOUND" wird, ähnlich wie bei "MUSIK", Tonhöhe und -länge, allerdings hier nur für einen Ton, erzeugt.
Hier nun wollen wir etwas tiefer "einsteigen".
Die Startadresse des Unterprogrammes "SOUND" befindet sich bei 0376*. Im ersten Kapitel haben wir schon den Befehl CD kennengelernt, der zum Aufrufen von Unterprogrammen dient. Diesen Befehl brauchen wir jetzt wieder.
Im Unterprogramm "SOUND" wird die Tonhöhe durch den Inhalt des Registers C bestimmt, der Inhalt der Register H und L gibt die Anzahl der auszuführenden Tonschwingungen an.
Was sind Register?
Das sind interne Arbeitsspeicher unseres Mikroprozessorschaltkreises U 880 D (sh. a. Abschnitt 8), in denen Daten, Adressen, Befehle usw., auch Rechenergebnisse, für kurze Zeit "abgelegt" und dann für weitere Operationen abgerufen werden. Für unsere Aufgabe brauchen wir diese Register jetzt erstmalig bewußt. Wenn sich dort Informationen für unseren Ton befinden sollen, wie kommen die dann dorthin?
Natürlich wieder durch Befehle. Wir müssen aus dem Befehlssatz unseres Mikroprozessors zunächst einen aussuchen, der uns die Tonhöhe in das Register 0 einschreibt.
Dieser Befehl müßte etwa heißen:

Lade Register C mit der Konstanten n

In einer verkürzten Schreibweise, die aus dem Englischen stammt, heißt er:

LD C,n

Diese Schreibweise wird als "Mnemonik" bezeichnet und ist in der Programmerstellung überall verbreitet. Auch wir wollen uns langsam an diese "Sprache" gewöhnen. Wenn wir in der Befehls-liste unseres U 880 D [4] nachschauen, finden wir unter dieser "Mnemonik" auf S.82 den Befehl

0E n.

Dieser Befehl wird auf S. 10 ausführlich erläutert, dort heißt er allgemein LD r, n. Dabei ist r ein beliebiges Register - auf S. 6 wird erklärt, wie die einzelnen Register in diesen Befehl eingesetzt werden können.
0E ist also der Befehl zum Laden des Registers C und n die 1Byte-Konstante (Operand), die wir dort hineinbringen wollen. Die Schreibweise 0E n wird vom Rechner direkt verstanden und heißt:

"OP-Code" .

abgeleitet von Operationscode.

Sehr viele neue Begriffe?
Wir werden das bald beherrschen und üben weiter!
In den Registern H und L soll die Anzahl der auszuführenden Tonschwingungen (Tonlänge) untergebracht werden, also:

Lade die Register H und L mit der Konstanten nn

Die beiden Register H und L sollen hier gemeinsam benutzt werden, deshalb sind 2 Bytes notwendig, was durch nn ausgedrückt werden soll.
In Mnemonikschreibweise lautet dieser Befehl:

LD HL, nn

Wir suchen in der Befehlsliste [4] auf S. 32 den OP-Code:

21 nn

heraus und studieren auf S. 16 die Wirkung des allgemeinen Befehls LD dd, nn.
Diese Arbeitsweise wollen wir in Zukunft beibehalten und so nach und nach die wichtigsten Befehle und deren Wirkung kennenlernen.
Jetzt wollen wir ein einfaches Programm aufstellen und dabei in unserer Tabelle die neuen Schreibweisen verwenden:

Adresse	OP-Code	Mnemonik	Kommentar
2000 2001	0E 25	LD C, 25	Lade Register C mit der Konstanten 25 (Tonhöhe)
2002 2003 2004	21 00 04	LD HL, 0400	Lade das Doppelregister HL mit der Konstanten 0400 (Tonlänge)
2005 2006 2007	CD 76 03	CALL 0376 *	Rufe das Unterprogramm "SOUND", dessen Startadresse 0376 ist und springe nach Abarbeitung auf Adresse 2008
2008	76	HALT	Halt!
2009	FF		

Wir beachten, daß immer dann, wenn der Operand aus 2 Byte besteht, zuerst der niederwertige Teil und danach der höherwertige Teil eingetragen werden muß. Wenn also die Konstante 0400 geladen werden soll, wird zuerst 00 und danach 04 programmiert, bei Aufruf des Unterprogrammes "SOUND" analog zuerst der niedere Adreßteil 76 und dann erst 03.

Unser kurzes Programm ist im Rechner, durch

RES ADR EX

wird es abgearbeitet - ein kurzer Piepton und Aufleuchten der "HALT-Leuchtdiode".

Durch gezielte Änderungen des Programmes verändern wir jetzt die Tonhöhe und Tonlänge, bis wir verstanden haben, wie alles funktioniert.

Wir stellen fest:

Tonhöhe: 01 ist ein sehr hoher Ton,
FF ein sehr tiefer Ton,

Tonlänge: 0001 = kurzer Ton,
FFFF = sehr langer Ton.

Die Tonlänge ist nicht wie beim Unterprogramm "MUSIK" ein fester Wert, sondern von der Tonfrequenz abhängig, da ja hier nur die Anzahl der Schwingungen programmiert wird. Und die vorgegebene Anzahl der Schwingungen ist bei einem sehr hohen Ton sehr schnell ausgeführt.

Wie nun aber weiter, eigentlich wollten wir ja eine Alarmsirene "bauen", Unser kurzer Ton ist uns zu wenig - Alarmsirenen haben veränderliche Töne ...

Wie können wir unseren Ton ständig verändern ... ?

Wir machen ihn zunächst einmal sehr kurz, z. B. 0001 in HL.

Wie das geht, wissen wir schon. Dann müßten wir eine Schleife programmieren, bei deren Durchlauf der Wert für die Tonhöhe (steht im Register C) ständig erhöht wird.

Gibt es solche Befehle?

Natürlich, unser Mikroprozessor kann das! Und zwar mit

INC C

Auf S. 81 und 36 von [4] können wir uns über diesen Befehl informieren, der im Klartext etwa heißen wird:

Inkrementiere (also erhöhe) den Inhalt von C um 1

Wir finden auch den OP-Code dieses Befehls:

0C

Jetzt, nachdem wir das alles wissen, basteln wir unser neues Programm zusammen:

Adresse	OP-Code	Mnemonik	Kommentar
2000 2001	0E 25	LD C, 25	Lade Register C mit der Konstanten 25 (Tonhöhe)
2002	0C	INC C	Erhöhe den Inhalt von C um 1
003 2004 2005	21 00 04	LD HL, 0001	Lade Doppelregister HL mit 0001
2006 2007 2008	CD 76 03	CALL 0376	Springe zum Unterprogramm "SOUND" auf Adresse 0376 und danach auf Adresse 2009
2009 200A 200B	C3 02 20	JMP 2002	Springe auf Adresse 2002
200C	FF		

Was macht der Rechner nun damit?

1. Sein Register C wird mit 25 geladen.
2. Dessen Inhalt wird um 1 erhöht, steht also jetzt auf 26.
3. Das Doppelregister HL wird mit 0001 (sehr kurzer Ton) geladen.
4. Das Unterprogramm "SOUND" wird aufgerufen, der programmierte Ton wird ausgestrahlt und das Programm auf Adresse 2009 fortgesetzt.
5. Der dort eingetragene Befehl führt zu einem Sprung nach Adresse 2002.
6. Hier steht der Befehl "Erhöhe C um 1", der Inhalt von C wird damit 27 und so geht das immer weiter, bis die Tonhöhe FF erreicht ist, ein sehr tiefer Ton. Dann geht es wieder mit 00 los

Eigentlich sehr lustig, unsere Sirene, nicht wahr?
Wir können jetzt sofort auch einen anderen "Sound" programmieren, vielleicht eine Sirene mit aufsteigendem Ton?
Dazu verwenden wir den Befehl:

DEC C

oder im OP-Code

OD

auf Adresse 2002.
Das heißt

Dekrementiere (erniedrige) den Inhalt von C um 1

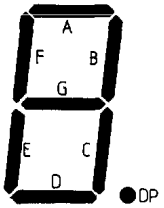
Also los!

Für Sound-Enthusiasten hier noch ein kurzes Programm ohne Kommentar:

```
2000    06 15       LD B, 15
2002    C5          PUSH BC
2003    0E 30       LD C, 30
2005    21 13 00    LD HL, 0013
2008    CD 76 03    CALL SOUND
200B    OE 40       LD C, 40
200D    21 11 00    LD HL, 0011
2010    CD 76 03    CALL SOUND
2013    0E 50       LD C, 50
2015    21 0F 00    LD HL, 000F
2018    CD 76 03    CALL SOUND
201B    C1          POP BC
201C    10 E4       DJNZ E4
201E    76          HALT
201F    FF
```

5. Unser Display

Bis jetzt haben wir Töne erzeugt - eine interessante Anwendung des LC-80. Aber auch "optisch" kann man eine ganze Menge mit ihm anfangen. Wir benutzen dazu das Display, das uns ja schon als Adressen- und Datenanzeige bekannt ist. Zur Anzeige von Adressen und Daten werden Unterprogramme des Betriebssystems genutzt. Wir wollen jedoch die Möglichkeiten des Displays voll ausnutzen und die Technik der Darstellung von Zeichen kennenlernen.
Unsere Anzeige besteht aus 6 Stellen (auch Digits genannt), von denen jede 7 Leuchtbalken (Segmente) und einen Dezimalpunkt besitzt. Wir kennen diese sog. 7Segment-Darstellung von unserer Digitaluhr und vom Taschenrechner her. Für die Bezeichnung der 7 Segmente hat sich folgendes Schema "eingebürgert":



In unserem LC-80 steuert ein spezieller Schaltkreis, die sog. System-PIO (sh. a. Abschnitt 8.5.) die Anzeige an. Dabei ist der Port A zuständig für die Segmente und Port B für die Auswahl der Digits. Was das alles im einzelnen ist, werden wir später behandeln.
Für uns ist jetzt wichtig, wie die einzelnen Segmente an den PIO-Port A angeschlossen sind.
Die folgende Tabelle zeigt das:

PIO-Anschluß	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0
Segment	D	E	C	DP	G	A	F	B

Analog die nächste Tabelle für die Verteilung der Digits:

PIO-Anschluß	B 7	B 6	B 5	B 4	B 3	B 2	B 1	B 0
Digit	1	2	3	4	5	6		
Funktion	Adressen				Daten			

Trotz der Tabellen sind wir noch nicht viel schlauer - aber wir brauchen sie!

Wie stellen wir nun eine Ziffer (oder ein Zeichen) auf unserem Display dar und vor allem, wie können wir das unserem Rechner mitteilen?

Fangen wir mit der ersten Tabelle an. Eine logische "1" auf dem entsprechenden PIO-Ausgang läßt das Segment leuchten, bei einer "0" bleibt es dunkel. Wenn wir im einfachsten Fall alle Segmente und den Dezimalpunkt leuchten lassen wollen, müssen wir also überall "1" eintragen. Wir schreiben das analog unserer obigen Segmenttabelle so:

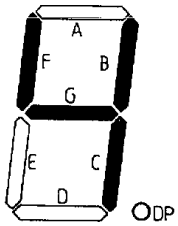
D	E	C	DP	G	A	F	B
1	1	1	1	1	1	1	1

Jetzt müssen wir unsere Kenntnisse im Hexadezimalsystem anwenden. Sollte das nicht mehr klappen, schauen wir nochmal im Kapitel 1 nach!

Wir haben ermittelt, daß unsere Zahl im Hexa-Code FF heißt. Zur Übung versuchen wir es einmal mit der "4".

Welche Segmente leuchten bei der "4"?

Wir zeichnen uns das am besten einmal auf:



und erkennen die Segmente B, C, F und G. Daraus stellen wir unsere Tabelle auf:

D	E	C	DP	G	A	F	B
0	0	1	0	1	0	1	1

Hexadezimal heißt das 2B.
Wir üben weiter und ermitteln den Hexa-Code für die 7Segment-Darstellung folgender Zeichen:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
aber auch für:
H, L, P, U und vor allem für ein dunkles Feld.
Das war bisher alles sehr theoretisch und wir wollen deshalb jetzt überprüfen, ob wir damit etwas anfangen können. Zuerst schreiben wir ein kleines Programm auf, welches uns erlaubt, einzelne Zeichen auf dem Display darzustellen.

```
2000        DD 21 00 21     LD IX, (2100)
2004        CD 5A 04 *      CALL DAK 1
2007        C3 00 20        JMP 2000
200A        FF
```

Der erste Befehl bewirkt die Übernahme der Daten des Speicherplatzes 2100 in das Indexregister IX. Das Unterprogramm DAK 1 dient zur Anzeige dieser Daten und - das ist besonders wichtig - auch der in den nachfolgenden 5 Speicherplätzen abgelegten Daten.
Dabei steht in:

2100 der Hexa-Code der letzten Stelle
2101 " 5. Stelle
2102 " 4. Stelle
2103 " 3. Stelle
2104 " 2. Stelle
2105 " 1. Stelle

Wenn wir Zeichen unserer Wahl darstellen wollen, müssen wir in diese Speicherstellen etwas eintragen.
Wie wär's mit:

```
2100   00
2101   E7
2132   02
2103   02
2104   6F
2105   6B
```

Wenn wir alles verstanden haben, probieren wir gleich aus:

```
--PAPA     012345     6789--
```

Durch eine kleine Programmänderung lassen wir unseren Text jetzt "laufen". Dazu geben wir erst den Text ein und beginnen damit bei der Adresse 20DA:

20DA	00	20E2	4F	20EA	6F	20F2	4F	20FA	6B
20DB	00	20E3	E7	20EB	6B	20F3	00	20FB	00
20DC	00	20E4	00	20EC	00	20F4	00	20FC	00
20DD	00	20E5	00	20ED	00	20F5	00	20FD	00
20DE	00	20E6	00	20EE	20	20F6	E7	20FE	00
20DF	00	20E7	E7	20EF	6F	20F7	C2	20FF	00
20E0	00	20E8	C2	20F0	4F	20F8	C2	2100	00
20E1	6F	20E9	C2	20F1	6F	20F9	6F		

Unser Rechnerprogramm sieht dann so aus:

2000	0E 22	LD C, 22
2002	DD 21 FB 20	LD IX, (20 FB)
2006	06 10	LD B, 10
2008	CD 83 04 *	CALL DAK 2
200B	10 FB	DJNZ FB
200D	DD 2B	DEC IX
200F	0D	DEC C
2010	20 F4	JRNZ F4
2012	C3 00 20	JMP 2000
2015	FF	

Zum Programm folgende kurze Erklärungen:

- Register C wird mit 22 geladen.
- Doppelregister IX mit 20 FB laden.
- Register B mit 10 laden.
- DAK 2 wird aufgerufen. Es bewirkt, daß in der 6. Stelle der Inhalt von 20 FB abgebildet wird, in der 5. Stelle der Inhalt von 20 FC usw. bis zur ersten Stelle (2100)
- DJNZ ist ein komplizierter Befehl:
  Register B wird um 1 erniedrigt; wenn es danach ≠0 ist, wird um die Distanz FB (rückwärts) zur Adresse 2008 gesprungen, bei =0 geht es weiter zu 2000.
- Der Inhalt von IX wird um 1 erniedrigt, das soll eine Verschiebung um eine Stelle im Display bewirken.
- Register C wird ebenfalls um 1 erniedrigt.
- Ist C ≠ 0, Sprung um Distanz F4 (zu Adresse 2006), wenn C = 0 (das bedeutet, daß alle Zeichen "durch" sind), wird bei 2000 neu begonnen.

Wenn alles funktioniert, können wir auch eigene "Texte" eingeben.

In unserem Programm haben wir erstmals relative Sprünge benutzt, so z. B. die Befehle DJNZ und JRNZ. Sie bewirken Sprünge nicht wie bisher auf festgelegte Adressen, sondern um Distanzen vorwärts oder rückwärts.
Wie geht das?
Nach dem eigentlichen Befehl (z. B. DJNZ) folgt noch eine Angabe (z. B. FE). Allgemein wird diese Distanz mit e bezeichnet, dabei kann e "negativ" oder "positiv" sein, je nachdem, ob zurück oder nach vorn gesprungen. werden soll.
Fangen wir mit Vorwärtssprüngen an:
Angenommen, der Befehl DJNZ steht auf Adresse 200B und wir wollen auf Adresse 2012 springen.
Dann sieht unser Programm so aus.

Adresse	OP-Code	relativer Sprung	
2 0 0 B	10	DJNZ	
2 0 0 C	05	Distanz 05	
2 0 0 D	XX		00
2 0 0 3	XX		01
200F	XX		02
2 0 1 0	XX		03
2 0 1 1	XX		04
2 0 1 2	Zieladresse		05

Wir zählen einfach byteweise vorwärts und fangen damit nach Befehl (10) und Distanz (05) auf Adresse 200D mit 00 an. Rückwärts geht es genauso, wobei die Zählweise 00, FF, FE, FD usw. ist:

Adresse	OP-Code	relativer Sprung	
2 0 0 6	Zieladresse		F9
2 0 0 7	XX		FA
2 0 0 8	XX		FB
2 0 0 9	XX		FC
2 0 0 A	XX		FD
2 0 0 B	10	DJNZ	FE
2 0 0 C	F9	Distanz	FF
2 0 0 D	XX		00

Wir merken, daß hexadezimales Rückwärtszählen gar nicht so einfach ist, aber man kann es lernen ...

Welchen Vorteil haben nun solche Distanzsprünge?
Ganz einfach - sie sind adressenunabhängig. Wir können unser Programm auch irgendwo anders im Speicher unterbringen, die Sprünge erfolgen immer zur richtigen Stelle.
Ein Problem gibt es noch: Wo erfolgt der Umschlag von "negativer" auf "positive" Sprungrichtung?
Ganz einfach - in der Mitte!

- 7F ist die höchste positive Sprungweite (vorwärts)
- 80 ist die höchste negative Sprungweite (rückwärts).

So, das zu den relativen Sprüngen.

Unsere beiden Befehle DJNZ und JRNZ haben aber noch eine Eigenheit - sie führen nicht immer zu Sprüngen, sondern nur beim Vorliegen bestimmter Bedingungen.
So wird z. B. bei JRNZ nur dann gesprungen, wenn das Ergebnis der vorherigen Rechenoperation (DEC C) nicht 0 war (Not zero - JRNZ). Ist das Ergebnis gleich 0, wird einfach der nächste Befehl auf Adresse 2012 usw. (C3 00 20) abgearbeitet. Der Befehl JRZ (Zero - JRZ) macht das genau umgekehrt, dort wird bei 0 gesprungen.

Genauer informieren wir uns in [7] , Teil 2 auf S.66.

6. Ein elektronischer Würfel

Nachdem wir alle möglichen Texte auf unser Display gebracht - haben, liegt es nahe, dem Rechner selbst die "Verantwortung" für den Anzeigeinhalt zu übertragen. Warum soll ein Computer z. B. nicht würfeln können?

Wir wollen uns vorher überlegen, wie unser Würfel funktionieren soll.

Legen wir also fest:

- Mit der [-]-Taste wollen wir das Display dunkel machen.
- Beim Betätigen der [+]-Taste soll ein kurzer Piepton ertönen, danach soll die Zufallszahl (1...6) von rechts nach links "einrollen" und auf der 1. Stelle des Display "liegenbleiben" bis wieder [-] gedrückt wird.

Folgende Unterprogramme werden gebraucht:

- DAK 1
  - zur ständigen Darstellung der gewürfelten Zahl
  - zur ständigen Abfrage der [-]-Taste (diese hat in DAK 1 die Wertigkeit 11)
- DAK 2
  - zur einmaligen Ansteuerung des Displays (bei der "laufenden" Ziffer) und der Tastatur ([+]-Taste hat hier die Wertigkeit 0A)
- SOUND
  - zur Erzeugung des Pieptones

Die Grundlage des Programmes ist die Erzeugung des Zufallswertes 1 ... 6. Das geht so vor sich:

- Rufe DAK 2 auf.
- Lade C mit 21. 21 ist im 7Segment-Code (sh. Kapitel 5) die "1".
- Frage einmal, ob die Taste OA ([+]) gedrückt ist. Wenn ja, springe auf Adresse 205D. Wenn nein, rufe DAK 2 auf.
- Lade C mit CD, das ist die "2" usw. Nach der "6" wird wieder auf "1" gesprungen.

Auf Adresse 205D wird immer dann gesprungen, wenn erkannt wurde, daß die [+]-Taste gedrückt wurde.

Der Rest des Programmes erzeugt den Ton, läßt die Ziffer "rollen" und sichert die ständige Anzeige des gültigen Wurfes. Die jeweils gewürfelte Ziffer wird in der Speicheradresse 2105 abgelegt, oberhalb und unterhalb muß zur Dunkelsteuerung und zur Sicherung des Laufeffektes 00 eingetragen werden (Bereich 2100 bis 210C).

Und so sieht unser Würfelprogramm aus:

```
2000   DD 21 00 21   LD IX, 2100
2004   CD 5A 04 *    CALL DAK 1          Anzeigen von
                                         2100 ... 2105
2007   FE 11         CP 11               [-]-Taste?
2009   20 F9         JRNZ F9
200B   DD 21 06 21   LD IX, 2106
200F   CD 83 04 *    CALL DAK 2          Anzeigen von
                                         2106 ... 210B
2012   0E 21         LD C, 21            "1" in 7Segment-
                                         Darstellung
2014   FE OA         CP 0A               [+]-Taste?
2016   CA 5D 20      JPZ 205D
2019   C3 1C 20      JMP 2010
2010   CD 83 04 *    CALL DAK 2
201F   OE CD         LD C, CD            "2"
2021   FE 0A         CP 0A               [+]-Taste?
2023   CA 5D 20      JPZ 205D
2026   c3 29 20      JMP 2029
2029   CD 83 04 *    CALL DAK 2
202C   0E AD         LD C, AD            "3"
202E   FE 0A         CP 0A               [+]-Taste?
2030   CA 5D 20      JPZ 205D
2033   C3 36 20      JMP 2036
2036   CD 83 04 *    CALL DAK 2
2039   0E 2B         LD C, 2B            "4"
```

```
203B   FE 0A          CP 0A                [+]-Taste?
203D   CA 5D 20       JPZ 205D
2040   C3 43 20       JMP 2043
2043   CD 83 04 *     CALL DAK 2
2046   OE AE          LD C, AE             "5"
2048   FE 0A          CP 0A                [+]-Taste?
204A   CA 5D 20       JPZ 205D
204D   C3 50 20       JMP 2050
2050   CD 83 04 *     CALL DAK 2
2053   0E EE          LD C, EE             "6"
2055   FE 0A          CP 0A                [+]-Taste?
2057   CA 5D 20       JPZ 205D
205A   C3 0F 20       JMP 200F
205D   79             LD A, C
205E   32 05 21       LD (2105), A
2061   0E 25          LD C, 25             Ton
2063   21 00 01       LD HL, 0100
2066   CD 76 03 *     CALL SOUND
2069   DD 21 06 21    LD IX, 2106
206D   0E 06          LD C, 06
```

```
206F   06 02          LD B, 02             Schnelligkeit
                                            der Laufschrift
2071   CD 83 04 *     CALL DAK 2           Lautschrift und
                                            Anzzeige
2074   10 FB          DJNZ FB
2076   DD 2B          DEC IX
2078   0D             DEC C
2079   20 F4          JRNZ F4
207B   03 00 20       JMP 2000
207E   00



2100   00
  .    .
  .    .
  .    .
2104   00
2105   E7                                  Anzeige der "0"
                                            bei Beginn
2106   00
  .    .
  .    .
  .    .
210B   00
```

7. Ein Digitalvoltmeter

Bisher haben wir unseren Rechner nur als abgeschlossenes Gerät betrieben. Einer seiner wesentlichen Vorteile ist aber, daß er mit seiner Umwelt "kommunizieren", d. h. Informationen austauschen kann. Dazu dienen die beiden Steckverbinder X 1 und X 2 an der rechten Seite des Gerätes. Dabei ist

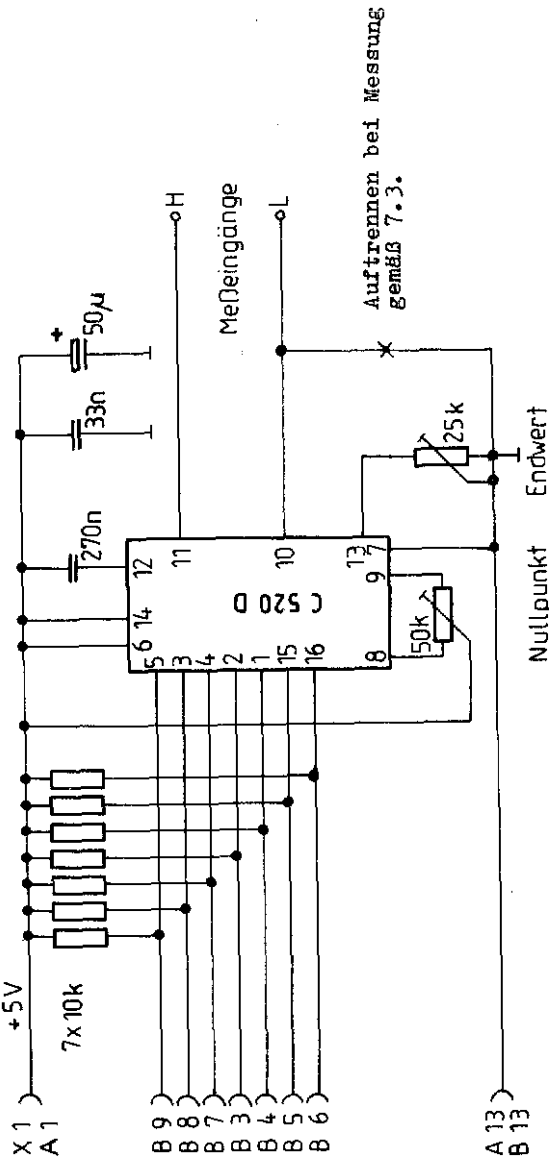
X 1 der sog. "USER-BUS" und
X 2 der komplette "CPU-BUS".

Wir wollen uns zunächst mit dem "USER-BUS" beschäftigen. Er enthält einen kompletten PIO-Port, einen weiteren halben PIO-Port sowie CTC-Signale (sh. auch Kapitel 8), die dem Benutzer (engl.: user) zur Verfügung stehen.
Die genannten PIO-Ports dienen zur Ein- oder Ausgabe von Daten in das Rechnersystem - PIO heißt ja "Parallel Input, Output", also paralleler Eingang, Ausgang. Diese wollen wir zunächst als Eingabemöglichkeit nutzen. Interessant ist dabei die Anwendung als Digitalvoltmeter.
Dabei haben wir zwei Aufgaben vor uns:

1. den Bau eines Adapters (A/D-Wandler),
2. das dazu benötigte Rechnerprogramm.

7.1. Die Zusatzschaltung

Wir bauen uns eine Schaltung nach folgendem Schaltplan auf:

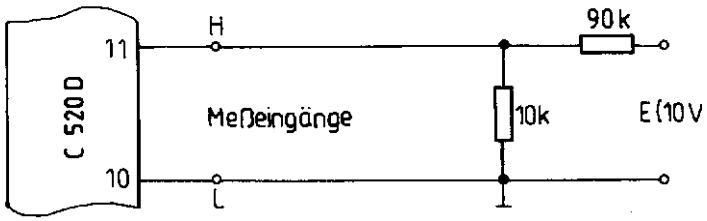

Auftrennen bei Messung gemäß 7.3.
Meßeingänge
H
L
50µ
33n
270n
25k
50k
C 520 D
Endwert
Nullpunkt
+5V
7x10k
X 1
A 1
B 9
B 8
B 7
B 3
B 4
B 5
B 6
A 13
B 13

Die folgende Tabelle zeigt die Zusammenschaltung:

Funktion	Steckverbindung Anschluß	USER-PIO Port	Erläuterungen
+ 5 V	X 1/A 1		LC-80 Brücke A 1/ + 5 V !
LSD NSD MSD	X 1/B 9 X 1/B 8 X 1/B 7	A 6 A 5 A 4	Digits
A B C D	X 1/B 3 X 1/B 4 X 1/B 5 X 1/B 6	A 0 A 1 A 2 A 3	BCD-Wert
Masse	A 1/A 13, B 13		

Achtung! Die Betriebsspannung + 5 V liegt noch nicht am Steckverbinder X 1 an. Wir müssen dazu erst eine Brücke zwischen X 1/A 1 und + 5 V in die Rechnerleiterplatte einlöten. Am besten geht das oberhalb des Steckverbinders X 1, dort sind dafür zwei Löcher vorgesehen.

Wenn wir den Eingang des C 520 D wie folgt beschalten, können wir später sofort Spannungen im 10 V - Bereich messen, also z. B. direkt die Betriebsspannung + 5 V. Das hat den Vorteil, daß wir unser DVM einfach eichen können.



Achtung! Es ist notwendig, daß wir das Modul im stromlosen Zustand stecken und dann erst unser Programm laden. Beim Einstecken unseres Zusatzmoduls kann es sonst passieren, daß nur eine Ziffer leuchtet - und zwar sehr hell. Wir müssen dann sofort RES drücken, um eine Überlastung des Displays zu verhindern!

7.2. Das Programm

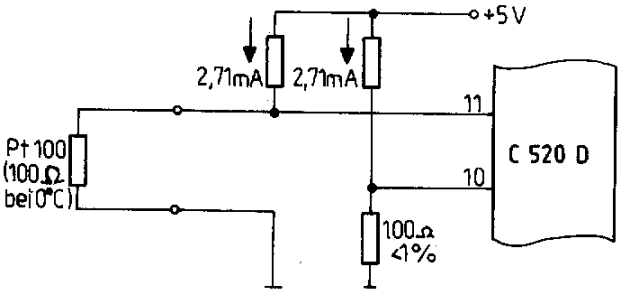
Damit wir unser DVM sofort testen können, schreiben wir uns das auf S. 47 der Bedienungsanleitung des L0-80 stehende Programm "AD-Anschluß" in unseren Arbeitsspeicher. Dabei sind die Adressen zu beachten, bei denen Unterprogramme abgerufen werden (200C, 2013) und je nach ROM-Variante evtl. zu korrigieren. Weiterhin haben sich in der Programmauflistung der Bedienungsanleitung (1. Ausgabe November 1984) zwei wesentliche Fehler eingeschlichen, hier die richtigen Daten:

- auf Adresse 2016 18 F0 JR ADO1
- auf Adresse 2053 20 C3 JRNZ AD10 .

Wenn alles richtig eingegeben wurde und unsere Zusatzschaltung in Ordnung ist, müßte es auf Anhieb funktionieren. Unser Display zeigt vierstellig an. Zur Eichung haben wir den Spannungsteiler 90 kOhm/10 kOhm eingebaut. Pin 10 des C 520 D liegt an Masse, der Eingang ist offen.
Wir drehen am Nullpunktregler so lange, bis 0000 angezeigt wird. Dann wird E(H) mit + 5 V verbunden und der Regler für den Endwert betätigt, bis 0500 erscheint, das sind dann + 5 V. Dies wird dann im Wechsel so lange wiederholt bis alles stimmt. Das DVM-Programm wird jetzt nicht erläutert, versuchen wir es einmal selbst. Wir brauchen dazu die U 880 D - Befehlsliste [4], die PIO-Beschreibung [2] sowie die Bedienungsanleitung des LC-80[5].

7.3. Eine nützliche Erweiterung

Wer einen Platin-Widerstands-Temperaturfühler Pt 100 hat, kann jetzt sehr einfach Temperaturen messen. Dazu dient folgender Vorsatz für unser DVM:



> Wenn wir unser DVM zur Temperaturmessung verwenden wollen, entfernen wir natürlich unseren Spannungsteiler 90 kOhm/ 10 kOhm und trennen Pin 10 des C 520 D von der Masse.

Die beiden Widerstände zur Einstellung des Stromes 2,71 mA müssen in unserem Fall genau 1,75 kOhm haben und vor allem beide gleich groß sein.

7.4. Was unser DVM wert ist

Bis jetzt könnte ja einer mit gutem Grund behaupten, daß wir ein DVM auch billiger haben könnten. Insbesondere der C 520 D ist mit wenigen "Handgriffen" in ein brauchbares DVM oder gar Multimeter umzuwandeln. Der dazu nötige Schaltungsaufwand ist sehr gering. Das stimmt ...

Aber wir können viel mehr!
Unser "Computer-DVM" ist ja rechnergesteuert, wir können durch Programmierung seine Anwendung beliebig an irgendein Problem anpassen, z. B.:

- den Zeitpunkt der Messung festlegen,
- Grenzwerte eingeben, bei deren Überschreitung Alarm ausgelöst wird. Das kann im einfachsten Fall ein Ton oder aber ein Lied sein (Oh, wie ist es kalt geworden).
- bei Überschreitung eines oder mehrerer Grenzwerte Schaltkommandos auslösen, z. B. Heizung aus ...,
- Meßergebnisse umrechnen, z. B. Spannungsabfälle in Widerstandswerte,
- nichtlineare "Geber" abfragen und in entsprechende Meßgrößen umrechnen

u. v. a. mehr.
Wir müssen "nur" unseren Rechner beherrschen lernen - und können nahezu "alles" damit machen.
Na, überzeugt?

8. Ein Kapitel "Hardware"

Unser Computer hat anderen Erzeugnissen etwas voraus, was uns jetzt zugute kommt - man kann ohne Probleme sein Innenleben sehen und dort messen. Wir wollen uns jetzt dafür interessieren, was er für Bausteine enthält und wie er elektrisch funktioniert. Dazu benötigen wir einmal den mitgelieferten Schaltplan und zum anderen irgendein Büchlein, in dem die Mikrorechnerschaltkreise des veb mikroelektronik "karl marx" erfurt etwas ausführlicher beschrieben werden [z. B. 1, 2, 3].

Wie fangen wir an?
Am besten wird sein, unseren Computer, der vor uns liegt, zunächst bezüglich seiner Schaltkreise zu untersuchen. Dabei nehmen wir uns zunächst einen der größten vor. Ganz oben links finden wir einen 40poligen "Käfer", der mit U 880 D beschriftet ist.

8.1. U 880 D

Dies ist der wichtigste Schaltkreis in unserem System, der eigentliche Mikroprozessor oder im Fachjargon CPU (Central Processing Unit). Dieser Schaltkreis ist verantwortlich für die Abarbeitung des Programmes, die Steuerung der übrigen Schaltkreise, das Auslesen und das Einschreiben der Daten in die Speicher u. v. a. m. Obwohl der U 880 D nicht der erste und auch nicht der neueste CPU-Schaltkreis ist, hat er weltweit unter der Bezeichnung Z 80 die weiteste Verbreitung gefunden. Viele Geräte wie Rechner, Steuereinheiten sowie ganze Gerätesysteme werden von ihm gesteuert. Wenn wir unseren Schaltplan studieren, erkennen wir, daß die Anschlüsse des U 880 D immer in Gruppen zusammengefaßt sind, so z. B. die Anschlüsse D 0 ... D 7 oder A 0 ... A 15.
Was ist das nun?
Diese "Anschlußbündel" heißen in der Fachsprache "Bus". Unser U 880 D hat einen "Datenbus", einen "Adreßbus" und einen "Steuerbus".

Datenbus

Die 8 Anschlüsse D 0 ... D 7 werden als Datenbus bezeichnet. Wie in einem richtigen (Auto-) Bus werden hier Daten "transportiert". Was Daten sind, haben wir schon im ersten Kapitel gelesen, es sind Informationen für unsere oder von unserer CPU. Das bedeutet, daß diese Informationen dort sowohl herauskommen (Ausgänge!) als auch hineinkommen (Eingänge!).

Wir haben es also mit Schaltkreisanschlüssen zu tun, die als Ausgang oder Eingang benutzt werden können. Zusätzlich können diese Anschlüsse hochohmig geschaltet werden, so als wären sie überhaupt nicht da. Ein solches System bezeichnet man als Tri-State-Anschluß (3 Zustände). Er kann als Ausgang entweder

- L-Pegel (Low oder niedriger Pegel) Logisch 0 oder
- H-Pegel (High oder hoher Pegel) Logisch 1 oder
- hochohmig, d. h. für seine Umgebung überhaupt nicht wirksam sein.

Unser Datenbus mit seinen 8 Anschlüssen D 0 ... D 7 ist in der Lage, genau 8 Informationen gleichzeitig zu transportieren. Das sind unsere 8 Bit aus dem ersten Kapitel. Wir erkennen jetzt, wie unser Mikroprozessor unsere eingegebenen Informationen erkennt und wie er eigene Daten ausgeben kann. Wenn wir ihm z. B. den Befehl CD geben wollen, sieht das an seinem Datenbus so aus:

```
CD                  - Befehl in Heaa-Code
1100 1101           - Befehl in Bitschreibweise
HHLL HHLH           - Befehl in Pegelschreibweise
D7... D0
```

Wir wollen uns merken, daß in unserem System die sog. positive Logik gilt:

Die logische 1 entspricht H-Pegel (High),
die logische 0 entspricht L-Pegel (Low).

Dabei kann der High-Pegel Spannungen von ca. 2,0 ... 5 V haben und der Low-Pegel solche von 0 ... 0,8 V.

Adreßbus

Die 16 Anschlüsse A 0 ... A 15 heißen Adreßbus. Im Gegensatz zum Datenbus handelt es sich um reine Ausgänge, allerdings auch wieder mit Tri-State-Eigenschaften. Dieser Adreßbus dient zur Ausgabe von Adressen. Auch hier benötigen wir wieder unsere Kenntnisse aus dem ersten Kapitel.

Wieviele Adressen können wir mit 16 Adreßleitungen "aufrufen"?
Die erste Adresse lautet:

```
0000 0000      0000 0000
   00             00          = 0000 ,
```

die zweite:

```
0000 0000      0000 0001
   00             01          = 0001
```

und die letzte:

```
1111 1111      1111 1111
   FF             FF          = FFFF .
```

Insgesamt sind 2^{16} Adressen aufrufbar, das ist die "riesige" Anzahl von 65536.
Unser Rechner ist so organisiert, daß die Anwenderprogramme bei der Adresse 2000 beginnen.
Wie würde unser Adreßbus pegelmäßig aussehen, wenn wir eine direkte Adressierung unserer Speicher voraussetzen (in Wirklichkeit wird die Adressierung noch über Spezialschaltkreise "verschlüsselt", was wir aber zunächst ignorieren)?

```
Adresse 2000                - im Hexa-Code
0010 0000 0000 0000         - in Bitschreibweise
LLHL LLLL LLLL LLLL         - in Pegelschreibweise
A 15 ...........A 0
```

Wenn diese Pegelverteilung vorliegt, wird also die Adresse 2000 aktiviert. Wo diese liegt, ist zunächst nicht bekannt. Ebenso ist noch unklar, was dort geschehen soll. Es ist ja möglich, dort eine Information herauszuholen oder eine dort abzuspeichern. Diese Kommandos werden durch den Steuerbus realisiert.

Steuerbus

Der Steuerbus beinhaltet eine Reihe von Kommandoleitungen, die hier nur kurz erwähnt werden sollen, nähere Informationen finden wir z. B. in [1].
An einem Beispiel wollen wir uns mit diesen Kommandoleitungen und ihrer Funktion vertraut machen. Wir betrachten dazu die beiden Anschlüsse /RD und /WR.

/RD ist ein Ausgang, der das Lesen ("Read") von Informationen z. B. aus Speichern ermöglicht. Der Querstrich über RD bedeutet, daß dieser Ausgang Low-aktiv ist, d. h. wenn "gelesen" werden soll, ist /RD = Low.

/WR ist ein Ausgang (Low-aktiv), der das Einschreiben von Daten ("Write") z. B. in externe Speicher ermöglicht. Wenn geschrieben werden soll, ist also /WR Low.

In ähnlicher Weise geben die anderen 5 Kommandoleitungen Anweisungen ab oder empfangen welche. Beispielsweise dient der Anschluß /MREQ (" Memory Request" - Speicheranforderung) dazu, Speicher überhaupt aktiv zu machen. Wenn also Speicher "angeschlossen" werden sollen, muß /MREQ = Low sein. Außerdem wird durch /RD oder /WR = Low entschieden, ob gelesen oder geschrieben werden soll.
Diese Einführung soll zunächst genügen. Wir haben ja noch weitere Schaltkreise in unserem Rechner, so z. B. 2 Stück U 505 D.

8.2. U 505 D

Der U 505 D ist ein ROM ("Read Only Memory" oder "nur lesbarer Speicher"). Wir können hier nur Informationen herausholen, nicht aber hineinbringen. Dies kann also nicht unser Arbeitsspeicher sein.
Hier sind alle Programmteile des sog. Betriebssystems enthalten, so z. B. die schon von uns benutzten Unterprogramme. Der U 505 D wird vom Bauelementehersteller bereits bei der Herstellung "programmiert" und ist damit nur für den Einsatz in unserem Computer verwendbar. Verändern lassen sich die in ihm enthaltenen Informationen nicht mehr. Er ist wie ein Buch, das wir lesen, aber nichts vom Inhalt verändern können. Anders sieht das bei einem EPROM aus.

8.3. U 2716 C

Manche Exemplare des LC-80 haben statt 2 U 505 D einen Baustein U 2716 C (oder äquivalente Importtypen - z. B. K 573 RF 2 aus der UdSSR). Diese Bauelemente fallen besonders durch ihr "Glasfenster" auf. Im LC-80 wurde dieses Fenster durch schwarzen Lack wieder verschlossen, um ein unbeabsichtigtes Löschen zu vermeiden. Der U 2716 C ist ein EPROM, d. h. wie "elektrisch programmierbares ROM". Zunächst handelt es sich um ein ROM, ist also nur zum Lesen da. Der Unterschied zum normalen ROM besteht darin, daß der Inhalt nicht beim Herstellungsprozeß eingebracht wird, sondern nach Fertigstellung des Bauelementes. Dies kann z. B. auch der Anwender tun, wenn er geeignete Gerät besitzt. Ein großer Vorteil dabei ist, daß sich diese Bausteine auch wieder "löschen" lassen - mit ultraviolettem Licht. Wenn wir unser EPROM also mit einer Höhensonne bestrahlen würden (wir tun das natürlich nicht!), wäre unser Betriebssystem weg. Ähnlich reagiert unser Bauelement bei längerer Sonnenbestrahlung. Nach einem solchen Löschvorgang kann unser EPROM wieder programmiert werden - natürlich nur mit einem speziellen (und recht teuren) Gerät.

Wenn unser LC-80 ein EPROM U 2716 0 enthält, ist darin das gesamte Betriebssystem enthalten. Das EPROM hat (wie die 16 in seiner Bezeichnung sagt) einen Speicherumfang von 16 kBit, das sind genau 16384 Bit oder 2048 Byte (2 kByte). Da der U 505 D nur den halben Speicherumfang aufweist (1 kByte), sind zwei Bausteine nötig, um das Betriebssystem unterzubringen. Das erklärt auch die unterschiedlichen Bestückungsvarianten unseres LC-80.

Wenn wir unseren LC-80 genau betrachten, finden wir unterhalb der beiden U 505 D (oder des einen U 2716 C) weitere freie Plätze mit je 24 Löchern. Hier können wir, wenn wir unseren Computer völlig beherrschen, weitere ROMs (oder EPROMs) mit eigenen Programmen einsetzen. Dies könnten z. B. komplette Spielprogramme (Master Mind), komplizierte Unterprogramme

oder sog. Zeichengeneratoren sein, wie sie für Textdarstellung auf Fernsehbildschirmen benötigt werden.
Aber das ist "Zukunftsmusik" ...
Wenden wir uns lieber den anderen, schon vorhandenen Bauelementen zu, z. B. den schon früher erwähnten RAMs.

8.4. U 214 D (U 224 D)

dir finden die zwei kleineren 18poligen Schaltkreise links neben unserer Tastatur. Je nach Bestückungsvariante können das auch Importschaltkreise 6514 o. a. sein.
Auch diese beiden Bauelemente sind Speicher. Zusammen haben sie einen Speicherumfang von 1 kByte. Das besondere an diesen Speichern ist, daß sie im normalen Betrieb sowohl gelesen als auch beschrieben werden können. Diese sog. RAMs (Random Acces Memory oder Speicher mit wahlfreiem Zugriff) besitzen Speicherzellen, in die beliebig Informationen eingeschrieben oder aus ihnen ausgelesen werden können, ohne daß diese Informationen beim Lesen verlorengehen. Erst wenn die Betriebsspannung abgeschaltet wird, gehen die Daten verloren. Dies ist aber bei ROMs oder EPROMs nicht der Fall - wir sehen jetzt ein, warum es so unterschiedliche Arten von Speichern geben muß:

- ROMs dienen zur Speicherung von Daten, die nicht verlorengehen dürfen (auch bei fehlender Betriebsspannung nicht!). Dafür sind ihre Informationen leider auch nicht (ohne Hilfsmittel) veränderlich.
- RAMs sind Speicher zur Aufbewahrung von Arbeitsdaten oder zur Aufstellung eigener Programme. Sie verlieren ihre Informationen, wenn die Betriebsspannung abgeschaltet wird. Glücklicherweise können wir mit dem LC-80 auch diese Informationen "retten", wie wir im Kapitel 2 gelernt haben.

Auch hier stellen wir fest, daß über unseren beiden RAMs weitere RAM-Plätze vorhanden sind. Diese können im Bedarfsfall mit gleichen RAMs bestückt werden und ermöglichen dann die Speicherung von sehr umfangreichen Programmen (bis max. 4 kByte).
Was finden wir weiter auf unserem LC-80?

8.5. U 855 D

Unser LC-80 ist mit zwei Schaltkreisen U 855 D bestückt. Diese 40poligen Bauelemente werden als PIO-Schaltkreise (Parallel-Input-Output oder Parallele Ein- und Ausgabe) bezeichnet. Sie dienen zur Verbindung der CPU mit der "Umwelt", die auch als "Peripherie" bezeichnet wird.
Im allgemeinen sind unter Peripherie zu verstehen:

- Baugruppen zur Eingabe von Daten (z. B. unsere Tastatur oder unser selbstgebautes Digitalvoltmeter),
- Baugruppen zur Datenausgabe (z. B. unser Display oder unser "Lautsprecher").

Ein PIO-Schaltkreis U 855 D besitzt zwei "Ports", die aus je 8 Anschlüssen bestehen (Port A, Port B). Diese Ports sind sehr universell nutzbar. Durch Programmierung über die CPU. können vier verschiedene Betriebsarten ausgewählt werden:

- Byte-Eingabe
- Byte-Ausgabe
- Byte-Ein-/Ausgabe (Zweirichtungsbetrieb)
- Bit-Ein-/Ausgabe (sog. Control-Mode).

Die Funktion des PIO-Schaltkreises ist sehr komplex und kann hier daher nicht vollständig erläutert werden. Im Kapitel 11 werden wir uns mit den Programmiermöglichkeiten der PIO auseinandersetzen. Trotzdem müssen wir auf spezielle Literatur

zurückgreifen, wenn wir mehr wissen wollen [2].
Wozu dienen nun unsere beiden PIOs?

- Die sog. System-PIO "bedient" das Display (Port A die Segmente, Bit 2 ... 7 von Port B die Digits). Bit B 0 und Bit B 1 versorgen den Kassettenanschluß. Die Bits B 2 ... B 7 dienen gleichzeitig zur Ansteuerung der Tastatur.
- Die sog. User-PIO fragt über Bit B 4 ... B 7 die Tastatur ab, die restlichen 4 Bit von Port B und der gesamte Port A werden über den Steckverbinder A 1 nach außen geführt und stehen dem Benutzer (User-Bus) zur Verfügung. Wir haben davon mit unserem DVM schon Gebrauch gemacht.

8.6. U 857 D

Diese sog. Zähler/Zeitgeber-Bausteine (Counter Timer Circuit - CTC) dienen als programmierbare Zeitnormale, als Ablaufsteuerungen und allgemein als Zähler innerhalb von Mikrorechnern. Im Augenblick machen wir uns keine Gedanken zu diesem Baustein. Wir werden ihn brauchen, wenn wir z. B. eine Digitaluhr programmieren wollen. Wenn wir erst Könner auf dem Gebiet Mikroprozessortechnik sind, werden wir den Nutzen des U 857 D erkennen und uns mit Spezialliteratur zu diesem Baustein beschäftigen [3].

8.7. DS 8205 D

Unser LC-80 ist mit zwei Schaltkreisen dieses Typs bestückt.

Der DS 8205 ist ein sog. 1 aus 8 - Dekoder. Er wandelt ein 3Bit-Wort (A, B, C) in 8 einzelne Zustände um. Drei Bit ergeben, wie wir schon wissen, $2^3 = 8$ verschiedene Möglichkeiten. Bei jeder Kombination der drei Eingangssignale wird einer der 8 Ausgänge aktiv.
In unserem Computer werden diese Schaltkreise benötigt, um den jeweils gewünschten Speicher bzw. die gewünschte PIO auszuwählen. Dazu einige Erläuterungen:
Wir haben weiter vorn das CPU - Bussystem erklärt. Jetzt wollen wir dessen entscheidenden Vorteil erkennen. Aus dem Schaltplan entnehmen wir, daß alle Schaltkreise (U 880 D, U 505 D, U 214 D, U 855 D) an den Datenbus parallel angeschlossen sind. Ähnlich ist das mit dem Adreßbus. Wenn das so ist, muß doch ein heilloses Durcheinander auf diesen Bussystemen herrschen ... Jeder Schaltkreis gibt Daten oder Adressen ab, wer ist denn eigentlich gemeint, wo sollen die Daten hin? Das Rätsel klärt sich schnell auf.
Wir haben in den vorangegangenen Abschnitten über Tri-State-Anschlüsse gesprochen. Alle verwendeten Schaltkreise können ausgangsseitig hochohmig geschaltet werden, sie beteiligen sich einfach nicht am "Gespräch". Die CPU ist "Gesprächsleiter" und erteilt den anderen Schaltkreisen das "Wort" oder verbietet es ihnen. Jeder dieser Schaltkreise hat einen speziellen Eingang /CS (Chip select, Schaltkreis ist ausgewählt). Je nach Zustand der Steueranschlüsse der CPU oder deren Adreßbus wird, eben mit Hilfe des DS 8205 D, ein ROM, ein RAM oder eine PIO "angesprochen" oder "abgehört".
Eindrucksvoll, nicht wahr?

8.8. DL 000 D und DL 014 D

Die beiden Schaltkreise dienen einmal zur Erzeugung des CPU-Taktes von ca. 1 MHz und zum anderen bestimmten logischen

Verknüpfungen und zur Realisierung von sauberen Schaltflanken an den Tasten RES und NMI.

8.9. B 861 D

Dieser Operationsverstärker ist verantwortlich für die Realisierung der Empfangs- und Sendefunktion im Kassettenbetrieb.

8.10. VQB 23

Unser LC-80 enthält drei Anzeigebauelemente VQE 23, von denen jede zwei Ziffern mit je 7 Segmente (+ Dezimalpunkt) darstellen kann. Diese Anzeige wird im sog. Multiplexverfahren betrieben. Das heißt, die Stellen (Digits) werden zeitlich nacheinander angesteuert. Gleichzeitig zur Digitauswahl wird die 7Bit-Information für diese Stelle an die 7 Segmente gegeben. Mit dem Digitwechsel verändert sich auch die 7Segment-Information. Das geht so schnell vor sich, daß das Auge diesen Wechsel nicht mehr wahrnehmen kann. Wenn wir das geschilderte Multiplexverfahren unbedingt optisch sehen wollen, können wir das durch Verringern der Taktfrequenz tun, aber Achtung !

> Die Taktfrequenz unseres L0-80 wird mit dem Regelwiderstand R 321 eingestellt, wird er verändert, laufen alle Vorgänge mit veränderter Geschwindigkeit ab, z. B. auch die Kassettenüberspielfunktion. Das bedeutet, unser Computer kann seine früher gespeicherten Programme nicht mehr lesen !!!
>
> Wir sollten diese Experimente also nur veranstalten, wenn wir meßtechnisch so ausgerüstet sind, daß wir die ursprüngliche Taktfrequenz wieder einstellen können.

Ansonsten glauben wir einfach, daß unsere Anzeige nicht statisch, sondern im schnellen Wechsel angesteuert wird.

8.11. B 3170 H (oder 7805)

Dieser unscheinbare Schaltkreis fällt nur wegen seines Kühlkörpers ins Auge. Tatsächlich wird hier Wärme erzeugt - wir merken das beim Berühren.
Dieser Schaltkreis ist ein moderner Spannungsregler, der die von unseren Schaltkreisen benötigte Spannung von genau 5 V bereitstellt.
Vorteilhaft sind seine Eigenschaften:

- sehr genaue Regelung bei Lastschwankungen bzw. Eingangsspannungsschwankungen,
- thermische Schutzeinrichtung,
- selbständige Strombegrenzung,
- geringer externer Bauelementebedarf.

8.12. Zusammenfassung

Wir haben die wesentlichen Bausteine unseres LC-80 kennengelernt. Über einzelne Widerstände, Dioden, Kondensatoren und Transistoren haben wir nicht gesprochen. Natürlich sind auch diese wichtig und gehören wie die Leiterplatte zur Hardware. Endlich taucht der Begriff aus der Überschrift dieses Kapitels wieder auf. Was ist das?

> Unter "Hardware" versteht man alle Bauteile, Leitungen, die Art deren Anordnung und komplette Geräte, die zu einem Rechnersystem gehören.

Das Gegenstück dazu ist die sog. "Software", die alle Programme und Unterprogramme (ob nun fest programmierte Betriebssysteme oder frei verfügbare Anwenderprogramme), Dateien usw. umfaßt.

9. Ein IC-Tester

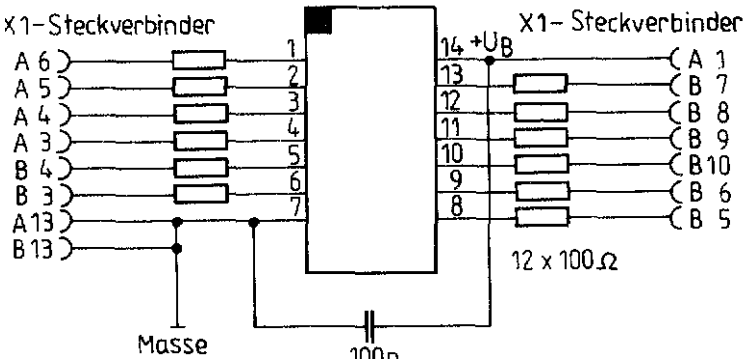
Wohl jeder Bastler steht hin und wieder vor der Frage, ob ein digitaler integrierter Schaltkreis aus seiner "Vorratskiste" noch funktioniert oder nicht. Bisher waren zur Klärung dieses Problems mehr oder weniger umfangreiche Vorbereitungen notwendig, wie z. B. der Aufbau einer Prüffassung, erheblicher Verdrahtungsaufwand, Meßgerät oder Logikprüfstift usw. Der LC-80 vereinfacht dies alles wesentlich, weil mit Ausnahme des einmaligen Aufbaues der Prüfleiterplatte alle Arbeiten "ohne Lötkolben", also programmtechnisch erledigt werden können.
Folgende Randbedingungen gibt es für unseren Tester:

1. Masseanschluß muß auf 7, $+U_B$ - Anschluß auf 14 des IC-Prüflings liegen.
2. Es lassen sich nur 14polige Schaltkreise testen (einschließlich $+U_B$ und Masse).
3. Die Tests sind auf reine Funktionskontrolle beschränkt. Grenzfrequenzen, Pegel oder andere Paramter lassen sich also nicht messen.
4. Es können praktisch alle digitalen Schaltkreise folgender Familien getestet werden:
   * TTL 1
   * Low-Power-Schottky-TTL
   * Schottky-TTL
   * CMOS

   } nicht bei open-collector-stufen!

9.1. Die Prüfleiterplatte

Die Prüfleiterplatte kann prizipiell in beliebiger "Technologie" aufgebaut werden. Wichtig ist nur, daß der Anschluß über einen 26poligen Steckverbinder zum User-Bus (X 1) hergestellt werden kann und daß die Anschlüsse der IC-Fassung (14polig) nach folgendem Schema an diesen Steckverbinder geführt werden:

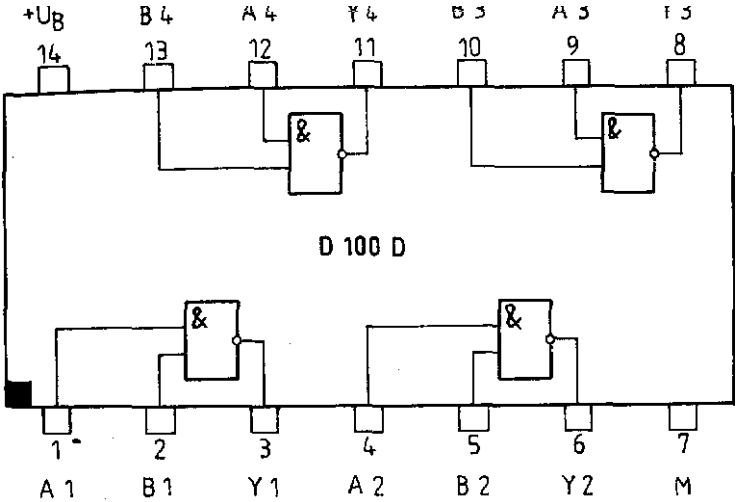


Nur bei der angegebenen Beschaltung (die im Interesse einfacher Gestaltung der Leiterplatte so gewählt wurde) funktionieren die im folgenden beschriebenen Prüfprogramme richtig.

9.2. Der typabhängige Programmteil

Für jeden Digitalschaltkreis existiert zur Beschreibung der Funktionsweise eine sog. Wahrheitstabelle. Diese kann entweder vorhandenen Datenblättern entnommen werden oder muß auf Grund der jeweiligen logischen Verknüpfung selbst erstellt werden. Zum besseren Verständnis der Arbeitsgänge sollen die

Abläufe für den Schaltkreis D 100 D beispielhaft erläutert werden. Der D 100 D wurde gewählt, weil er in nahezu allen Bastelkisten vorrätig sein dürfte. Eine Reihe weiterer Typen werden im Abschnitt 9.5. behandelt.
Die folgende Abbildung zeigt zunächst das Anschlußschema dieses IC:



A, B – Eingänge
Y - Ausgänge

Der D 100 D enthält 4 NAND-Gatter. Diese logische Verknüpfung liefert am Ausgang nur dann eine logische "0" (Low-Pegel), wenn beiden Gattereingängen zugleich eine logische "1" (High-Pegel) angeboten wird.
Die Wahrheitstabelle für ein solches Gatter (z. B. Gatter 1) sieht wie folgt aus:

	Eingang 1	Eingang 2	Ausgang
	A 1	B 1	Y 1
1. Test	0	0	1
2. Test	0	1	1
3. Test	1	0	1
4. Test	1	1	0

Diese Wahrheitstabelle ist die Grundlage des Testablaufes. Sie legt fest, welche Informationen vom Rechner auf die Prüflingseingänge ausgegeben werden müssen bzw. welche dazugehörigen Ausgangsinformationen des Prüflings vom Rechner erwartet werden. Zunächst muß deshalb festgelegt werden, welche Anschlüsse des User-Bus mit Prüflingseingängen bzw. -ausgängen verbunden sind. Für unser Beispiel (D 100 D und Verwendung der o. a. Leiterplatte) ergibt sich folgende Zusammenschaltung in Tabellenform:

User-PIO-Anschluß	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	B 3	B 2	B 1	B 0
Steckverbinder X 1/Anschluß	B10	B 9	B 8	B 7	B 6	B 5	B 4	B 3	A 3	A 4	A 5	A 6
Prüflings-Pin	10	11	12	13	9	8	5	6	4	3	2	1
Funktion bei D 100 D	B 3	Y 4	A 4	B 4	A 3	Y 3	B 2	Y 2	A 2	Y 1	B 1	Al
E/A-Definition	0	1	0	0	0	1	0	1	0	1	0	0

Die letzte Zeile "E/A-Definition" besagt nichts anderes, als daß in diesem Schema alle Prüflingsausgänge (PIO-Eingänge) mit "1" und alle Prüflingseingänge (PIO-Ausgänge) mit "0" belegt werden. Diese Zeile wird ein eine hexadezimale Form gebracht.

```
0100 | 0101    1111 | 0100
   4 | 5          F | 4
  (Port A)       (Port B)
```

Beim Port B ist zu beachten, daß die höherwertigen Bits (B 4 ... B 7) für die Tastaturabfrage erforderlich sind und ebenfalls alle als Eingänge benutzt werden (deshalb dort immer F eintragen!). Dies sind bereits unsere ersten, für jeden Typ spezifischen Datenwerte, die wir ab Adresse 2106 eintragen müssen; also:

```
2106   4 5
2107   F 4
```

Als nächstes entwickeln wir die komplette Wahrheitstabelle (für alle Gatter) und benutzen dazu wieder die gleiche Tabellenform:

Funktion

Funktion b. D 100 D	B 3	Y 4	A 4	B 4	A 3	Y 3	B 2	Y 2	A 2	Y 1	B 1	A 1
1. Test (00)	0	1	0	0	0	1	0	1	0	1	0	0
2. Test (01)	1	1	0	1	0	1	1	1	0	1	1	0
3. Test (10)	0	1	1	0	1	1	0	1	1	1	0	1
4. Test (11)	1	0	1	1	1	0	1	0	1	0	1	1

Analog werden auch diese Informationen in hexadezimale Form gebracht:

(Port A)	(Port B)
4 5	F 4
D 7	F 6
6 D	F D
B A	F B

Diese Daten werden ab Adresse 210A untergebracht. Auf Adresse 2108 tragen wir noch die Anzahl der Tests (hexadezimal) und auf Adresse 2109 00 ein. Die Adressen 2100 bis 2105 werden zur Typbezeichnung in 7Segment-Darstellung reserviert.

Zusammengefaßt:

2100	00		
2101	E9	"d"	
2102	E7	"0"	Typbezeichnung
2103	E7	"0"	"d 100 d"
2104	21	"1"	
2105	E9	"d"	
2106	45		E/A-Festelegung
2107	F4		
2108	04		Anzahl der Tests
2109	00		
210A	45		1. Test
210B	F4		
210C	D7		2. Test
210D	F6		
210E	6D		3. Test
210F	FE		
2110	BA		4. Test
2111	FB		

9.3. Das allgemeine Programm

Neben dem unter 9.2. beschriebenen typabhängigen Programmteil muß ein typunabhängiges Grundprogramm erstellt werden. Dieses Grundprogramm soll im folgenden kurz beschrieben werden:

- IX wird mit 2100 geladen und DAK 1 aufgerufen. Der Typ wird angezeigt.
- Mit Taste ST wird die Prüfung begonnen.
- IY wird mit 2106 (Adresse E/A-Definition mit Inhalt) geladen.
- P10-Mode 3 wird eingestellt (für Port A und B).
- E/A-Definition wird durchgeführt.
- Durch Erhöhen von IY (jetzt 2108) wird die Anzahl der Tests abgefragt und in Register B abgelegt.
- IY wird zweimal erhöht (Beginn 1. Test). Beide PIO-Ports geben an den festegelegten Ausgängen die ersten Informationen gemäß Wahrheitstabelle an den Prüfling.
- Gleichzeitig werden damit die Register D und E geladen.
- Die beiden IN-Befehle (203A und 203D) übernehmen die an den PIO-Eingängen liegenden "Antworten" des Prüflings in den Akkumulator und gleich in die Register H und L.
- IY wird erhöht.
- Der Befehl SBC HL, DE bewirkt eine Subtraktion HL - DE. In DE waren die Sollwerte des D 100 D gemäß Wahrheitstabelle abgespeichert, HL enthält die über die IN-Befehle ermittelten "Istwerte" des Prüflings. Ist alles richtig abgelaufen, muß also das Ergebnis der Subtraktion 0 sein.
- JRNZ prüft das. Ist das Ergebnis = 0, wird auf Adresse 2046 weitergemacht, ist es = 0 erfolgt Sprung auf Adresse 2053.
- DJNZ dekrementiert Register B (das ja mit der Anzahl der Tests geladen war). Wenn der eben durchgeführte Test nicht der letzte war, wird die Schleife ab 202C erneut durchlaufen. War es der letzte Test, wird das Programm bei 2048 fortgesetzt ("PASS-Anzeige").
- 2048 und 2053 ("FAIL"-Anzeige) bewirken eine kurze Testanzeige und danach einen Rücksprung zum erneuten kompletten Prüfdurchlauf.
- Die Befehle B7 sind notwendig zur Korrektur des Carry-Bits.

```
2000   DD 21 00 21   LD IX, 2100      }  Anzeige des Typs
2004   CD 5A 04 *    CALL DAK 1       }
2007   FE 1E         CP 1E               Taste [ST] ?
2009   20 F5         JRNZ F5
200B   FD 21 06 21   LD IY, 2106
200E   3E PF         LD A, FF         }  PIO-Mode 3 (Port A)
2011   D3 FA         OUT FA           }
2013   FD 7E 00      LD A, (IY+0)     }  E/A-Definition
2016   D3 FA         OUT FA           }  Port A
2018   FD 23         INC IY
201A   3E FF         LD A,FF          }  PIO-Mode 3 (Port B)
201C   D3 FB         OUT FB           }
201E   FD 7E 00      LD A, (IY+0)     }  E/A-Definition
2021   D3 FB         OUT FB           }  Port B
2023   FD 23         INC IY
2025   FD 46 00      LD B, (IY+0)        Anzahl der Tests
2028   FD 23         INC IY
202A   FD 23         INC IY
202C   FD 7E 00      LD A, (IY+0)
202F   57            LD D, A
2030   D3 F8         OUT F8              Ausgabe gem. Wahr-
2032   FD 23         INC IY              heitstabelle Port A
2034   FD 7E 00      LD A, (IY+0)
2037   5F            LD E, A
2038   D3 F9         OUT F9              Ausgabe gem. Wahr
                                         heitstabelle Port B
203A   DB P8         IN F8            }  Einlesen und Ablegen
203C   67            LD H, A          }  in H, Port A
203D   DB F9         IN F9            }  Einlesen und Ablegen
203F   6F            LD L, A          }  in L, Port B
2040   FD 23         INC IY
2042   ED 52         SBC HL, DE          Subtraktion HL - DE

2044   20 0D         JRNZ 0D
2046   10 E4         DJNZ E4
2048   DD 21 80 20   LD IX, 2080      }  Anzeige "PASS"
204C   CD 83 04 *    CALL DAK 2       }
```

```
204F   B7            OR A
2050   C3 0B 20      JMP 200B
2053   DD 21 85 20   LD IX, 2085     }   Anzeige "FAIL"
2057   CD 83 04 *    CALL DAK 2      }
205A   B7            OR A
205E   C3 0B 20      JMP 200B            Testwiederholung
                                          (Dauerlauf!)
```

Der anzuzeigende Text "PASS" bzw. "FAIL" wird ab Adresse 2080 eingetragen:

```
2080   00
2081   AE            "S"
2082   AE            "S"
2083   6F            "A"
2084   4F            "P"
2085   00
2086   C2            "L"
2087   21            "I"
2088   6F            "A"
2089   4E            "F"
208A   00
```

9.4. Bedienung und weitere wichtige Hinweise

Ist unsere Prüfleiterplatte fertig und, was sehr wichtig ist, auch auf Verdrahtungsfehler und andere Pannen kontrolliert, können wir unseren ersten Schaltkreis D 100 D auf die Prüffassung stecken (unbedingt richtige Lage beachten!). Dieser erste Prüfling sollte bereits vorher auf "herkömmliche" Weise ausprobiert werden.

Wenn alles soweit klar ist und auch unser Programm richtig im Speicher ist, können wir es starten. Es müßte jetzt der Text

"d 100 d"

erscheinen. Dies ist die letzte Kontrollmöglichkeit.
Die eigentliche Prüfung beginnt, wenn die ST -Taste gedrückt wird.
War die Prüfung erfolgreich, erscheint

"PASS" ,

wenn nicht

"FAIL" .

In diesem Fall müssen wir alles (Leiterplatte, Programm und Prüfling) nochmals kontrollieren.
Die Prüfung des IC wird ständig wiederholt, d. h. wenn "PASS" erscheint und wir einfach durch Herausziehen des D 100 D einen Defekt vortäuschen, wird "FAIL" angezeigt. Diese interessante Eigenschaft des IC-Tasters kann vorteilhaft zum Finden verdeckter IC-Fehler wie z. B. innere Wackelkontakte oder Wärmefehler genutzt werden. Soweit ist alles recht gut gelaufen.
Bevor wir aber weiter experimentieren, noch einige sehr wichtige Hinweise:

Achtung!

- Unser IC-Tester ist vorzugsweise für funktionstüchtige Schaltkreise gedacht. Defekte ICs (z. B. mit harten Kurzschlüssen) oder Programmfehler führen zur Überschreitung der Grenzdaten unserer PIO. Die zum Schutz eingebauten Widerstände begrenzen z. B. den Kurzschlußstrom nicht ausreichend. Höhere Widerstände würden aber wieder die Funktion unseres Testers einschränken (TTL-ICs könnten wir nicht mehr sicher prüfen). Erfahrungen mit solchen Überlastungen lassen zwar keine Schädigung der PIO erwarten, die Einwirkungszeit muß aber sehr kurz gehalten Werden. Deshalb:

  Bei Anzeige "FAIL" Programm mit RES stoppen, Fehler suchen!

- ICs auf keinen Fall verdreht aufsetzen!
- Immer beachten, daß nur 14polige ICs mit Masse am Anschluß 7 und $+U_B$ (5 V) am Anschluß 14 geprüft werden können!
- Bei CMOS-Bauelementen sind die Behandlungsvorschriften exakt einzuhalten (elektrostatischer Schutz!). Hier ist außerdem zu beachten, daß CMOS-Bauelemente nur im stromlosen Zustand gesteckt werden dürfen. Also erst LC-80 ausschalten, CMOS-IC aufstecken, LC-80 wieder einschalten und dann Programm laden!
- Keine Bauelemente mit abweichender Betriebsspannung prüfen!

Erst jetzt sollten wir den Abschnitt 9.5 lesen und uns an Programme für weitere Typen wagen.

9.5. Weitere Typen

Zur Sicherheit üben wir das Erstellen des typabhängigen Programmteiles mit einem weiteren Typ, z. B. dem D 110 D:

User-PIO-Anschluß	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	B 3	B 2	B 1	B 0
Prüflings-Pin	10	11	12	13	9	8	5	6	4	3	2	1
Funktion b. D 110 D	B 3	C 3	Y 1	C1	A 3	Y 3	C 2	Y 2	B 2	A 2	B 1	A 1
E/A-Definition	0	0	1	0	0	1	0	1	0	0	0	0
000	0	0	1	0	0	1	0	1	0	0	0	0
001	0	1	1	1	0	1	1	1	0	0	0	0
010	1	0	1	0	0	1	0	1	1	0	1	0
100	0	0	1	0	1	1	0	1	0	1	0	1
011	1	1	1	1	0	1	1	1	1	0	1	.0
101	0	1	1	1	1	1	1	1	0	1	0	1
110	1	0	1	0	1	1	0	1	1	1	1	1
111	1	1	0	1	1	0	1	0	1	1	1	1

Aus dieser Tabelle entwickeln wir wieder den typabhängigen Programmteil für den D 110 D:

```
2100   00
2101   E9            "d"
2102   E7            "0"
2103   21            "1"
2104   21            "1"
2105   E9            "d"
2106   25      }     E/A
2107   F0
2108   08            Testanzahl
```

```
2109   00
210A   25        1. Test
2108   F4
2100   77        2. Test
210D   F0
210E   A5        3. Test
210F   FA
2110   2D        4. Test
2111   F5
2112   F7        5. Test
2113   FA
2114   7F        6. Test
2115   F5
2116   AD        7. Test
2117   FF
2118   DA        8. Test
2119   FF
```

Wenn wir einen D 110 D besitzen, probieren wir auch dieses Programm gleich aus.
Zum Abschluß dieses Kapitels finden wir noch einige Musterprogramme zum Ausprobieren und als Anregung zum Selbermachen. Wer mit elektronischen Bauelementen zu tun hat, wird sich evt. eine ganze Bibliothek von typabhängigen Programmen anlegen und am besten gleich auf Kassette speichern. Dabei ist es natürlich möglich, das Hauptprogramm und ein typabhängiges Programm einzeln zu speichern. Wichtig dabei ist, daß getrennte FILE-Namen verwendet werden und die unterschiedlichen Anfangs- und Endadressen beachtet werden. Dann ist das ganze System sehr brauchbar und man ist schnell meßbereit.

Adresse	D 104 D	D 174 D	V 4001 D	V 4011 D	V 4030 D
2100	00	00	E9	E9	E9
2101	E9	E9	21	21	E7
2102	2B	2B	E7	21	AD
2103	E7	25	E7	E7	E7
2104	21	21	2B	2B	2B
2105	E9	E9	E3	E3	E3
2106	A5	0F	C0	C0	C0
2107	FA	F0	FC	FC	FC
2108	02	OA	04	04	04
2109	00	00	00	00	00
210A	A5	85	C0	C0	00
210B	FA	F8	FC	FC	F0
210C	5A	1A	19	D9	D9
210D	F5	F1	F2	FE	FE
210E		E5	26	E6	E6
210F		FE	F1	FD	FD
2110		1A	3F	3F	3F
2111		F1	F3	F3	F3
2112		9A			
2113		F9			
2114		D5			
2115		FD			
2116		B5			
2117		FB			
2118		FA			
2119		FF			
211A		9A			
211B		F9			
211C		E5			
211D		FE			

10. Geschicklichkeitsspiel

In den letzten Kapiteln haben wir recht intensiv arbeiten müssen, eine kleine Auflockerung kann deshalb nicht schaden. Eine auch international typische Anwendung von Computern im Heimbereich sind Computerspiele aller Art. Trotz der begrenzten Darstellungsmöglichkeiten unseres Lerncomputers kann man recht gut mit ihm "spielen". Unser Würfelspiel von Kapitel 6 ist erst der Anfang ...

10.1. Die Spielregeln

Unser Spiel soll so ablaufen:

- Nach dem Programmstart erscheint nach ca. 1 sec. im rechten Feld des Displays eine Zufallsziffer (0 ... F).
- Innerhalb von wiederum etwa 1 sec, müssen wir die dazugehörige Taste finden und betätigen.
- Wenn das gelungen ist, wird in den beiden linken Feldern der Treffer angezeigt (mit 01) und ein kurzer Ton erzeugt.
- Die nächste Ziffer erscheint.
- Bei richtiger Tastenbetätigung piept es wieder und die Trefferanzeige wird auf 02 erhöht.
- usw.

Scheinbar mühelos zu bewältigen ... Aber der Teufel liegt im Detail. Nach jeder Zufallszahl wird nämlich die Zeit zum Finden und Drücken der Taste kürzer. Zum Schluß haben nur noch Könner eine Chance. Die Treffersumme bis zum automatischen Ende des Spieles ist ein "Dokument" der erreichten Fertigkeiten. Die bei mehreren Spieldurchlaufen jeweils erreichte Höchstpunktzahl wird als "Rekord" gespeichert. Dieser Rekord

wird bei Spielende auf den beiden rechten Feldern des Display, angezeigt. Wird der Rekord überboten, ertönt eine Siegesfanfare, wird er verfehlt, ist der Computer "traurig" und zeigt das auch. Das klingt recht vielversprechend und deshalb wollen wir erst einmal das im folgenden Abschnitt aufgelistete Programm eingeben und mit dem Spielen beginnen.

10.2. Das Programm

In diesem Abschnitt finden wir das unkommentierte Programm. Wer sich intensiver damit beschäftigen will, findet im Abschnitt 10.3. einige Erläuterungen.

Geschicklichkeitsspiel

```
2000   21 00 00       LD HL, 0000
2003   22 FB 20       LD (20FB), HL
2006   22 FD 20       LD (20FD), HL
2009   22 FF 20       LD (20FF), HL
200C   22 01 21       LD (2101), HL
200F   22 03 21       LD (2103), HL
2012   06 FF          LD B, FF
2014   DD 21 FE 20    LD IX, 20FE
2018   CD 83 04 *     CALL DAK 2
201B   10 FB          DJNZ FB
201D   0E FF          LD C, FF
```

```
201F   ED 5F          LD A, R
2021   E6 0F          AND 0F
2023   32 FD 20       LD (20FD), A
2026   CD CA 04 *     CALL ONESEG
2029   32 PE 20       LD (204E), A
202C   41             LD B, C
202D   3E 33          LD A, 33
202F   21 FD 20       LD HL, 20FD
2032   CD 83 04       CALL DAK 2
2035   5F             LD E, A
2036   16 21          LD D, 21
2038   1A             LD A, (DE)
2039   BE             CPM
203A   28 0D          JRZ 0D
203C   10 EF          DJNZ EF
203E   79             LD A, C
203F   D6 02          SUB 02
2041   38 03          JRC 03
2043   4F             LD C, A
2044   18 D9          JR D9
2046   03 69 20       JMP 2069
2049   C5             PUSH BC
204A   0E 25          LD C, 25
204C   21 20 00       LD HL, 0020
204F   CD 76 03 *     CALL SOUND
2052   3A FC 20       LD A, (20FC)
2055   3C             INC A
2056   27             DAA
2057   32 FC 20       LD (20FC), A
205A   21 02 21       LD HL, 2102
205D   CD D9 04 *     CALL TWOSEG
2060   C1             POP BC
2061   CD 83 04 *     CALL DAK 2
2064   10 FB          DJNZ FB
2066   C3 3E 20       JMP 203E
2069   3A FC 20       LD A, (20FC)
```

```
2060    47             LD B, A
206D    3A 0A 21       LD A, (210A)
2070    B8             CPB
2071    38 14          JRC 14
2073    21 FE 20       LD HL, 20FE
2076    CD D9 04 *     CALL TWOSEG
2079    0E 01          LD C, 01
207B    21 02 00       LD HL, 0002
207E    CD 76 03 *     CALL SOUND
2081    0C             INC C
2082    20 F7          JRNZ F7
2084    C3 9E 20       JMP 209E
2087    3A FC 20       LD A, (20FC)
208A    32 0A 21       LD (210A), A
208D    21 FE 20       LD HL, 20FE
2090    CD D9 04 *     CALL TWOSEG
2093    DD E5          PUSH IX
2095    FD 21 18 21    LD IY, 2118
2099    CD EE 04 *     CALL MUSIK
2090    DD E1          POP IX
209E    CD 5A 04. *    CALL DAK 1
20A1    C3 00 20       JMP 2000
```

* ROM-Bestückung beachten!

```
2104    00
2105    01
2106    02
2107    03
2108    04
2109    05
210A    00
```

```
210E   07                  Anzeigebereich:       20FE ... 2103
2100   08                  Anzeige Zufallszahl:  20FE
210D   09                  Anzeige Trefferzahl:  2102, 2103
210E   0E                  Anzeige Rekord:       20FE, 20FF
210F   0B
2110   0C                  Speicher Trefferzahl: 20FC
2111   OD                  Speicher Zufallszahl: 20FD
2112   OA                  Speicher Rekord:      210A
2113   OP
2114   FF                  Bereich Noten:        2118 ... 2120
2115   FF
2116   06
2117   FF
2118   0B 04
211A   0F 04
211C   12 04
211E   17 10
2120   80
2121   FF
```

10.3. Wie funktioniert das?

In diesem Abschnitt wollen wir uns einige Abläufe des Spielprogrammes klarmachen. Dieses Programm ist bewußt einfach aufgebaut, um die Übersicht zu behalten. Das kostet natürlich Speicherplatz, der durch mancherlei Tricks verringert werden könnte. Wer es jetzt schon vereinfachen kann - bitte sehr! Es geht los:

- HL wird mit 0000 geladen und diese Nullen in die Speicherbereiche 202B bis 2104 transportiert.

- B wird mit FF geladen, DAK 2 wird aufgerufen (10 ms Anzeige des durch IX adressierten Bereiches, der jetzt noch "dunkel" ist).
- Durch DJNZ wird B dekrementiert (jetzt also FE). Da das nicht gleich 0 ist, Rücksprung zu DAK 2, 10 ms Anzeige usw.
- Ist B 0, wird C mit FF geladen.
- A wird mit Inhalt des Refresh-Registers R geladen - unsere Zufallszahl.
- Der Befehl AND 0F (logische UND-Verknüpfung des Akkumulatorinhaltes mit der Konstanten 0F) bewirkt, daß die niederwertigen 4 Bit des Akkumulators erhalten bleiben, die höherwertigen 4 aber zu 0 werden. Das ist erforderlich, weil A 8 Bit enthält, für eine Ziffer (die wir ja haben wollen), aber nur 4 Bit erforderlich sind.
- Die jetzt errechnete Ziffer wird auf Adresse 20FD abgelegt, ist aber auch noch in A.
- Das Unterprogramm ONESEG macht daraus die 7Segmentform, die wieder in A zwischengespeichert wird und dann endgültig auf 20FE gespeichert wird (dies ist im Anzeigebereich).
- B wird mit Inhalt von C, also FF geladen, A mit 33. Diese Zahl wurde gewählt, um Verwechslungen mit den Tastencodes zu vermeiden.
- HL wird mit 20FD geladen.
- DAK 2 wird aufgerufen, einmal zur Anzeige der Zufallszahl, zum anderen zur Aktivierung der Tastatur.
- Beim Betätigen einer Taste steht der Tastencode (dies ist ein anderer als auf S.58 der LC-80 - Bedienungsanleitung angegeben - dieser gilt nur für DAK 1) im Register A.
- Übernahme dieses Tastencodes in E.
- Laden von D mit 21. Dieser und der vorige Schritt sind ein Trick. Der Tastencode steht in E und 21 in D. Das Doppelregister DE sieht also so aus:

  DE = 21 und Tastencode.

Dies wird zusammen zu einer Adresse und zwar zu der, die die echte Tastenwertigkeit enthält, z. B. 2109 enthält den Wert 05 - also Taste 5.

- A wird mit dieder "echten" Tastenwertigkeit geladen.
- CMPM (oder CPM) vergleicht den Inhalt von A mit dem der durch HL adressierten Speicherzelle (20FD). Diese enthält unsere Zufallszahl. Stimmen beide überein, wird zu Adresse 2049 gesprungen, wenn nicht, erzwingt DJNZ einen erneuten Anzeige- und Tastenabfragezyklus, bis B = 0 ist.
- A wird mit Inhalt von C geladen.
- Der Inhalt von A wird um 2 vermindert.
- Vorausgesetzt. daß A noch nicht negativ ist, wird der Inhalt wieder in C geladen.
- JR bewirkt einen Rücksprung zur erneuten Erzeugung einer Zufallszahl, diesmal dauert aber der Durch—lauf nicht ganz so lange, weil C nicht mehr FF, sondern FT ist usw.
- Der oben erklärte Sprung zur Adresse 2049 (immer dann, wenn eine Taste richtig betätigt wurde) bewirkt mit PUSH BC ein "Retten" des Inhalte des BC-Registers.
- Die folgenden Befehle dienen zur Tonauslösung, die wir schon kennen.
- Jetzt wird der Inhalt der Speicherzelle 20FC (hier ist die aktuelle Trefferzahl gespeichert) in A geladen und um 1 erhöht - wir haben ja ein "Tor" erzielt.
- Der DAA-Befehl korrigiert die Hexadezimal-Zählweise auf verständliche Dezimalzahlen, danach wird die neue Trefferzahl wieder in 20FC abgelegt.
- Die noch in A stehende Trefferzahl wird mit TWOSEG in 7Segmentdarstellung umgewandelt und auf 2102 abgelegt (Anzeigebereich), nachdem der Inhalt von BC wieder aus dem Kellerspeicher geholt wurde.

- Anzeige des Anzeigebereiches mit DAK 2 für die durch B festgelegte Zeit, danach wieder 203E, Zeitverkürzung, neuer Durchlauf usw.
- Ist der durch Subtraktion verkleinerte Wert von C unter 0 "geraten", wird nach 2069 gesprungen.
- Hier wird die Trefferzahl aus 20FC in A geholt und in B geladen.
- Die Rekordzahl wird aus 210A in A geladen und mit B verglichen.
- Ist die aktuelle Trefferzahl kleiner als der Rekord, wird der Rekord mit TWOSEG in 7Segmentform umgewandelt, auf 20FE und 20FF abgelegt und ein ansinkender Sirenenton abgestrahlt und dann auf 209E gesprungen.
- Ist ein neuer Rekord aufgestellt worden, wird die aktuelle Trefferzahl (20FC) in den "Rekordspeicher" 210A eingetragen und die 7Segmentform in 20FE und 20FF abgelegt.
- IX wird gerettet, "Musik" vorbereitet (die Noten stehen in 2118 ... 2120).
- Nach dem "Tusch" werden mit DAK 1 die neuen Zahlen (Treffer und Rekord) dargestellt.
- Bei Tastendruck (z. B. EX) wird DAK 1 verlassen und auf 2000 gesprungen.
  Ein neues Spiel beginnt.

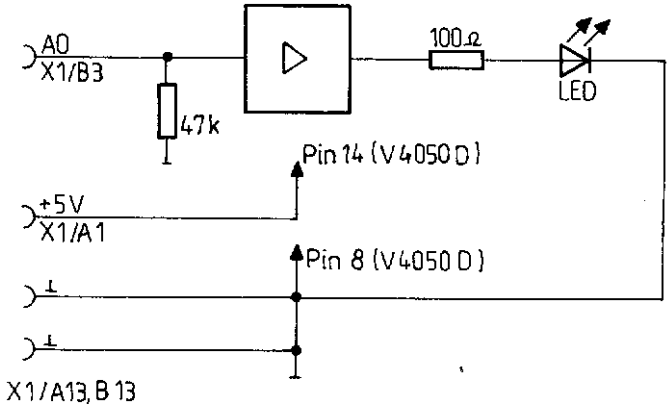
11. Die USER-PIO im Kreuzverhör

Im Kapitel 8 haben wir die grundsätzlichen Aufgaben der beiden PIO-Bausteine unseres LC-80 kennengelernt. Die 12 frei verfügbaren Aus- bzw. Eingabeleitungen der USER-PIO haben wir für die verschiedensten Zwecke schon genutzt. im Fall des DVM z. B. als reine Eingabe, im Fall des IC-Testers als kombinierte Ein- und Ausgabe.
So richtig zufrieden können wir aber eigentlich noch nicht sein, da uns das "Rezept" für die beliebige Anwendung der PIO bisher noch fehlt. Beim Einstieg in dieses Problem soll das folgende Kapitel helfen. Den Rest erreichen wir mit Übung, Probieren und Grundlagenliteratur, wie z. B. der Technischen Beschreibung des U 855 D [2] .

11.1. Ein optisches Hilfsmittel

Zum Verständnis der Funktionsweise unserer PIO benötigen wir "Anschaulichkeit", d. h. wir wollen "sehen", was sie macht. Zu diesem Zweck bauen wir uns ein einfaches Zusatzgerät, mit dem durch Leuchtdioden (LED) der jeweilige Zustand einer Ausgangsleitung optisch angezeigt wird. Wir wollen uns zunächst nur mit Ausgabeproblemen beschäftigen und festlegen, daß logisch "1" (oder High-Pegel) durch Leuchten der Anzeigediode dargestellt wird. Der Zustand logisch "0" (Low-Pegel) wird durch Nichtleuchten charakterisiert.
Wie sieht unser Zusatzgerät aus?
Wir wollen Leuchtdioden antreiben, was durch CMOS-Bausteine bei 5 V Betriebsspannung praktisch noch funktioniert, obwohl es lt. Datenblatt knapp zugeht. Besonders vorteilhaft

ist z. B. der V 4050 D, der 6 nichtinvertierende Treiberstufen enthält. Mit zwei solchen ICs ist unser Problem gelöst, da wir 12 LEDs treiben wollen. Die Schaltung für eine Ausgangsleitung sieht so aus:



Besonders wichtig ist, daß unbedingt Masse und Betriebsspannung an die CMOS-Bausteine geführt werden. Sonst sind Zerstörungen nahezu unvermeidlich. Dabei kontrollieren wir nochmals, daß der Steckverbinderanschluß X1/A1 auf der LC-80 - Leiterplatte mit + 5 V verbunden ist (Brücke oberhalb des X 1 - Steckverbinders).
Die komplette Tabelle aller benötigten Anschlüsse dient uns zur Orientierung:

PIO-Port	PIO-Anschluß	Steckverbinder-Anschluß
A	A0	X 1/B 3
	A1	X 1/B 4
	A2	X 1/B 5
	A3	X 1/B 6
	A4	X 1/B 7
	A5	X 1/B 8
	A6	X 1/B 9
	A7	X 1/B 10
B	B0	X 1/A 6
	B1	X 1/A 5
	B2	X 1/A 4
	B3	X 1/A 3
	Masse	X 1/A 13
		X 1/B 13
	U_B = + 5 V	X 1/A 1

Bei der Konstruktion der Leiterplatte beachten wir die geometrische Anordnung der Leuchtdioden. Die richtige Reihenfolge (A 0, A 1 ... A 7, B 0 ... B 3) sollte der Übersicht wegen unbedingt eingehalten werden!

Wenn alles fertig ist und kontrolliert wurde, stecken wir die neue Baugruppe bei abgeschaltetem LC-80 in den Steckverbinder X 1. Alle Leuchtdioden müssen im Grundzustand (RESET) dunkel sein. Zur Funktionskontrolle dient folgendes Programm:

```
2000    01 00 00        LD BC, 0000
2003    3E 0F           LD A, 0F
2005    D3 FA           OUT FA
2007    D3 FB           OUT FB
2009    3E 55           LD A, 55
200B    D3 F8           OUT F8
200D    D3 F9           OUT F9
200F    10 FE           DJNZ FE
2011    0D              DEC C
2012    20 FB           JRNZ FB
2014    3E AA           LD A, AA
2016    D3 F8           OUT F8
2018    D3 F9           OUT F9
201A    10 FE           DJNZ FE
201C    0D              DEC C
201D    20 FB           JRNZ FB
201F    18 E8           JR E8
```

Wenn alle LEDs blinken, ist alles in Ordnung! ' Ansonsten muß der Fehler gesucht werden, zuerst jedoch RES betätigen, um eventuelle Schäden zu verhindern!

Wem dieser "Disko-Effekt" genügt, der mag hier zunächst aufhören. Aber weitere "Lichtspiele" werden folgen ...

11.2. Praktische Programmierung

Unser Testprogramm realisiert eine wechselnde Ausgabe an den Zusatzbaustein. Wie so etwas funktioniert, wollen wir jetzt kennenlernen.
Zunächst beschränken wir uns auf die Anwendung des Befehls OUT n. Er bewirkt die Ausgabe des Akkumulatorinhaltes an den Ausgabekanal mit der Adresse n . Adresse n bezeichnet die

jeweils gewünschte PIO und deren ausgewählten Port (A oder B). Das besondere dabei ist, daß Steuerkommandos an die PIO und Datenausgaben durch unterschiedliche Adressierung kenntlich gemacht werden. Für unsere UFER-PIO gelten folgende Adressen:

Steuerkommandos an Port A (AC): FA (AC→A-Control)

Steuerkommandos an Port B (BC): FB (BC→B-Control)

Datenausgabe an Port A (AD): F8 (AD→A-Data)

Datenausgabe an Port B (BD): F9 (BD→B-Data).

Die gleiche Tabelle finden wir auch auf unserer Tastatur abgedruckt.

In der Regel geht unsere PIO-Programmierung immer so vor sich, daß zuerst der Akkumulator mit dem Steuerkommando (oder Datenwort) geladen wird und danach ein Ausgabebefehl (OUT n) an die jeweilige Adresse erfolgt. Die Unterscheidung von Steuerkommandos und Daten nimmt die PIO anhand der Adresse selbst vor.
Bevor aber überhaupt Daten nach außen übertragen werden können, muß die PIO vorbereitet ("initialisiert") werden. Das geschieht grundsätzlich mit Steuerkommandos, die z. B. festlegen:

- in welcher Betriebsart die PIO betrieben werden soll,
- welche Signale von außen den inneren Programmablauf unterbrechen sollen (Interrupt-Festlegungen),
- welche Leitungen Eingänge und welche Ausgänge sein sollen usw.

Wir fangen vorerst immer mit der Festlegung der Betriebsart (Mode) an. Diese Festlegung erfolgt durch ein Steuerwort mit folgendem Aufbau:

D7	D6	D5	D4	D3	D2	D1	D0
M	M	X	X	1	1	1	1

Die mit X bezeichneten Stellen haben für uns keine Bedeutung, wir setzen sie einfach gleich 0. Die mit M bezeichneten Stellen kennzeichnen die Betriebsart:

Betriebsart	D 7	D 6	Funktion
Mode 0	0	0	Byte-Ausgabe
Mode 1	0	1	Byte-Eingabe
Mode 2	1	0	Bidirektional
Mode 3	1	1	Einzelbitsteuerung

Das komplette Steuerwort hat demnach im Hexa-Code folgende Form:

Mode 0 - 0F
Mode 1 - 4F
Mode 2 - 8F
Mode 3 - CF

11.2.1. Mode 0

Wir wollen alle Leitungen als Ausgabe gebrauchen und entscheiden uns für Mode 0. Zur Vereinfachung beschränken wir uns zunächst auf Port A, dann erfolgt die Initialisierung so:

```
3E 0F   LD A, 0F      Das Steuerwort OF wird in den
D3 FA   OUT FA        Akku geladen. Der Inhalt von A wird
                      (als Steuerwort) für den Kanal A zur
                      Festlegung der Betriebsart benutzt.
```

Das war's schon, die PIO ist bereit, im Mode 0 Daten auf dem Port A auszugeben. Wir wählen als Datenwort z. B. 01 und geben das aus:

```
3E 01   LD A, 01            Das Datenwort 01 wird in den
D3 F8   OUT F8              Akku geladen. Der Inhalt von
                            A wird (als Datenwort) auf
                            Port A ausgegeben.
```

Zusammenhängend geben wir ein:

```
2000    3E 0F   LD A, 0F  }   Initialisierung für PIO-Mode 0
2002    D3 FA   OUT FA    }   des Kanals A
2004    3E 01   LD A, 01  }   Ausgabe von 01
2006    D3 F8   OUT F8    }   auf Kanal A
2008    76      HALT
```

Nach Programmstart leuchtet sofort die LED für A 0 auf (außerdem natürlich die HALT-LED), was den Wert 01 symbolisiert. Wir können das sofort mit anderen Hexadezimalzahlen auf Adresse 2005 weiterführen und werden bei richtiger Funktion immer die Bitdarstellung für diese Zahlen erhalten. Allein das sollte uns für die Bastelei entschädigen, Haben wir doch jetzt ein einfaches Hilfsmittel, Hexa- in Binärzahlen umzuwandeln oder z. B. Registerinhalte in Bitdarstellung anzuschauen.
Wenn Port B einbezogen werden soll, muß sowohl eine gesonderte Initialisierung als auch eine gesonderte Datenausgabe erfolgen, also:

```
2000    3E 0F   LD A, 0F
2002    D3 FA   OUT FA          Mode 0, Port A
2004    D3 FB   OUT FB          Mode 0, Port B
2006    3E 33   LD A,   33
2008    D3 F8   OUT F8          Ausgabe von 33 auf Port A
200A    3E 03   LD A, 03
200C    D3 F9   OUT F9          Ausgabe von 03 auf Port B
200E    76      HALT
```

Ist die PIO einmal initialisiert (in unserem Beispiel auf den Adressen 2000 bis 2005), ist das für weitere Ausgaben nicht mehr erforderlich.

Voraussetzung dafür ist natürlich, daß 'wir die PIO in der gleichen Betriebsart weiterarbeiten lassen wollen.
Alles verstanden?
Wir wollen jetzt etwas "Leben" in unsere Zusatzschaltung bringen und sie einfach einmal binär zählen lassen (nur Port A). Das Programm dazu sieht z. B. so aus:

```
2000   01 00 00      LD BC, 0000
2003   3E 0F         LD A, 0F       PIO-Mode 0,
2005   D3 FA         OUT FA         Port A
2007   3E FF         LD A, FF
2009   3C            INC A
200A   D3 F8         OUT F8
2000   10 FE         DJNZ FE
200E   0D            DEC C          Zeitschleife
200F   20 FB         JRNZ FB
2011   18 F6         JR F6
```

Oder eine Lauflichtsteuerung über 8 Stufen?

```
2000   3E 0F         LD A, 0F       PIO-Mode 0,
2002   D3 FA         OUT FA         Port A
2004   3E 01         LD A, 01
2006   06 FF         LD B, FF       Geschwindigkeit
2008   0E FF         LD C, FF       bei 10 schnell
200A   0D            DEC C
200B   20 FD         JRNZ FD
200D   10 F9         DJNZ F9
200F   D3 F8         OUT F8
2011   07            RLCA
2012   18 F2         JR F2
```

Durch Variation der Geschwindigkeit auf Adresse 2007, durch den Inhalt der Adresse 2005 (z. B. 03 oder 07 erzeugen Striche) und durch Änderung des Inhalts von 2011 (0F ist der Befehl RRCA und verschiebt das Ganze in die andere Richtung)

können wir unser Lauflicht beliebig gestalten.
Unser Leuchtpunkt (oder -strich) kann auch hin und her flitzen, was durch das folgende Programm realisiert wird:

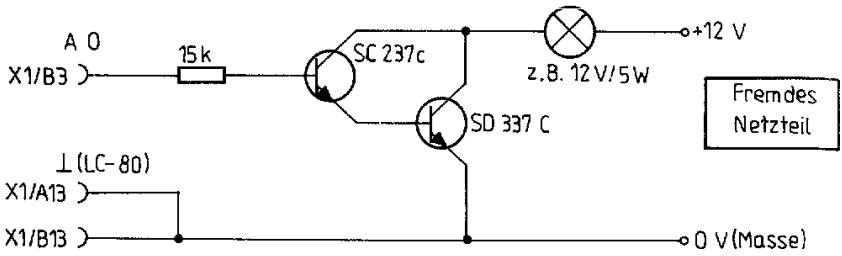
```
2000   3E 0F          LD A, 0F
2002   D3 FA          OUT FA
2004   3E 01          LD A, 01              03     07
                                            Punkt oder Striche
2006   16 07          LD D, 07              06     05
2008   CD 00 21       CALL 2100
200B   07             RLCA
2000   15             DEC D
200D   20 F9          JRNZ F9
200F   16 07          LD D, 07              06     05
2011   CD 00 21       CALL 2100
2014   0F             RRCA
2015   15             DEC D
2016   20 F9          JRNZ F9
2018   18 EC          JR EC
```

Und das dazugehörige Unterprogramm:

```
2100   06 FF          LD B, FF              Geschwindigkeit
2102   0E FF          LD C, FF
2104   0D             DEC C
2105   20 FD          JRNZ FD
2107   10 F9          DJNZ F9
2109   D3 F8          OUT F8
210A   C9             RET
```

Soll ein Strich hin und her laufen, müssen die jeweils untereinander stehenden Alternativdaten auf den Adressen 2005, 2007 und 2010 geändert werden - und zwar alle!
Diese kleine Auswahl von Lichteffekten mag für's erste genügen. Wer diese Fähigkeiten unseres LC-80 ausbauen will, kann die Bits B 0 ... B 3 des Ports B mit heranziehen oder weitere Effekte ausprobieren.

Natürlich lassen sich auch stärkere Lasten als Leuchtdioden ansteuern, z. B. kräftige Diskolampen. Dabei ist jedoch immer zu beachten, daß keine nennenswerten Steuerleistungen aus dem LC-80 entnommen werden dürfen.
Eine mögliche Ansteuerung über Transistoren zeigt die folgende Abbildung:



Auch hier wurde wieder nur ein Ausgang dargestellt!

Es muß nochmals darauf hingewiesen werden, daß bei solchen Leistungssteuerungen externe, zusätzliche Netzteile verwendet werden müssen und die Steuerleistungen, wie hier z. B. durch eine Darlingtonschaltung, so gering wie möglich gehalten werden!
Hände weg von Ansteuerungen direkter Netzverbraucher!!
Die dazu einzuhaltenden Vorschriften sind so vielfältig, daß jeder Versuch unterbleiben sollte.

11.2.2. Mode 3

In unserem Kurs über die Möglichkeiten der PIO haben wir uns mit dem Mode 0 (Byte-Ausgabe) beschäftigt, ohne spezielle Eigenschaften, z. B. Interruptverhalten und Hand-Shake-Betrieb, zu behandeln. Auch Mode 1 (Byte-Eingabe) und Mode 2 (bidirektionaler Betrieb) überspringen wir.
In diesem Abschnitt wollen wir uns mit Mode 3 (Einzelbitsteuerung) anfreunden. Diese Arbeitsweise der PIO ist sehr interessant, weil wir völlig frei festlegen können, welche PIO-Leitungen als Eingänge und welche als Ausgänge arbeiten sollen. Diese Arbeitsweise haben wir bei unserem IC-Tester (Kapitel 9) schon kennengelernt.
Auch hier beschränken wir uns zunächst auf Port A und vernachlässigen alle Interruptprobleme.
Im Abschnitt 11.2. haben wir gelernt, mit einem Steuerwort als erstes die Betriebsart festzulegen. Dies geschieht auch im Mode 3 so, das Steuerwort wäre hier z. B. CF. Da die Bits D 5 und D 4 frei verfügbar sind, könnte hier genauso gut FF stehen (wie im Kapitel 9).
Die Initialisierung erfolgt damit so:

```
33 CF   LD A, CF      Das Steuerwort CF wird in den
D3 FA   OUT FA        Akku geladen. Der Inhalt von A wird
                      (als Steuerwort) für den Kanal A zur
                      Wahl des Mode 3 benutzt.
```

Und jetzt kommt eine Besonderheit im Mode 3.
Nach der Ausgabe des Steuerwortes zur Festlegung der Betriebsart muß im Mode 3 ein weiteres Steuerwort zur Festlegung der Ein- bzw. Ausgabefunktion des betreffenden PIO-Anschlusses gesendet werden.
Dabei definiert

eine "1" den Anschluß als Eingang (Eselsbrücke: 1=Eingang)
eine "0" den Anschluß als Ausgang.

Wir probieren es einmal mit folgendem Muster:

A7	A6	A5	A4	A3	A2	A1	A0
1	0	1	0	1	0	1	0

Also:

```
3E AA   LD A, AA      Das Steuerwort AA wird in den
D3 FA   OUT FA        Akku geladen. Der Inhalt von A wird
                      (als Steuerwort) für den Kanal A zur
                      Festlegung von Ein- und Ausgängen
                      benutzt.
```

Zur Kontrolle geben wir jetzt FF als Datenwort an Adresse F8 aus:

```
3E FF   LD A, FF  }   Datenausgabe FF auf Port A
D3 F8   OUT F8    }
```

Zusammenhängend geben wir also ein:

```
2000   3E CF   LD A, CF  }  Auswahl der Betriebsart
2002   D3 FA   OUT FA    }  Mode 3, Port A
2004   3E AA   LD A, AA  }  Eingangs- bzw. Ausgangs-
2006   D3 FA   OUT FA    }  definition Port A (A 7,
                             A 5, A 3, A 1 Eingänge,
                             Rest Ausgänge)
2008   3E FF   LD A, FF  }  Ausgabe des Datenwortes FF
200A   D3 F8   OUT F8    }  auf Port A (FF heißt ja, daß
                             alles logisch "1" ist)
200C   76      HALT
```

Wir stellen fest, daß unser eingegebenes Datenwort nicht erscheint, sondern nur die LEDs der Anschlüsse A 0, A 2, A 4 und A 6 leuchten. Das ist ganz richtig, denn wir haben nur

diese Leitungen als Ausgänge definiert. Wir können das noch deutlicher verfolgen, wenn wir auf Adresse 2009 einmal 01 und einmal 02 eintragen. 01 heißt, daß auf A 0 eine "1" aus-gegeben werden soll (A 0 ist als Ausgang definiert → (LED leuchtet). 02 heißt, daß auf A 1 eine "1" ausgegeben werden soll (A 1 ist aber als Eingang definiert und damit erfolgt keine Ausgabe → (LED leuchtet nicht).

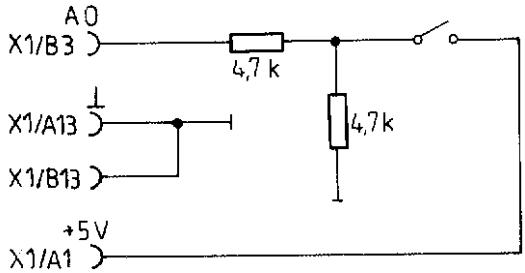
Die im Mode 3 mögliche Bit-Eingabe wird im folgenden Abschnitt mit behandelt, hier nur soviel:

- Die als Eingänge definierten Anschlüsse können von außen mit High- oder Low-Pegel belegt werden.
- Ein IN-Befehl übernimmt diese logischen Zustände in die CPU bzw. in den Akkumulator). Dabei ist zu beachten, daß die als Ausgänge geschalteten Leitungen dann den Pegel führen, den sie zuletzt als Ausgänge hatten. Das heißt, daß nach IN-Operationen z. B. in A abgelegte Datenwort setzt sich zusammen aus den Eingangswerten (für die als Eingänge geschalteten Anschlüsse) und den Ausgangswerten (für die sie Ausgänge geschalteten Anschlüsse).

11.2.3. Mode 1

Diese Betriebsart realisiert eine byteweise Eingabe, d. h. alle Anschlüsse eines PIO-Kanals sind als Eingänge geschaltet. Auch hier wäre es sinnvoll - analog zum LED-Anschauungsmodell für die Ausgabe - eine einfache und übersichtliche Eingabeeinheit aufzubauen. Wer das tun möchte - die nach-stehende Schaltung realisiert es mit normalen Kontakten.

Sehr praktisch sind sog. DIL-Switch-Schalter, bei denen z. B. 8 Schiebekontakte in IC-Form aufgebaut sind. Die Schaltung zeigt wieder nur eine Leitung:



Die eigentlichen Anwendungsgebiete von Mode 1 sind aber andere, wie bsw.:

- Eingabe von peripheren Geräten (Lochstreifenleser, Tastatur usw.),
- Eingabe aus elektronischen Einheiten - unser DVM war ein Beispiel.

Bei der Programmierung erfolgt, wie wir nun schon wissen, zunächst die Initialisierung, diesmal mit dem Steuerwort 4F:

```
3E 4F   LD A, 4F        Dez Steuerwort 94 wird in den
D3 FA   OUT FA          Akku geladen. Der Inhalt von A wird
                        (als Steuerwort) für den Kanal A zur
                        Festlegung des Modes 1 benutzt.
```

Der nächste Befehl könnte ein IN n - Befehl sein, der im Gegensatz zum OUT n - Befehl die am PIO-Port A liegenden Informationen in den Akku übernimmt.

Zusammengefaßt sieht unser Eingaberitual dann so aus:

```
2000   3E 4F   LD A, 4F  }   Initialisierung Port A
2002   D3 FA   OUT FA    }   Mode 1
2004   DB F8   IN F8          Übernahme der am Port A an-
                               liegenden Daten in den Akku
2006   76      HALT
```

Die einfachste Möglichkeit, diese Funktionen zu überprüfen, ist die Benutzung der Registeranzeige. Dazu wird in die Speicherzellen:

```
2340   C3
2341   90
2342   06
```

eingegeben (sh. a. Bedienungsanleitung LC-80, S. 17). Jetzt ist es einfach, durch Drücken von NMI den Inhalt des Doppelregisters AF anzuschauen.
Die beschriebene Eingabe erfolgt natürlich nur einmal. Wenn der Eingabekanal ständig überwacht werden soll, muß der IN-Befehl in einer Schleife dauernd durchlaufen werden, etwa so:

```
2000   3E 4F          LD A, 4F
2002   D3 FA          OUT FA
2004   DB P8          IN F8   4
2006   CD C3 04 *     CALL DADP
2009   CD 83 04 *     CALL DAK 2
2000   18 F6          JR F6
```

Gleichzeitig haben wir uns die Möglichkeit geschaffen, auf den beiden rechten Stellen des Displays den Akkumulatorinhalt in Hexa-Darstellung zu beobachten.
Ein Nachteil der beschriebenen Verfahrensweise ist, daß zur Überwachung von Eingabedaten der Eingabekanal ständig abgefragt werden muß. Die CPU ist dauernd beschäftigt und kann nur unter Schwierigkeiten ihre anderen Aufgaben bearbeiten.

Aus diesem Grund besitzt unser Rechner eine sehr komfortable Einrichtung, das sog. Interruptsystem. Wir werden dessen Grundlagen im nächsten Abschnitt kennenlernen.

11.3. Interruptsystem

Wohl kaum ein Problem der Mikroprozessortechnik ist für den Anfänger so schwer zu durchschauen und praktisch zu handhaben wie das Interruptsystem. Gleich anfangs muß jedoch gesagt werden, daß hier nur erste Hinweise und Anregungen gegeben werden können. Die umfassende und über das Anfängerniveau hinausgehende Darstellung muß weiterführender Literatur vorbehalten bleiben.

11.3.1. Was soll und was ist ein "Interrupt"?

Stellen wir uns einmal vor, unser Haus hätte keine Klingel. Wenn wir nun Besuch erwarten, müßten wir eigentlich unentwegt aus dem Fenster schauen, um ihn nicht zu verpassen.
Alle anderen Tätigkeiten müßten im wesentlichen unterbleiben. Mit Klingel dagegen ist das viel praktischer. Wir können fast alles tun wie gewohnt - erst beim Ertönen der Glocke "unterbrechen" wir unsere Tätigkeit, wobei wir z. B. erstmal alles aus der Hand legen, und widmen uns dann dem Empfang unseres Besuches.

Genauso ist es beim Telefon - kein Mensch lauscht den ganzen Tag am Hörer. Erst wenn es klingelt, werden die normalen Tätigkeiten "unterbrochen", das Telefongespräch erledigt und dann geht es mit dem Tagesablauf normal weiter. Unsere "Unterbrechungen" haben ein wichtiges Merkmal, sie kommen alle von außen und unsere Reaktion auf sie besteht aus einer geordneten Reihenfolge von Abläufen (und das ist wichtig - wir stellen erst einmal den Staubsauger weg, drehen das Radio leise usw.).

Auch unser Rechner kommuniziert bei bestimmten Anwendungen mit seiner Umwelt, erwartet Daten von außen, gibt Kommandos ab usw. Dabei wäre es günstig, analog zum obigen Beispiel "aus dem Leben" ein "Klingelsystem" zu haben, das dem Rechner ermöglicht, seine "Arbeit" geordnet zu unterbrechen (engl.: interrupt), wenn Signal aus seiner Umwelt dies erfordern. Danach müßte eine automatische Rückkehr zur normalen Programmbearbeitung erfolgen.

Unser Rechner hat ein solches System, das aber noch viel umfangreicher ist als unser Haushalt. Es besitzt bei vollem Ausbau Hunderte von "Klingeln" und ist daher eher mit einer Intensivstation im Krankenhaus zu vergleichen.

Ebenso wie dort hat jeder Baustein eines Mikrorechnersystems, der mit der Umwelt kommuniziert, Interruptmöglichkeiten. Im LC-80 sind das die USER-PIO und der CTC-Baustein. Beide können zur Interrupterzeugung herangezogen werden. Wenn das bei bestimmten Anwendungen erfolgen soll, müssen programmtechnisch Vorbereitungen dazu getroffen werden.

Eine Interruptmöglichkeit haben wir, die die Programmbearbeitung immer unterbricht, den sog. nicht maskierbaren Interrupt (NMI). Wie wir wissen, existiert eine Taste mit gleicher Bezeichnung auf unserer Tastatur. Im folgenden Kapitel werden wir dies programmtechnisch nutzen.

Wenn es einen nicht maskierbaren Interrupt gibt, ist zu erwarten, daß die anderen Interrupts maskierbar sind ...

Was ist nun maskierbar?
Es heißt nichts anderes, als daß wir mit verschiedenen Möglichkeiten (Masken) zu bestimmten Zeiten, an bestimmten Programmstellen oder beim Zutreffen bestimmter Bedingungen Interrupts zulassen oder verbieten können - und das für jeden Eingabekanal getrennt.
Manche dieser Interruptverbote erfolgen "automatisch", so z. B. wird ein Interrupt verboten, wenn gerade ein anderer Interrupt abgearbeitet wird. Der anstehende Interrupt wird dann in Speicherzellen (den sog. Interrupt-Flip-Flops) der CPU gespeichert, bis diese wieder frei dafür ist. Wie im "richtigen Leben" gibt es auch hier Prioritäten bei der Abarbeitung von Interrupts. Diese können z. B. durch die Verdrahtung der Bauelemente im System, aber auch programmtechnisch festgelegt werden.

So, das soll für's erste genügen. Wir wissen jetzt, wozu Interrupts dienen, welche grundsätzlichen Möglichkeiten sich hier anbieten und daß noch einiges gelernt werden muß, z. B. die praktische Handhabung von Interrupts.

11.3.2. Praktische Handhabung von Interrupts

Wir müssen uns noch einmal erinnern, was wir mit der Anwendung von Interrupts eigentlich erreichen wollen und was der Rechner alles tun bzw. wissen muß:

- Der oder die betreffenden Eingänge müssen interruptfähig gemacht werden.
- Der Rechner muß wissen, was er beim Auftreten eines bestimmten Interrupts tun muß, z. B. sich die Stelle merken, an der das Hauptprogramm verlassen wurde und er muß erfahren, wo er die Vorschrift für die Abarbei-

tung des Interrupts findet. Das Ganze heißt "Interruptroutine".

- Nach Abarbeitung der Interruptroutine muß er in das Hauptprogramm zurückkehren. Alle Registerinhalte, die in der Interruptroutine verändert werden, müssen mit PUSH- und POP-Befehlen gerettet werden, wenn sie im Hauptprogramm wichtig sind.

Und genauso gehen wir praktisch vor:
Zum besseren Verständnis nehmen wir uns das Quarzuhrenbeispiel aus Kapitel 12 vor:

- Auf Adresse 2200 verbieten wir mit dem Befehl DI zunächst alle Interrupts (außer NMI).
- Wir legen auf Adresse 2220 den Interruptmode (IM 2) fest, die drei möglichen Interruptmodes werden in [1] ausführlich beschrieben.
- Für uns ist im Augenblick nur IM 2 interessant. Dieser ist dadurch gekennzeichnet, daß ein 16Bit-Zeiger gebildet wird, der zum Herausholen der Startadresse der Interruptroutine dient. Dabei müssen die oberen 8 Bit des Zeigers im Interruptregister I gespeichert werden. Das tun wir, indem wir auf Adresse 221F erst A mit 23 und dann I mit A laden. 23 ist der höherwertige Teil unseres Zeigers (ein Teil der Adresse!).
- Der niederwertige Teil des Zeigers wird von dem Schaltkreis geliefert, der den Interrupt anfordert, also unsere PIO. Das tut sie, nachdem sie über einen Ausgabebefehl (Steuerwort) mit dem sog. Interruptvektor (Low-Teil) geladen wurde. In unserem Fall findet das auf Adresse 222B über einen Ladebefehl und einen Ausgabebefehl statt. Der Lowteil ist dabei 50, so daß der vollständige Zeiger 2350 heißt. Hier ist wichtig, daß das niederwertigste Bit des Low-Teiles immer 0 sein muß.

- Auf den Speicherstellen 2350 und 2351 muß nun die Startadresse des Interruptprogrammes stehen, in unserem Fall ist das die Adresse 22A0.
- Auf Adresse 222F wird das sog. Interrupt-Steuerwort geladen. Das ist für den PIO-Mode 3 wichtig. Hier kann über das

  Bit 7 die Interruptfreigabe (ja/nein),
  Bit 6 die Wahl der Verknüpfung (Und/Oder),
  Bit 5 die Festlegung des aktiven Pegels (Low/High) und über
  Bit 4 die Festlegung einer Maske (ja/nein)

  erfolgen.
  Wichtig ist, daß die unteren 4 Bit D 0 ... D 3 immer 7 (in Hexa-Schreibweise) sein müssen, in unserem Beispiel 97.
- Wenn Bit 4 = 1, muß das nächste an diesen PIO-Kanal gerichtete Steuerwort eine Maske darstellen, d. h. es werden die Bits ausgewählt, die auf die Interruptbildung Einfluß nehmen sollen, in unserem Beispiel nur B 0. Das geschieht auf Adresse 2233.
- Auf Adresse 2279 erfolgt jetzt mit dem Befehl EI die Interruptfreigabe.
- Das eigentliche Interruptprogramm beginnt, wie schon gesagt, auf Adresse 22A0. Dieses Programm kann man als eine Art Unterprogramm betrachten, das zu jeder Sekunde, und zwar beim Übergang von High nach Low auf der B 0 - Leitung, angesprungen wird. In unserem Beispiel wird der Uhrzeitspeicher um eine Sekunde weitergestellt sowie beide Alarmzeiten mit der aktuellen Uhrzeit verglichen.
  Für unseren Interrupt - "Lehrgang" sind folgende zwei Punkte wichtig:

1. Wenn die CPU einen Interrupt empfangen hat und ausführt, werden automatisch weitere Interrupts von ihr nicht akzeptiert. Das wirkt sich genauso aus wie ein DI-Befehl. Erst durch einen EI-Befehl wird der nächste Interrupt erlaubt. Wir finden ihn im Interruptprogramm auf Adresse 2279. Diese Interruptfreigabe dürfen wir bei derartigen Programmen nie vergessen!
2. Jedes Unterprogramm wird mit RET (C9) abgeschlossen. Analog müssen Interruptprogramme mit einem speziellen Return-Befehl RETI (ED 4D) abgeschlossen werden. Auch das dürfen wir nie vergessen, da nur so auch unsere PIO wieder interruptfähig gemacht werden kann.

Sehr kompliziert!
Aber wir lassen uns nicht entmutigen. Mit Hilfe weiterführender Literatur werden wir später die Feinheiten begreifen.

Manches in unserem Mikroprozessorsystem scheint uns noch sehr umständlich zu sein. Wir müssen aber wissen, daß unser LC-80 nur eine sehr kleine Maschine ist. Das Mikroprozessorsystem U 880 mit seinen Komponenten ist äußerst ausbaufähig. So lassen sich z. B. 128 PIO-Ports ansteuern (in Worten: einhundertachtundzwanzig). Das sind 1024 einzelne Leitungen! Jeder Kanal kann einen eigenen Interrupt auslösen, d. h. jedesmal kann ein anderes Interruptprogramm ablaufen. Auch unterschiedliche Wertigkeiten lassen sich festlegen. Jeder PIO-Baustein besitzt eine durch den Aufbau der Schaltkreise - also hardwaremäßig festgelegte Interruptpriorität.

Also - Kopf hoch, alles muß man nicht sofort wissen und können. Eine Eigenschaft unserer PIO wollen wir aber noch kennenlernen, nämlich die Möglichkeit des sog. Hand-Shake-Betriebes.

11.4. Die "Hand-Shake" – Signale

Unser USER-Bus führt noch 4 Signale der PIO nach außen, die wie folgt bezeichnet sind:

ARDY, /ASTB,
BRDY, /BSTB.

Ein erfolgreicher Handel wird mit einem Händedruck (Hand-Shake) abgeschlossen. Unser Rechner "handelt" mit Daten, über den USER-Bus findet der Datenaustausch mit peripheren Geräten statt. Unsere 4 Signale dienen der gegenseitigen Bestätigung von Rechner und peripherem Gerät über die Bereitschaft zur Übergabe und Übernahme von Daten. So gilt z. B. im Mode 0:

Wenn der LC-80 z. B. über den PIO-Port A Daten bereitgestellt hat (nach einem OUT-Befehl), geht ARDY auf High (ARDY ist High-aktiv).

Wenn die periphere Schaltung die Daten übernommen hat, gibt sie einen Low-Impuls auf /ASTB (Low-aktiv). Das setzt seinerseits ARDY zurück auf Low.

Die Bedeutung der beiden Signale ist bei den vier Betriebsarten der PIO-Kanäle unterschiedlich. Wichtig für uns ist, daß die Quittierungssignale im Mode 3 nicht verwendet werden dürfen. Einzelheiten über das Verhalten der PIO finden wir z. B. in [2].

Wozu brauchen wir eigentlich eine solche Einrichtung, genügt es nicht, z. B. Daten auszugeben, die die periphere Schaltung sofort verarbeitet?
Genau hier liegt das Problem - im "sofort"!
Wir erinnern uns, daß Mikrorechner nicht nur schnelle Elektronik steuern sollen, sondern auch langsame Mechanik, wie Drucker, Magnetbandeinheiten oder gar mechanische Stellglieder, die mit Motoren in den jeweiligen Zustand "gefahren" werden müssen. Besonders hier wird der große Nutzen dieser

Quittierungssignale deutlich. Erst die Bestätigung des "Empfängers" - das kann das periphere Gerät oder der Mikrorechner selbst sein - erlaubt einen neuen Datentransfer.
So haben wir sicher schon überlegt, welchen Sinn "Interrupts" bei reiner Datenausgabe der PIO haben könnten.
Das wird jetzt klar - Daten werden ausgegeben, das periphere Gerät übernimmt sie und quittiert das über /ASTB bzw. /BSTB - und das kann einen Interrupt auslösen!

Eine weitere Möglichkeit, die wir z. B. im Kapitel 13 ausnutzen, ist die Verbindung der beiden zusammengehörigen Quittierungssignale, z. B. ARDY mit /ASTB.
Welchen Sinn hat das?
Der D/A-Converter im Kapitel 13 setzt die digitale Information von Port A und Port B uni in analoge Spannungswerte. Dabei sollen Port A und Port B gleichzeitig, über den D/A-Converter umgesetzt werden. Das aber geht bei der PIO nicht, da jeder Port einzeln über einen OUT-Befehl geladen werden muß. Wenn diese Zeitdifferenz stört, können die Quittierungssignale helfen. Bei Verbindung von ARDY und /ASTB entsteht ein kurzer Impuls, der in unserem Beispiel als Übernahmeimpuls in die 12 Register (V 1012 D) dient. Dabei werden immer die Daten auf dem Port B zuerst ausgegeben (sie bleiben dort stabil stehen, bis der nächste OUT-Befehl für Port B erfolgt) und dann die Daten auf Port A. Jetzt wird der kurze Übernahmeimpuls gebildet und die Daten von Port B und Port A gleichzeitig in die V 4012 D eingeschrieben.

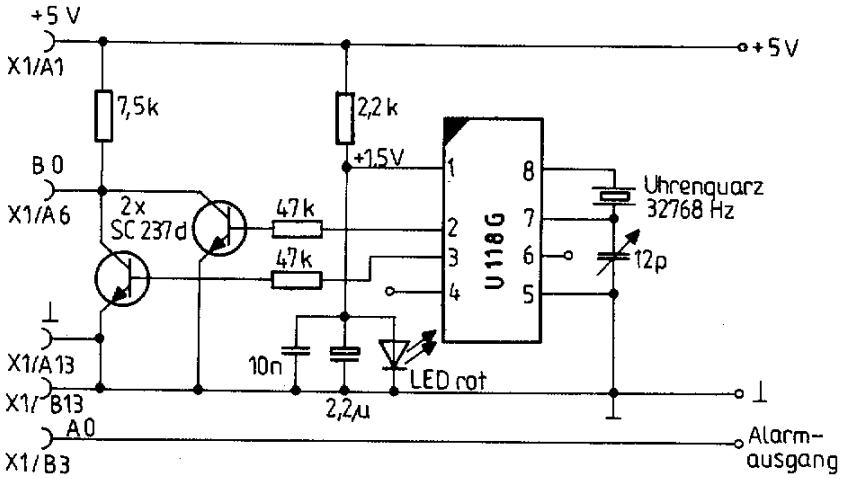
12. Eine interessante Quarzuhr.

In der Bedienungsanleitung des LC-80 finden wir u. a. ein Anwendungsbeispiel als Digitaluhr mit Wecker. Da diese Uhr mit dem internen Takt des LC-80 arbeitet, ist die erreichbare Ganggenauigkeit sehr gering. Immerhin bietet dieses Programmbeispiel die Möglichkeit, die Taktfrequenz zu kontrollieren und gegebenenfalls zu korrigieren. Dazu ist der Einstellregler (vorsichtig!) so zu verstellen, daß die Uhr annähernd richtig geht.
Für eine "ordentliche" Uhr ist das Programm jedoch ungeeignet; das geht nur mit Hilfe einer Quarzzeitbasis. In diesem Kapitel wollen wir uns so etwas aufbauen und dabei die Kenntnisse über die USER-PIO und deren Programmierung anwenden. Andererseits sollen Möglichkeiten aufgezeigt werden, was man mit einer solchen Quarzzeitbasis bzw. dem Uhrenprogramm sonst noch alles anfangen kann.

12.1. Die Quarzzeitbasis

Unsere Quarzuhr wird über eine extern anzuschließende Quarzzeitbasis angesteuert. Am einfachsten laßt sich diese mit modernen CMOS-Uhren-ICs realisieren, wie z. B. mit U 118 G oder (mit Einschränkungen) U 114 D aus dem veb mikroelektronik "karl marx" erfurt - stammbetrieb. Wichtig ist, daß das Quarzmodul jede Sekunde einen kurzen Impuls abgibt, wie dies z. B. für Quarz-Analog-Uhren erforderlich ist.

Die an den beiden Ausgängen des Uhren-ICs erzeugten Impulse werden auf TTL-Pegel verstärkt, logisch verkoppelt und über eine PIO-Leitung eingegeben. Da alle genannten Uhrenschaltkreise eine Betriebsspannung von 1,5 V benötigen, wird diese auf der Zusatzleiterplatte ebenfalls erzeugt. Für die Sekundentakteingabe wählen wir B 0 der USER-PIO, während A 0 im Beispiel als Alarmausgabe benutzt wird.
Die komplette Schaltung für die Anwendung des U 118 G zeigt folgende Abbildung:



Wer es mit einer anderen "Hardware" versuchen will, bitte. Wichtig ist lediglich, daß an B 0 jede Sekunde ein (TTL-) Impuls (Low-aktiv) erzeugt wird, der nach einigen Millisekunden wieder auf High geht.
Mit dem U 114 D ist das nicht ganz so einfach. Dieser IC hat zwar auch zwei Ausgänge, diese führen aber für jeweils eine ganze Sekunde H- oder L-Pegel. Ein Ausweg wäre die Benutzung nur eines Ausganges, der aber dann eine Periodendauer von

2 Sekunden hat. Das können wir programmtechnisch dadurch ausgleichen, daß wir im Programm auf Adresse 22A6 den zu addierenden Wert von 01 auf 02 ändern. Die Uhr schaltet dann einfach alle 2 Sekunden um 2 Sekunden weiter. Aber auch die Differenzierung beider Ausgangssignale wäre eine Möglichkeit. Die Betriebsspannung des Uhren-ICs wird durch eine Stabilisierungsschaltung mit roter Leuchtdiode auf + 1,5 V gehalten.

12.2. Das Programm

Das folgende Programm realisiert eine quarzgesteuerte Digitaluhr mit unabhängig programmierbarer Alarmein- und -ausschaltung. Das Konzept der Interruptsteuerung belastet den LC-80 zeitlich nur geringfügig, so daß gleichzeitig andere Programme abgearbeitet werden können. Deshalb wurde das Uhrenprogramm auf den hinteren Bereich des Arbeitsspeichers verlegt. Die Adressen 2000 ... 21CF sind für anderen Gebrauch frei verfügbar. Zunächst einige kurze Erläuterungen zum Programmaufbau. Das Programm ist aus vier wesentlichen Teilen zusammengesetzt, und zwar aus:

- dem Grundprogramm (Adresse 2200 ... 2289), das die Eingabe der verschiedenen Zeiten

  Uhrzeit
  Alarm ein (AE) und
  Alarm aus (AA)

  realisiert, die PIO initialisiert, den Start der Uhr erlaubt und ein Verlassen des Uhrzeitprogrammes ermöglicht,

- dem Interruptprogramm (Adresse 22A0 ... 22F6), das das eigentliche Rechenprogramm zur Aktualisierung der Uhrzeit und deren Vergleich mit den beiden Alarmzeiten realisiert,
- - dem Unterprogramm UA zur Darstellung der Zeitspeicherinhalte auf dem Display und
- dem Unterprogramm VP zur Realisierung einer taschenrechnerähnlichen Eingabe der verschiedenen Zeiten über die Tastatur.

Dazu kommen die Programmteile ab Adresse 2340 zum "Wiedereinsteigen" in das Uhrenprogramm bei NMI – Betätigung und die auf den Adressen 2350 und 2351 stehende Sprungadresse 22A0 zum Start des Interruptprogrammes.

Hier das aufgelistete Programm:

Grundprogramm

```
2200   F3           DI                  ab jetzt Inter-
                                         rupt verboten
2201   21 00 00     LD HL, 0000         Laden des Uhr-
2204   22 00 23     LD (2300), HL       zeitspeichers
2207   22 01 23     LD (2301), HL       mit 00 00 00
220A   21 FF FF     LD HL, FFFF         Laden der Alarm-
220D   22 04 23     LD (2304), HL       zeitspeicher
2210   22 07 23     LD (2307), HL       mit FF FF
2213   32 AE        LD A, AE            Laden des Symbols
2215   32 03 23     LD (2303), A        "Alarm ein" (AE)
2218   3E AA        LD A, AA            Laden des Symbols
221A   32 06 23     LD (2306), A        "Alarm aus" (AA)
221D   ED 5E        IM 2                Interruptmode 2
221F   3E 23        LD A, 23            High-Teil der
2221   ED 47        LD I, A             Sprungadresse bei
                                         INT, Interruptre-
                                         gister laden mit A
```

```
2223   3E FF          LD A, FF        }  PIO-Mode 3 für
2225   D3 FB          OUT FB          }  Port B
2227   3E F1          LD A, F1        }  E/A-Definition für
2229   D3 FB          OUT FB          }  Port B, nur B 0
                                          ist Eingang
222B   3E 50          LD A, 50        }  Interruptvektor
222D   D3 FB          OUT FB          }  Low-Teil der
                                          Sprungadresse
222F   3E 97          LD A, 97        }  Interrupt-
2231   D3 FB          OUT FB          }  steuerwort
2233   3E FE          LD A, FE        }  Maske, nur B 0 ist
2235   D3 FB          OUT FB          }  interruptfähig
2237   3E FF          LD A, FF        }  PIO-Mode 3 für
2239   D3 FA          OUT FA          }  Port A
223E   3E 00          LD A, 00        }  E/A-Definition für
223D   D3 FA          OUT FA          }  Port A, alle sind
                                          Ausgänge
223F   11 03 23       LD DE, 2303        Alarmzeit AE eintrage
2242   CD E0 21       CALL UA
2245   CD 5A 04 *     CALL DAK 1
2248   FE 10          CP 10              [+]-Taste gedrückt?
224A   28 05          JRZ 05
2240   CD D0 21       CALL VP
224F   18 EE          JR EE
2251   11 06 23       LD DE, 2306        Alarmzeit AA eintrgen
2254   CD E0 21       CALL UA
2257   CD 5A 04 *     CALL DAK 1
225A   FE 10          CP 10              [+]-Taste gedrückt?
225E   28 05          JRZ 05
225E   CD D0 21       CALL VP
2261   18 EE          JR EE
2263   11 00 23       LD DE, 2300        Uhrzeit eintragen
2266   CD E0 21       CALL UA
2269   CD 5A 04 *     CALL DAK 1
226C   FE 10          CP 10              [+]-Taste gedrückt?
226E   28 CF          JRZ CP
```

```
2270   FE 12          CP 12          [EX]-Taste gedrückt?
2272   28 05          JRZ 05         (wenn ja, läuft Uhr an)
2274   CD D0 21       CALL VP
2277   18 EA          JR EA
2279   FB             EI             Interrupt ab jetzt erlaubt
227A   CD 83 04 *     CALL DAK 2
227D   FE 01          CP 01          [ST]-Taste gedrückt? wenn ja,
                                      Verlassen des laufenden
                                      Uhrenprogrammes)
227F   28 08          JRZ 08
2281   11 00 23       LD DE, 2300
2284   CD E0 21       CALL UA
2287   18 F1          JR F1
2289   C3 00 00       JMP 0000       "künstliches" RESET
```

Interrupt-Programm

```
22A0   E5             PUSH HL        Retten von HL
22A1   F5             PUSH AF        Retten von AF
22A2   3A 00 23       LD A, (2300)   A mit sec. laden
22A5   C6 01          ADD 01         zu A 01 addieren
22A7   27             DA             Ergebnis in BCD korrigieren
22A8   FE 60          CP 60          Vergleich mit 6o, Carry hier
                                      = 1, wenn = 60, dann A = 00
                                      setzen, Carry = 0 setzen,
                                      Carry negieren
22AA   20 01          JRNZ 01
22AC   AF             XOR A
22AD   3F             CCF
22AE   32 00 23       LD (2300), A   2300 mit neuen sec. laden
```

```
22B1   3A 01 23      LD A, (2301)   A mit min. laden
22B4   CE 00         ADC 00         Carry-Bit addieren
22B6   27            DAA            Ergebnis in BCD umformen
22B7   FE 60         CP 60          Vergleich mit 60, Carry hier
                                    = 1, wenn = 60, dann A = 00
                                    setzen, Carry = 0 setzen,
                                    Carry negieren
22B9   20 01         JRNZ 01
22BB   AF            XOR A
22BC   3F            CCF
22BD   32 01 23      LD (2301), A   2301 mit neuen min. laden
22C0   3A 02 23      LD A, (2302)   A mit Std. laden
22C3   CE 00         ADC 00         Carry-Bit addieren
22C5   27            DAA            Ergebnis in BCD umformen
22C6   FE 24         CP 24          Vergleich mit 24, wenn = 24,
                                    dann A = 00 setzen
22C8   20 01         JRNZ 01
22CA   AF            XOR A
22CB   32 02 23      LD (2302), A   2302 mit neuen Std. laden
22CE   21 05 23      LD HL, 2305    HL mit 2305 laden (h AE)
22D1   BE            CPM            h mit hAE vergleichen
22D2   20 0B         JRNZ 0B        Bei Übereinstimmung
22D4   3A 01 23      LD A, (2301)   A mit min. laden
22D7   2B            DEC HL         HL jetzt 2304 (min. AE)
22D8   BE            CPM            min. mit min.AE vergleichen
22D9   20 04         JRNZ 04        Bei Übereinstimmung
22DB   3E 01         LD A, 01       High-Pegel an A 0, bei
22DD   D3 F8         OUT F8         Alarm ausgeben
```

```
22DF   3A 02 23      LD A, (2302)   Der selbe Vergleich
22E2   21 08 23      LD HL, 2308    der aktuellen Uhrzeit
22E5   BE            CPM            mit der Alarmaus-
22E6   20 0B         JRNZ 0B        schaltzeit AA wie
22E8   3A 01 23      LD A, (2301)   oben
22EB   2B            DEC HL
22EC   BE            CPM
22ED   20 04         JRNZ 04
22EF   3E 00         LD A, 00       Bei Übereinstimmung
22F1   D3 F8         OUT F8         Low-Pegel an A 0 bei Alarm
                                    ausgeben
22F3   F1            POP AF         AF in alten Zustand
22F4   E1            POP HL         HL in alten Zustand
22F5   FB            EI             neuer Interrupt erlaubt
22F6   ED 4D         RETI           zurück aus Interruptprogramm
                                    ins Hauptprogramm

2340   E1            POP HL         Rückkehr in das
2341   21 7A 22      LD HL, 227A    Uhrenprogramm mit
2344   E5            PUSH HL        NMI-Taste
2345   FB            EI
2346   ED 45         RETN
2350   A0                           Startadresse des
                                    Interruptprogrammes
2351   22                           Interruptvektor
                                    steht auf 2350
```

Unterprogramm VP

```
21D0   62           LD H, D      } aktuellen Inhalt von DE
21D1   6B           LD L, E      } in HL laden
21D2   23           INC HL
21D3   ED 6F        RLD          ) Verschiebung von jeweils
21D5   23           INC HL       ) 4 Bit nach links beim
21D6   ED 6F        RLD          ) Eintragen der Ziffern
21D8   C9           RET
```

Unterprogramm UA

```
21E0   06 03        LD B, 03     ) dreimaliges Umformen
21E2   D5           PUSH DE      ) der durch DE adressier-
21E3   21 10 23     LD HL, 2310  ) ten Speicherstellen
21E6   1A           LD A, (DE)   ) (Uhrzeit, AE und AA) in
21E7   CD D9 04 *   CALL TWOSEG  ) 7Segmentform und Ablegen
21EA   13           INC DE       ) im Anzeigebereich
21EB   10 F9        DJNZ F9
21ED   21 12 23     LD HL, 2312  ) Setzen der Dezimalpunkte
21F0   CB E6        SET 4, M     ) in 2312 und 2314 des
21F2   21 14 23     LD HL, 2314  ) Anzeigebereiches
21F5   CB E6        SET 4, M     )
21F7   DD 21 10 23  LD IX, 2310  IX vorbereiten für DAK 1
21FB   Dl           POP DE
21FC   C9           RET
```

Speicherorganisation

Anzeige-	2315	2314	2313	2312	2311	2310
bereich	h 10	h 1	min 10	min 1	sec 10	sec 1

Uhrzeit	2302	2301	2300
	h	min	00

Alarm	2305	2304	2303
ein	h	min	"AE"

Alarm	2308	2307	2306
aus	h	min	"AA"

Speicherung auf Kassette

FILE-Name: 24AA
Startadresse: 21D0
Endadresse: 2352

Die Programmauflistung enthält alle wesentlichen Erläuterungen, so daß eigentlich alles klar sein müßte.

12.3. Benutzung

Wenn alles aufgebaut, programmiert und kontrolliert wurde, starten wir das Programm, diesmal auf
Adresse 2200!
Das ist wichtig! Es erscheint dann:

```
F F.F F.A E
```

d. h. die Alarmeinschaltzeit kann über die zehn Zifferntasten eingegeben werden (Vornullen beachten!). Ist diese Zeit eingestellt, drücken wir

[+]

und es erscheint:

F F.F F.A A

also wird die Alarmabschaltzeit eingegeben. Danach wieder

[+]

und

0 0.0 0.0 0

zeigt an, daß die augenblickliche Uhrzeit eingegeben werden kann. Auch hier werden nur Stunden und Minuten eingetippt.
Mit [+] würden wir wieder AE erreichen und so fort.
Beim Betätigen von [EX] wird die Uhr gestartet, jetzt laufen auch die Sekunden mit.
Wird die Alarmeinschaltzeit A3 erreicht, geht A 0 auf High-Pegel und bleibt solange in diesem Zustand bis die Alarmausschaltzeit AA erreicht wird. A 0 geht dann auf Low-Pegel zurück.

Eine Besonderheit unserer Quarzuhr ist, daß wir bei Betätigung von [ST] in den Grundzustand des Rechners gelangen. Er zeigt wie gewohnt LC-80 an. Wir können jetzt Programme schreiben und auch laufen lassen, ohne daß die "innere Uhr" beeinflußt wird. Dabei müssen wir folgende "Spielregeln" einhalten:

- Niemals [RES] drücken!
  (Der LC-80 ist bereits im Grundzustand. Durch Betätigen von [ADR] ,[DAT] und allen Eingabetasten können wir auch ohne [RES] fast alles erreichen).

- Wieder zurück in das Uhrenprogrammgelangen wir mit NMI .
- Die Adressen oberhalb 21CF stehen nicht zur Verfügung.

12.4. Ausblicke

Im Gegensatz zu normalen Quarzuhren bietet unser LC-80 wieder einmal neue Möglichkeiten:

- Schon jetzt haben wir durch getrennte Programmierung von Einschalten und Ausschalten des Alarmes völlig neue "Gebrauchswerte", wir denken z. B. an das programmierte Mitschneiden von Hörfunksendungen in der Nacht usw.
- Aber auch als Timer von 1 Minute bis 24 Stunden läßt sich unsere Uhr verwenden.
- Durch veränderte Programmierung können weitere Alarmzeiten erreicht werden. Vom Port A stehen uns noch 7, vom Port B noch 3 Leitungen zur Verfügung ...
- Die "innere Uhr" des LC-80 kann aber auch benutzt werden, um den Zeitpunkt von irgendwelchen Ereignissen zu speichern oder Meßwerte mit der dazugehörigen Zeit auszugeben.

Es zeigt sich, daß eine Computer-Uhr doch mehr ist als die einfache Nachahmung vorhandener Digitaluhren. Im Gegensatz zu diesen kann sie sehr einfach unseren jeweiligen Bedürfnissen angepaßt werden.

Folgende Aufgabenstellungen könnten für uns reizvoll sein:

- ein Zähler mit Quarzzeitbasis zum wahlweisen Messen von Frequenzen direkt in Hz oder Drehzahlen in U/min
- Kombination von Digitalthermometer gemäß Kapitel 7 und Quarzuhr zur stündlichen Temperaturerfassung und -abspeicherung (automatische Wetterwarte)
- Quarzuhr mit Westminstergong oder Kuckucksuhr
- Stoppuhr mit Erfassung der Zeiten aller (!) Läufer.

Also ans Werk!

13. LC-80 und Oszillograph

Alle diejenigen, die mit einem Oszillographen experimentieren, sollten dieses Kapitel recht aufmerksam lesen - es lohnt sich!

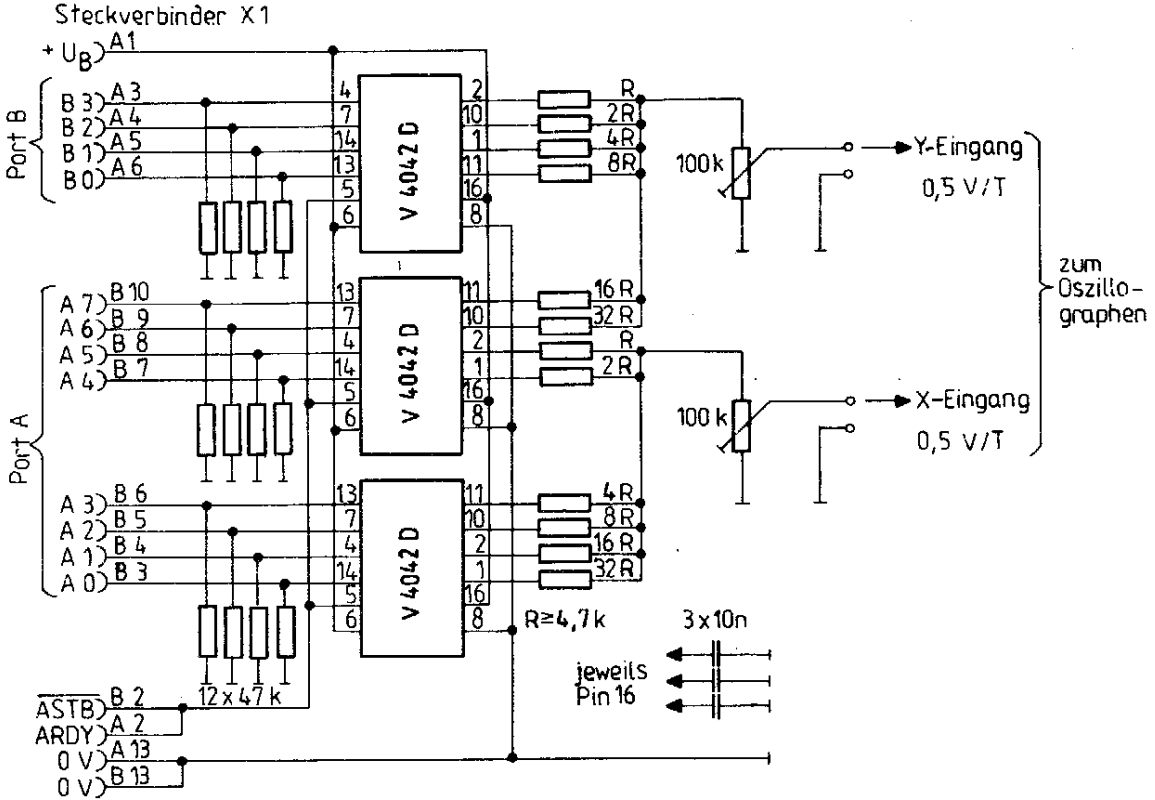
Unser LC-80 kann, wie wir im Kapitel 11 gesehen haben, sehr einfach digitale Informationen über die USER-PIO ausgeben. Genauso wie unser DVM aus Kapitel 7 Analogwerte in (rechnerverständliche) Digitalwerte umwandeln kann, geht das mit einer geeigneten Hardware auch umgekehrt. Im ersten Fall spricht man von Analog-Digital-Wandlern (A/D-Converter), im zweiten von Digital-Analog-Wandlern (D/A-Converter). Ein solches Gerät

wollen wir in diesem Kapitel bauen und uns mit den verblüffenden Eigenschaften einer Kopplung von Rechner, D/A-Wandler und Oszillograph vertraut machen.
Wir beginnen, wie schon gewohnt, mit dem Aufbau einer Zusatzleiterplatte.

13.1. D/A-Converter

Unsere Zusatzschaltung soll natürlich kein Präzisionsgerät werden. Deshalb beschränken wir uns auf die Umwandlung von 6 Bit. Das heißt, wir können eine Analogspannung in 64 Einzelwerte zerlegen.
Warum gerade 6 Bit?
Weil wir genau 12 Port-Leitungen zur Verfügung haben und damit zwei gleichartige D/A-Converter aufbauen können. So sind wir also in der Lage, gleichzeitig zwei unabhängig programmierbare Analogspannungen zu erzeugen.

Wie funktioniert nun unsere Zusatzschaltung?
Wir benutzen zur D/A-Wandlung ein Widerstandsnetzwerk, das von CMOS-Bausteinen des Typs V 4042 D des veb mikroelektronik "karl marx" erfurt - stammbetrieb angesteuert wird. Diese Bausteine sind 4Bit-Auffangregister, die den am Eingang anliegenden Pegel nach einem Übernahmetakt an den Ausgang weitergeben und auch speichern.
Drei solche V 4042 D liegen eingangsseitig an der USER-PIO. Die Quittierungssignale /ASTB, und ARDY sind miteinander verbunden und der damit erzeugte kurze Impuls beim Ausgeben von Daten am Port A wird als Übernahmeimpuls für die Auffangregister benutzt.
Die beiden Widerstandsnetzwerke sind vom Typ R, 2 R, 4 R, 8 R, 16 R und 32 R.
Die folgende Abbildung zeigt die Gesamtschaltung.



2fach D/A-Converter zu je 6 Bit

Hinweise zum Aufbau

- Es ist wichtig, daß die Eingänge der CMOS-Bausteine mit Widerständen gegen Masse auf definiertem Potential gehalten werden!
- Die Verdrahtung der einzelnen PIO-Leitungen und der Speicher-Flip-Flops der V 4042 D ist an sich beliebig. Wenn vom vorgeschlagenen Schema abgewichen wird, ist nur sicherzustellen, daß die Zuordnung der Widerstandswerte zu den jeweiligen PIO-Bits erhalten bleibt, sonst geht alles durcheinander.
- Über Masse (Pin 8) und + UB (Pin 16) der V 4042 D ist jeweils ein Kondensator von 10 nF zu legen!
- Masse und Betriebsspannung an den CMOS-ICs auf keinen Fall vergessen!
- Die Widerstände R, R 2,.., R 32 setzen wir am besten aus je zwei Einzelwerten (z. B. Reihenschaltung) zusammen: Beispiel:

  R = 4,7 kOhm
  2 R = 9,4 kOhm = 4,7 kOhm + 4,7 kOhm
  4 R = 18,8 kOhm = 15,0 kOhm + 3,8 kOhm usw.

  Dabei werden wir um etwas aussuchen und ausmessen nicht herumkommen. Die Werte sollten mit etwa 1% genau liegen, sonst gibt es Probleme.

13.2. Erprobung

Bevor wir uns an kompliziertere Dinge wagen, wollen wir die Funktion der beiden D/A-Converter überprüfen.

Am einfachsten geht das natürlich mit dem Oszillographen. Wir nehmen dazu etwas abgeschirmtes Kabel mit BNC-Stecker (oder zum Oszillographen passenden Stecker!), stecken das in die Y-Buchse des Oszillographen und schließen das andere Kabelende zunächst an den X-Ausgang des D/A-Converters (Masse nicht vergessen!) an. Wenn wir dann ein kleines Prüfprogramm eingeben, können wir leicht die Funktion der Anordnung überprüfen:

```
2000   3E 0F   LD A, 0F        }   PIO-Mode 0
2002   D3 FA   OUT FA          }   Byte-Ausgabe
2004   3E 00   LD A, 00        }   Ausgabe von 00
2006   D3 F8   OUT F8          }   auf Port A
2008   3C      INC A               A erhöhen
2009   18 FB                       und wieder ausgeben usw.
```

Wenn alles klappt und der Oszillograph auf 0,5 V/Teilstrich sowie 0,5 ms/Teilstrich eingestellt ist, müßte jetzt eine Sägezahnspannung zu sehen sein. Wenn nicht, könnte es an dem 100 kOhm - Regler am D/A-Converter-Ausgang liegen, der vielleicht am masseseitigen Anschlag liegt. Oder es ist etwas anderes nicht in Ordnung!

Jetzt klappt hoffentlich alles und wir lösen das Kabel oszillographenseitig und stecken es in den X-Eingang des Oszillographen. Gleich danach bauen wir noch solch ein Kabel, welches wir dann in den Y-Eingang des Oszillographen stecken und am Y-Ausgang des D/A-Converters anlöten.
Wenn jetzt beides in den richtigen Buchsen steckt, geben wir wieder ein kleines Prüfprogramm ein:

```
2000   3E 0F          LD A, 0F
2002   D3 FA          OUT FA
2004   D3 FB          OUT FB
```

```
2006    21 FF FF       LD HL, FFFF
2009    06 00          LD B, 00
200B    23             INC HL
200C    7E             LD A, M
200D    D3 F9          OUT F9
200F    23             INC HL
2010    7E             LD A, M
2011    D3 F8          OUT F8
2013    10 F6          DJNZ F6
2015    18 EF          JR EF
```

Wir schalten den Oszillographen in X-Betrieb und Y auf 0,5 V/ Teilstrich. Sollte alles funktionieren, sehen wir jetzt einen künstlichen Sternenhimmel ...
Auch nicht schlecht, oder?
Was wir da sehen, sind Teile des LC-80 - Betriebssystems aus dem ROM, die wir willkürlich in Analogsignale umwandeln und dabei jeweils X- und Y-Koordinaten erzeugen. Das bildet dann eine flächenhafte Darstellung, die wir durch Verstellen der beiden Regler des D/A-Converters auf ein Quadrat formen können.

13.3. "Sticken" auf elektronisch

Wir wollen das ungeordnete Durcheinander auf unserem Bildschirm in eine für uns sinnvolle Form bringen und z. B. einen Text darstellen.
Dazu müssen wir wie beim Sticken Punkt für Punkt programmieren. Das ist eine enorme Fleißaufgabe. Wer keine Geduld hat, hört bei der Textprogrammierung einfach bei der Adresse 20D9 auf.

Das Textbeispiel beginnt auf Adresse 2040. Dort steht die Y-Koordinate für den ersten Punkt. Danach folgt dessen X-Koordinate auf Adresse 2041, auf Adresse 2042 die Y-Koordinate des 2. Punktes usw. Die beiden Koordinaten bewegen sich im Bereich von 00 ... 3F, also genau 6 Bit.

Los geht's:

```
2040    36 0A          Y-Koordinate 1.Punkt, x-Koordinate 1.Punkt
2042    35 0A          Y-Koordinate 2.Punkt, X-Koordinate 2.Punkt
        34 0A          usw.
        33 0A
        32 0A
        31 0A
        30 0A
        33 0B
        33 0C
        33 0D
        36 0E
        35 0E
        34 0E
        33 0E
        32 0E
        31 0E
        30 0E
2062    30 12
        31 12
        32 12
        33 12
        34 12
        35 13
        36 14
        35 15
        34 16
        33 13
        33 14
```

	33 15		30 2B		22 2C
	33 16		30 2C		23 2C
	32 16		30 2D		24 2C
	31 16		31 2E		25 2C
	30 16		32 2E		26 2D
2082	36 1A		33 2E		26 2C
	35 1A		34 2E		26 2B
	34 1A		35 2E	2112	26 22
	33 1A	20CE	36 34		26 23
	32 1A		35 34		26 24
	31 1A		34 34		26 25
	30 1A		33 34		25 26
	30 1B		32 34		24 26
	30 1C		30 34		23 25
	30 1D	20DA	26 36		23 24
	30 1E		25 36		23 23
2098	36 22		24 36		23 22
	35 22		23 36		22 22
	34 22		22 36		21 22
	33 22		21 36		20 22
	32 22		20 36		20 23
	31 22		22 35		20 24
	30 22		23 34		20 25
	30 23		24 33		21 26
	30 24		26 32		22 26
	30 25		25 32		24 22
	30 26		24 32		25 22
20AE	36 2D		23 32	213A	26 16
	36 2C		22 32		25 16
	36 2B		21 32		24 16
	35 2A		20 32		23 16
	34 2A	2020	20 2D		22 16
	33 2A		20 2C		21 16
	32 2A		20 2B		20 16
	31 2A		21 2C		23 15

	23 14		14 03		13 25
	23 13		14 05		14 26
	20 12		15 05		15 26
	21 12		16 05		16 25
	22 12	2198	16 0A		16 24
	23 12		15 0A		16 23
	24 12		14 0A		15 22
	25 12		13 0A		14 22
	26 12		12 0A		12 22
215C	25 0E		11 0A		11 22
	26 0D		10 0A		10 23
	26 0C		10 0B		10 24
	26 0B		10 0C		10 25
	25 0A		10 0D		11 26
	24 0A		10 0E		12 26
	23 0A	21AE	11 16	21F4	10 2D
	22 0A		10 15		10 2C
	21 0A		10 14		10 2B
	20 0B		10 13		11 2A
	20 0C		11 12		12 2A
	20 0D		12 12		13 2A
	21 0E		13 12		14 2A
2176	26 05		14 12		15 2A
	26 04		15 12		16 2B
	26 03		16 13		16 2C
	25 04		16 14		16 2D
	24 04		16 15		15 2E
	23 04		15 16		14 2E
	22 04	2108	13 1A		13 2E
	21 04		13 1B		12 2E
	20 05		13 1C		11 2E
	20 04		13 1D	2214	14 33
	20 03		13 1E		15 33
218C	16 03	21D2	13 23		16 33
	15 03		13 24		16 35

```
       15 35                02 26                01 0E
       14 35                01 26                00 0E
2220   05 3A                00 25                02 0D
       06 3B                00 24                03 0C
       06 3C                00 23                04 0B
       06 3D                00 22                06 0A
       05 3E                01 22                05 0A
       04 3E                02 22                04 0A
       03 3D                03 22                03 0A
       03 3C                04 22                02 0A
       02 3C                05 22                01 0A
       00 3C          2276  06 12                00 0A
2234   06 2E                06 13          22BC  06 06
       05 2E                06 14                05 06
       04 2E                06 15                04 06
       03 2E                05 16                03 06
       02 2E                04 16                02 06
       01 2E                03 16                01 06
       00 2D                02 16                00 05
       00 2C                01 16                00 04
       00 2B                00 15                00 03
       01 2A                00 14                01 02
       02 2A                00 13                02 02
       03 2A                00 12                03 02
       04 2A                01 12                04 02
       05 2A                02 12                05 02
       06 2A                03 12                06 02
2252   06 22                04 12
       06 23                05 12
       06 24          229A  06 0E
       06 25                05 0E
       05 26                04 0E
       04 26                03 0E
       03 26                02 0E
```

Jetzt haben wir uns eine Pause verdient!

Es ist eigentlich erstaunlich, was in unseren RAM-Speicher hineinpaßt, nicht wahr?
Wir haben genau 333 Punkte mit je zwei Koordinaten zu je zwei Zeichen eingegeben, das waren mit der [+]-Taste insgesamt 1998 Tastenbetätigungen!!!

Jetzt wollen wir aber wissen, was wir da programmiert haben - aber es hilft nichts, wir müssen noch das Anzeigeprogramm laden. Das steht im folgenden Abschnitt. Aber Mut - es ist diesmal kurz.

13.4. Programm für Textdarstellung

```
2000   3E 0F          LD A, 0F
2002   D3 FA          OUT FA          PIO-Mode 0, Port A
2004   D3 FB          OUT FB          PIO-Mode 0, Port B
2006   21 40 20       LD HL, 2040     Anfangsadresse d. Textes
2009   7E             LD A, M         Y-Koordinate 1.Punkt
200A   FE FF          CP FF           Vergleich, ob FF
200c   28 F8          JRZ F8          Wenn ja, wieder zum
                                      Textanfang
200E   CB 0F          RRCA       }    Inhalt von A (Y-Koordi-
2010   CB 0F          RRCA       }    nate 2x nach rechts
                                      verschieben
2012   D3 F9          OUT F9          Ausgabe auf Port B
2014   E6 C0          AND C0          Bits 0 . 5 von A
                                      werden = 0 gesetzt
2016   23             INC HL          Adresse der X-Koordinate
                                      1.Punkt
2017   86             ADD M           Bit 0 ... 5 aus M, 6,7 a.
2018   D3 F8          OUT F8          Ausgabe auf Port A
201A   23             INC HL          Y-Koordinate 2.Punkt
201B   18 EC          JR EC
```

Dieses Programm realisiert die punktweise Darstellung beliebiger Zeichen auf dem Oszillographenschirm. Der Text beginnt auf Adresse 2040 mit der Koordinate des ersten Punktes, danach folgt dessen X-Koordinate usw. Die erste Y-Koordinate mit dem Inhalt FF führt zum Abbruch und erneuten Durchlauf des Textes bis dorthin. Nach Initialisierung der PIO wird die erste Y-Koordinate in A geladen und geprüft, ob sie FF ist. Wenn nicht, wird die aus 6 Bit bestehende Koordinate um zwei Bitpositionen in Richtung niederwertige Bits verschoben. Danach befinden sich die 4 oberen Bits der ursprünglichen Koordinate auf den Bits 0 ... 3 und die zwei unteren Bits auf den Bits 6 und 7:

0	1	2	3	4	5	6	7
A	B	C	D	E	F	0	0

ursprüngliche Anordnung der Y-Koordinate

6 Bit

Nach zweimaliger Verschiebung sieht das ganze so aus:

0	1	2	3	4	5	6	7
C	D	E	F	0	0	A	B

Ausgabe auf Port B

Die Bits 0 ... 3 werden auf Port B ausgegeben (der ja leider nur 4 Bit nach außen geben kann). Das ist der Grund für alle komplizierten Umformungen.
Der Befehl AND C0 setzt jetzt alle Bits außer 6 und 7 = 0:

0	1	2	3	4	5	6	7
0	0	0	0	0	0	A	B

Als nächstes wird HL inkrementiert und enthält damit die Adresse, auf der die X-Koordinate gespeichert ist. Diese hat folgenden Aufbau:

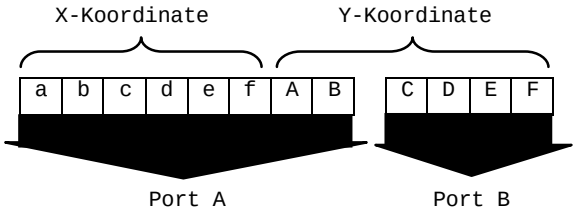
0	1	2	3	4	5	6	7
a	b	c	d	e	f	0	0

ursprünglicher Aufbau der X-Koordinate

Der Befehl ADD M bewirkt die Addition des Akkumulators (mit A und B auf Bit 6 und 7, alles andere = 0) und dem Inhalt der durch HL adressierten Speicherzelle (mit der kompletten X-Koordinate). Das im Akkumulator abgelegte Ergebnis sieht dann so aus:

0	1	2	3	4	5	6	7
a	b	c	d	e	f	A	B

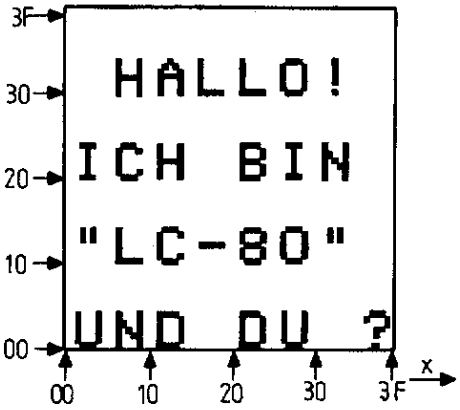
Genau das wird auf Port A ausgegeben. Somit erhalten wir nach der Ausgabe beider Ports:



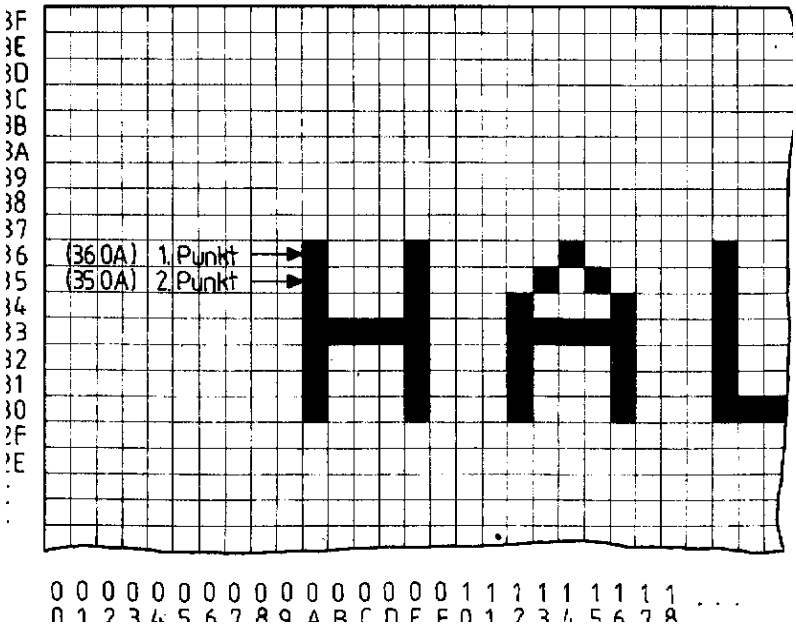
zum 2 x 6Bit - D/A - Wandler

Nach erneuter Inkrementierung von HL wird der nächste Punkt mit Y- und X-Koordinate abgearbeitet.

Wenn alles geklappt hat, müßte auf dem Oszillographenschirm jetzt folgendes erscheinen:



"Vergrößert" sieht es dann so aus:



Wer Lust hat, kann dem Rechner jetzt antworten und seinen Namen programmieren. Wie das geht, dürfte nun klar sein:

- Als erstes fertigen wir eine Skizze des Anzeigefeldes 00 bis 3F im Quadrat an (Rechenpapier).
- Danach werden die gewünschten Punkte angekreuzt. Das muß nicht unbedingt Text sein - alles ist erlaubt. Zum Beispiel geht es auch mit kleinen Zeichnungen oder Handschrift!
- Nun werden die Koordinaten aller Punkte ab Adresse 2040 eingetragen, und zwar immer zuerst die Y-Koordinate und dann die X-Koordinate!
- Die erste Speicherstelle mit FF (Y-Koordinate) bezeichnet das Ende der Punktkette.

Alles übrige erledigt der LC-80!

Natürlich ist unsere Art der Textdarstellung mit dem Oszillographen eine "Knochenarbeit" für den Programmierer - aber sehr einfach hardwareseitig zu realisieren. Für eine Darstellung auf dem Fernsehschirm wäre der Hardwareaufwand wesentlich größer.
Und noch einen Vorteil haben wir. Unser. System ist "voll grafikfähig", d: h. wir können jeden Punkt einzeln programmieren. Das ist für verschiedene Anwendungen sehr praktisch. Es lassen sich Diagramme, beliebige Texte, bewegliche Darstellungen, Kurvenverläufe und vieles andere realisieren.

Aber auch ohne Oszillographen kann man einiges mit dem vorgestellten D/A-Converter anfangen, z. B. Schwingungen mit beliebiger Kurvenform erzeugen. Unser "Sägezahn" am Anfang des Kapitels war ein einfaches Beispiel. Es war eine Sägezahnschwingung mit langsam ansteigender Spannung. Durch die Änderung eines Befehls geht es genau anders herum. Schreiben wir auf Adresse

```
2008   3D            DEC A,
```

ist es schon passiert.

Zu beachten ist lediglich, daß unser D/A-Converter recht hochohmig ist. Wenn also größere Lasten als etwa 100 kOhm getrieben werden sollen, muß ein Spannungsfolger mit Operationsverstärker o. ä. nachgeschaltet werden.
Dann wird z. B. auch eine digital programmierbare Spannungsquelle möglich.

Aber auch ein anderes Format für den D/A-Converter ist denkbar, so z. B. ein 8Bit-Wandler nur für Port A. Das vereinfacht die Ausgabe erheblich, weil die umständliche "Sortierung" auf 2 x 6 Bit entfällt. Wenn man mit 4 Bit für die Y-Koordinate auskommt, kann z. B. eine. einzeilige Textdarstellung, dann aber mit einer X-Auflösung von 256 Punkten, erfolgen.
Auch eine wunderschöne Laufschrift als Geburtstagsgratulation läßt sich damit programmieren!

So, das waren genug Anregungen für Spielereien auf dem Oszillographen.

14. Schlußkapitel

Wir haben in den vorangegangenen Kapiteln die Bedienung und einige nützliche Anwendungsgebiete unseres LC-80 kennengelernt. Ganz bewußt waren diese Anwendungsbeispiele so ausgewählt, daß die Stärke unseres Kleinstcomputers auf steuerungstechnischem Gebiet herausgearbeitet werden konnte. So können wir auch abschließend die Grenzen des LC-80 zu den

"richtigen" Heimcomputern ziehen, mit denen der Laie ihn oft vergleichen wird. Dieser Vergleich ist wie der zwischen Äpfeln und Birnen, beide haben Vor- und Nachteile, die wir im folgenden einmal diskutieren und bewerten wollen.

Was ist ein Heimcomputer?
Es handelt sich um einen Kleinstrechner, der ähnlich wie seine "großen Brüder" Daten verarbeitet. Die Ein- und Ausgabe erfolgt über alphanumerische Einheiten (Schreibmaschinentastatur, Bildschirm). Im wesentlichen muß der Nutzer nicht seinen inneren Aufbau (Hardware) und seine Befehlsabläufe kennen, um ihn zu bedienen. Eine Benutzung der maschinenorientierten Befehlsabläufe ist meist nur über Umwege möglich.

Was ist LC-80?
Auch ein Kleinstrechner. Aber einer, der so bedient und behandelt werden muß, wie es die innere Rechnerstruktur verlangt - in Maschinensprache. Also ein Nachteil? Ja und nein. Für den, der einfach rechnen und logische Operationen durchführen lassen will, ist der LC-80 etwas umständlich. Seine Kommunikationsmöglichkeiten sind begrenzt.
Wer sich aber für die inneren Abläufe interessiert dort gegebenenfalls Untersuchungen und Messungen durchführen will, ist mit dem LC-80 bestens bedient. Ein richtiger Lerncomputer - wobei das Kennenlernen der Arbeitsweise eines Mikrorechners gemeint ist.
Auch in einer anderen Richtung ist der LC-80 nützlich. Es ist zu erwarten, daß immer mehr Aufgaben auf allen Gebieten elektronisch gelöst werden. Bisher mußte für jedes Problem eine speziell dafür geeignete Schaltung entwickelt und produziert werden. Die Mikroprozessortechnik führt zu einer Verlagerung

von der Schaltungs- zur Softwareentwicklung. Das bedeutet, dieselben Schaltkreise (und oft auch dieselbe Leiterplatte) lösen völlig unterschiedliche Probleme - je nach Programm. Damit wird der LC-80 zu einem kleinen Mikrorechner-Entwicklungssystem, mit dessen Hilfe singuläre Lösungen auf Mikroprozessorbasis getestet werden können.

Im 2. Teil unseres Buches werden wir diesen Weg gehen und mit Hilfe des LC-80 eigene Funktionseinheiten auf Mikroprozessorbasis entwickeln und testen.

Was erwartet uns im 2. Teil des Buches noch?

Nun, z. B. ein EPROM-Programmierboard, mit dem wir EPROMs vom Typ U 2716 C testen und programmieren können.

Oder etwas über die häufig vernachlässigten "Bindeglieder" eines Mikrorechners mit der Umwelt - Sensoren und Stellglieder bzw. deren Realisierung durch Amateure.

Außerdem weitere Anwendungsbeispiele wie immer in gewohnt aufgelockerter Form. Aber - es wird komplizierter!

Die Beispiele verlangen sowohl einiges an "Bastlerfähigkeiten" als auch Systemdenken bei der Lösung digitaler Probleme mit Mikrorechnern.

Glücklicherweise ist zu erwarten, daß Käufer und Benutzer des LC-80 diese Eigenschaften besitzen werden. Diese "Bastler von heute" werden sich auch mit dem 2. Teil des Buches anfreunden und viele Anregungen für den privaten und evtl. auch den beruflichen Bereich finden.

Literaturverzeichnis

[1] Technische Beschreibung
Zentrale Verarbeitungseinheit CPU U 880 D
veb mikroelektronik "karl marx" erfurt – stammbetrieb

[2] Technische Beschreibung
Schaltkreis für parallele Ein- und Ausgabe PIO U 855 D
veb mikroelektronik "karl marx" erfurt – stammbetrieb

[3] Technische Beschreibung
Schaltkreis für Zähler- und Zeitgeberfunktion CTC U 857 D
veb mikroelektronik "karl marx" erfurt – stammbetrieb

[4] Befehlsbeschreibung U 880 D
veb mikroelektronik "karl marx" erfurt – stammbetrieb

[5] Bedienungsanleitung Lerncomputer LC-80
1. Ausgabe November 1984
veb mikroelektronik "karl marx" erfurt – stammbetrieb

[6] Hertzsch, A.: CMOS-Logikschaltkreise,
Band 212 der Reihe "elektronika"
Militärverlag der DDR, Berlin 1983

[7] Barthold/Bäurich: Mikroprozessoren - mikroelektronische
Schaltkreise und ihr Anwendung (Teile 1 bis 3)
Bände 186, 188 und 203 der Reihe "elektronika"
Militärverlag der DDR, Berlin 1980

[8] Kieser, H./Meder, M.: Mikroprozessortechnik - Aufbau und
Anwendung des Mikroprozessorsystems U 880
Verlag Technik, Berlin 1982

[9] Schwarz,W./Meyer, G./Eckhardt, D.:
Mikrorechner - Wirkungsweise, Programmierung, Applikation
3. Auflage
Verlag Technik, Berlin 1984

[10] Polycomputer 880 - Dokumentation
- Bedienhandbuch
- Systemhandbuch
- Arbeitsbuch, Teil I
VEB Kombinat Polytechnik und Präzisionsgeräte
Karl-Marx-Stadt

[11] Handbuch LC-80
veb mikroelektronik "karl marx" erfurt – stammbetrieb

RFT

veb mikroelektronik
›karl marx‹ erfurt
stammbetrieb

DDR-5023 Erfurt, Rudolfstraße 47
Telefon: 5 80, Telex: 061306

elektronik
export·import
Volkseigener Außenhandelsbetrieb der
Deutschen Demokratischen Republik
DDR - 1026 Berlin, Alexanderplatz 6
Telex: BLN 114721 elei, Telefon: 2180