

MICRO-PROFESSOR

MPF-I Handbuch



Copyright © 1981 by Multitech Industrial Corp.
No part of this publication may be reproduced,
stored in a retrieval system,
or transmitted,
in any form or by any means,
electronic, mechanical, photocopying,
recording, or otherwise,
without the prior written permission of the publisher.



MULTITECH INDUSTRIAL CORPORATION

OFFICE 977 MIN SHEN E ROAD TAIPEI 105 TAIWAN, R.O.C.
TEL (02) 769 1225 (10 LINES) TELEX 23756. MULTIIC, 19162
FACTORY 5, TECHNOLOGY ROAD III
HSINCHU SCIENCE BASED INDUSTRIAL PARK
HSINCHU TAIWAN, 300 R.O.C.
TEL (035) 775102 (3 LINES)

1

12

13

14

15

16

17

18

19

20

This manual is translated and printed
with the permission of Multitech Industrial Corp.

Copyright © 1982 MSB-Verlag
Deutsche Auflage 54321



Mangoldstraße 10
7778 MARKDORF
Telefon (07544) 3058
Telex 734 628 msb d

ISBN 3-922187-01-3

ersetzung des ersten Teils
freundlicher Unterstützung durch:

rschinger Mess-Technik GmbH
tfach 11 55 · D-8036 Herrsching a. Ammersee
1982

zweite Teil des MPF I Handbuches
I mit dem Gerät im Original geliefert

ersetzung des dritten Teils:
dbuch für den MPF I

tsche Bearbeitung: Johannes Blumenrath

die freundliche Unterstützung bedanke ich mich bei
rn Anton Kaldenbach.

ses Handbuch entstand in Anlehnung an das englische Handbuch als Arbeitsunterlage
den Kursus „Aufbau und Anwendung von Mikroprozessoren“ bei der INFO-Neuß
titut für berufliche Fortbildung GmbH).

MICROPROFESSOR

Handbuch für den
MPF - I

1. Teil: Bedienungsanleitung

den ersten 50 Jahren unseres 20. Jahrhunderts wurden die Verbrennungsmaschinen erfunden. Diese vergrößerten die körperlichen Kräfte des Menschen.

In der zweiten Hälfte des 20. Jahrhunderts wird der Microprocessor unsere geistigen Fähigkeiten erweitern.

Die Anwendungen dieses erstaunlichen Produkts in den unterschiedlichsten Industrien werden einen starken Einfluß auf unser aller Leben zu nehmen, daher sprechen wir von der zweiten industriellen Revolution.

1. TEIL: MPF - I BEDIENTUNGSANLEITUNG

INHALTSVERZEICHNIS

MPF-I Spezifikationen	I - 1
1.1 Hardware Spezifikationen	I - 1
1.2 Software Spezifikationen	I - 3
1.3 Mechanischer Aufbau	I - 4
Allgemeine Beschreibung	I - 5
2.1 Funktion des Monitor Programmes	I - 5
2.2 Begriffsdefinition	I - 6
2.3 Fehler Anzeige	I - 8
2.4 RAM Adressierung	I - 9
Bedienung	I - 10
3.1 Einführung	I - 10
3.1.1 System Reset — RS Taste	I - 11
3.1.2 Verändern des Speichers — ADDR und DATA Taste	I - 11
3.1.3 Prüfen und Ändern der Register — REG und DATA Taste	I - 14
3.1.4 Programm Zähler — PC Taste	I - 17
3.2 Programm Ausführung	I - 18
3.2.1 Programm Start — GO Taste	I - 18
3.2.2 Einzelschritt — STEP Taste	I - 19
3.2.3 Unterbrechung — SRH Taste	I - 20
3.2.4 Löschen der Unterbrechung — CBR Taste	I - 23
3.2.5 Sofortiger Stop — MON Taste	I - 24
3.2 Übertragungsfunktionen	I - 25
3.3.1 Block Transfer — MOVE Taste	I - 25
3.3.2 Daten löschen — DEL Taste	I - 27
3.3.3 Daten einfügen — INS Taste	I - 28
3.3.4 Relative Adresse — RELA Taste	I - 29
3.3.5 Speichern auf Band — TAPK WR Taste	I - 30
3.3.6 Wiedergabe vom Band — TAPK RD Taste	I - 32

4. Software und Hardware Beschreibung	34
4.1 Speicher Adresse	34
4.2 Adresse Eingabe/Ausgabe	36
4.3 Programm Unterbrechung	39
4.4 Software Stop — Instruktion RST 30 H (Opcode F7)	40
4.5 Stapelblock	41
4.6 Reset	42
4.7 Bandaufzeichnungs-Format	43
5. Monitor Unterroutinen	44
5.1 Überblick	44
5.2 SCAN1	45
5.3 SCAN	46
5.4 HEX7	47
5.5 HEX7SG	48
5.6 RAMCHK	49
5.7 TONE	50
5.8 TONE1K	51
5.9 TONE2K	52
5.10 Programm Beispiele	53
6. Überprüfen der Speicher	58
6.1 EPROM 0000 — 07FF	58
6.2 RAM Region 1800 — 1FFF	59
ANHANG	
A. Display Format, Positions Code und interner Code	A-1
B. Theorie der Schaltkreise	B-1
C. Z80-CPU Programme, Befehle	C-1
D. Referenz Nachweis	D-1

2. TEIL: SOURCE LISTING MONITOR PROGRAM

(Wird mit dem Gerät im Original mitgeliefert)

Einführung in die Programmierung des Z-80

	Seite
Microcomputerprogrammierung	III - 1
Schreibung der Befehlsgruppen des Z-80	III - 2
Programmierung 1 – Ladebefehle	
Bit-Ladebefehle	III - 5
Programmierung 1 A	
Bit-Ladebefehle	III - 6
Programmierung 2	
Zweigte Programme – Programmschleifen	III - 10
Flowchart. (engl.)	III - 12
Flußdiagramm (deutsch)	III - 13
Programmbeispiele	
Arithmetische Befehle	III - 17
Programmverzweigungen (Loops)	III - 21
Verschachtelte Verzweigungen, Zeitschleife	III - 23
Transfer- und Suchbefehle	III - 25
Stack und Unterprogramm- Technik	III - 27
Unterprogramm für Addition	III - 31
Rotations- und Schiebebefehle	III - 33
Binäre Multiplikation	III - 35
Binäre Division	III - 39
Umwandlung binärer in BCD-Zahlen	III - 43
Umrechnung von BCD-Zahlen in Binärzahlen	III - 47
Berechnung der Quadratwurzel	III - 51
Anzeige der Anzeige	III - 56
Anzeigeeinheit des MPF-1	III - 57
Monitor-Unterprogramme für Tastatur und Anzeige	III - 61
Programmierung der Anzeige „HILFE!“	III - 63
Demonstrationsprogramm „Lauffeuer“	III - 67
Stoppuhr für hundertstel Sekunden	III - 70
Uhrenprogramm	III - 73
Uhr mit dem CTC-Baustein, interruptgesteuert	III - 77
Programmbeispiele zur Tonausgabe:	
Telefon-Ton	III - 82
Martinshorn	III - 84
Microcomputer-Orgel	III - 85
Musikbox	III - 89

MPF-I Spezifikationen

Hardware Spezifikationen

- (1) CPU: (zentraler Rechner)
Zilog Z-80 CPU mit 158 Befehlen und 2.5 MHz maximaler Clock Rate.
Für MPF-I ist die Arbeitsfrequenz 1.79 MHz.
- (2) ROM: (Read Only Memory)
Ein +5V EPROM 2516 (2532) x1, total 2K(4K) Bytes.
Monitor EPROM Adresse: 0000-07FF(0FFF).
- (3) RAM: (Readom Access Memory)
Statisches RAM: 6116, total 2K Bytes.
RAM Adresse: 1800-1FFF.
- (4) Speichererweiterung:
Ein +5V EPROM 2516/2716/2532/2732 EPROM oder 6116 statisches RAM
Erweiterungs Adresse: 2000-2FFF.
- (5) I/O Port:
Programmierbarer I/O Port 8255x1, total 24 parallele I/O Linien
I/O Adresse: 00-03.
Programmierbare PIO, total 16 parallele I/O Linien,
I/O Adresse: 80-83H
Programmierbarer CTC, total 4 unabhängige Zeitablaufsteuerungen
I/O Adresse: 40-43H
- (6) Anzeige:
6 Digit 0.5'' 7-Segment rote LED Anzeige.
- (7) Tastatur:
36 Tasten incl. 19 Funktions-Tasten, 16 hex-decimale Tasten und
1 frei programmierbare Taste.
- (8) Lautsprecher:
eingebaut, Durchmesser 57 mm
- (9) Erweiterungsbereich
Auf der Leiterplatte sind 8,9 x 3,5 cm Fläche zur beliebigen Erweiterung
vorgesehen.

(10) Tonband Interface:

Anschlußmöglichkeit für beliebige Tonbandgeräte. Die Daten-Rate ist 165 bps.

(11) System-Frequenz:

3.58 MHz Quarz, 1:2 Teiler, Pulsdauer 0.56 μ sec.

(12) Spannungsversorgung:

5 V, 500 mA.

(13) Netzanschluß:

Adapter 220 V 50 Hz: 9 V/500 mA.

(14) Abmessungen:

Höhe: 1,60 cm (Nur Leiterplatte)

Breite: 15,75 cm (Nur Leiterplatte)

Länge: 22,30 cm (Nur Leiterplatte)

Gewicht: 1,5 kg incl. Verpackung, Netzteil, Handbuch, Platine

Software Spezifikationen

MPF-I enthält ein 2K-Byte Monitor-Programm. Nach Einschalten beginnt der Monitor sofort zu arbeiten und überprüft alle Speicher.

Tastenfunktionen:

- (1)

RS

 : Neustart
- (2)

ADDR

 : Speicheradresse
- (3)

REG

 : Register Name
- (4)

DATA

 : Dateneingabe
- (5)

PC

 : Programm Zähler
- (6)

+

 : Speicher Adresse oder Register um 1 erhöhen
- (7)

-

 : Speicher Adresse oder Register um 1 verringern
- (8)

STEP

 : stufenweise Programmdurchführung
- (9)

SBR

 : Eingabe eines Unterbrechungspunktes im Programm
- (10)

CBR

 : Löschen eines Unterbrechungspunktes im Programm
- (11)

MON

 : sofortiger Programmabbruch
- (12)

GO

 : Programm- oder Monitor-Start
- (13)

INS

 : Einfügen von 1 Byte in den Speicher
- (14)

DEL

 : Herausnehmen von 1 Byte aus dem Speicher
- (15)

MOVE

 : Datenblock Verschiebung
- (16)

RELA

 : Kalkulation von relativen Adressen
- (17)

TAPE WR

 : Speicherung von Daten auf Band vom Speicher
- (18)

TAPE RD

 : Überspielen von Daten vom Band auf Speicher
- (19)

INTR

 : Unterbrechen: verbunden mit INT-Anschluß des CPU
- (20)

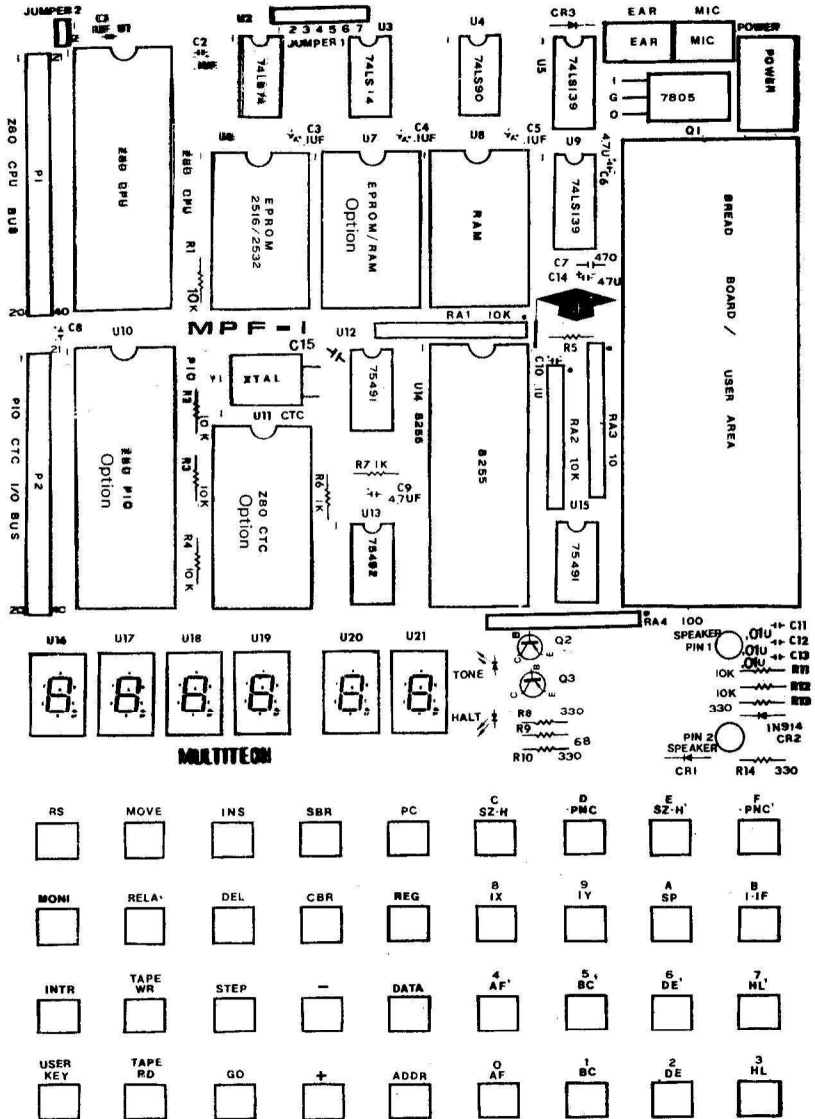
USER KEY

 : frei programmierbare Taste, verbunden mit Port 00, Bit 6.
- (21) * :

AF	BC	DE	HL	AF	BC	DE	HL	IX	IV	SP	I-IP	SE-H	-PNC	BZ-H	-PNC
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

: Hexadezimale Zahl oder Register Name

1.3 Mechanischer Aufbau: (M ~ 1 : 1)



MP 8121

Allgemeine Beschreibung

Funktion des Monitor Programmes

Im Monitor Programm sind bereits alle Service-Routinen enthalten, um den Benützern ein einfaches Arbeiten zu ermöglichen:

- (1) Einspeichern von Programmen in das RAM, anschließendes Überprüfen und/oder Ändern
- (2) Programme durchführen
- (3) Einzelschritte oder Unterbrechungsfunktionen erlauben eine stufenweise Programmausführung, wobei es möglich ist zwischendurch Daten oder Register abzuändern. Diese Möglichkeit ist in der Programmentwicklung sehr nützlich.
- (4) Ausrechnen von relativen Adressen, Tonbandanschluß, Arbeiten mit RAM-Speicher.
- (5) MPF-I ist sehr flexibel — Hardware und Software können erweitert werden. Der Benutzer kann eigene Programmvorstellungen verwirklichen.

2.2 Begriffsdefinition

(1) Hexadecimales Zahlen-System und sieben-Segment LED Anzeige:

hexadecimale Zahl	decimale Zahl	binär Zahl	7-Segment Anzeige
0	0	0000	0
1	1	0001	1
2	2	0010	2
3	3	0011	3
4	4	0100	4
5	5	0101	5
6	6	0110	6
7	7	0111	7
8	8	1000	8
9	9	1001	9
A	10	1010	A
B	11	1011	b
C	12	1100	C
D	13	1101	d
E	14	1110	E
F	15	1111	F

Bild 2-1 Zahlenzuordnung

- (2) Das Symbol für die 6-stellige Anzeige ist in Bild 2-2 gezeigt:

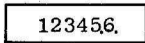


Bild 2-2 Anzeige Symbol

- (3) Wenn der Inhalt der Anzeige unbekannt ist oder keine Rolle spielt, wird die Zahlenposition mit "X" gekennzeichnet.
- (4) Ein Quadrat wird verwendet für die Tasten:



Bild 2-3 Symbole für Tasten

- (5) <address> wird für Speicheradresse verwendet; diese besteht aus den vom Anwender gewählten 4 hexadezimalen Zahlen. Bei Eingabe von mehr als 4 Zahlen werden die letzten 4 Ziffern weiterverwendet. Bei Eingabe von weniger als 4 Zahlen werden die ersten fehlenden Ziffern als 0 angenommen.
- (6) <data> steht für 1 Byte von Daten; dieses Byte besteht aus 2 vom Anwender gewählten hexadezimalen Ziffern. Eingaberegeln wie <address>.
- (7) Bei Kennzeichnung mit '[]', z.B. [{<address>}], wird diese Zahlenfolge nicht eingegeben.

2.3 Fehler Anzeige

Durch das Betätigen einer nicht zulässigen Taste wird die Anzeige gelöscht, solange die Taste gedrückt bleibt: Überprüfen und richtige Taste betätigen.

Einige Programmierfehler bewirken die Anzeige

-Err

 : Eingabe wiederholen und richtige Eingabefolge erarbeiten.

RAM Adressierung:

Die Adressen im RAM der Grundversion lauten 1800-1FFF,
im RAM für die Erweiterung 2000-27FF.

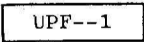
Die Adressen 1F9F-1FF3 im 1. RAM werden vom Monitor mitbelegt.
Bitte Kapitel 4 lesen.

3. Bedienung

Einführung, Programm-Ausführung und Übertragungsfunktionen werden hier ausführlich erklärt.

3.1 Einführung

3.1.1 System Reset Taste —

Betätigen der Reset Taste bewirkt sofortige Anzeige 

Nach dem Einschalten erscheint eine Ziffer nach der anderen und bewegt sich gleichzeitig nach links. In beiden Fällen läuft das Monitor-Überprüfungsprogramm sehr schnell ab.

2 Verändern des Speichers: **ADDR** und **DATA** Tasten

ADDR [address] **DATA** [(data)] **+** [(data)] **+**

Beispiel: überprüfe Inhalt von 0000-0003

Taste

Anzeige

Bemerkung

ADDR

x.x.x.x x

4 Indexpunkte zeigen an, daß die Adresse verändert werden kann.

AF
0

0.0.0.0 6

Der Inhalt von Adresse 0 ist 6.

= 0 . 0 . 0 0 - 0 0
[Address Bytes] [Daten Bytes]

siehe I - 12

+

0 0 0 1 0.0.

+

Taste erhöht den Addr. Zähler um eins.

+

0 0 0 2 1.0.

+

0 0 0 3 F.E.

ADDR 0003 ist FE.

Beispiel: ändere Inhalt von 1800 in AB, 1801 in DC

Taste

Anzeige

Bemerkung

ADDR

x.x.x.x x

[ADDR Feld]

4 Indexpunkte zeigen an, daß die Adresse verändert werden kann.

BC 1 **JK** 8 **AF** 0 **AF** 0

1.8.0.0 x

gebe 1800 durch Betätigen der entsprechenden Tasten ein.

DATA

1 8 0 0 x.x.

[DATA Feld]

Betätigen von DATA ermöglicht Eingabe von Datenbytes.



18003A.

Gebe 3A in das DATA Feld.



1800A.B.

Gebe B in das Data Feld.



1801x.x.

ADDR Feld erhöht sich um eins.
Erneute Datenbyte-Eingabe möglich.



1801C.D.

Datenbyte: CD eingeben.

Beispiel: Verändere den Dateninhalt zu Adresse 0000:

Taste

Anzeige

Bemerkung



0.0.0.0 6

ADRESSE ist 0000



0000.6



Der Inhalt vom ROM kann **nicht** verändert werden.
Solange Tasten betätigt bleiben, erscheint keine
Anzeige. Nach Lösen der Taste erscheint die ur-
sprüngliche Anzeige.

0000.6.

rkklärung]:

Addr weist auf eine Adresse. Nach Betätigen dieser Taste ist das Display in der Grundeinstellung. Die linken 4 Zahlen stehen für die Adresse, die rechten beiden für den Dateninhalt. Die Adressziffern sind durch 4 Indexpunkte gekennzeichnet. Bei Eingabe von mehr als 4 Zahlen werden nur jeweils die letzten 4 Zahlen benutzt. Bei weniger als 4 Zahlen werden die ersten automatisch mit "0" ergänzt.

Nach Betätigen von  , sind nur noch die 2 rechten Zahlen mit Indexpunkten versehen: = Hinweis für den Anwender, Daten einzugeben. Der Inhalt des RAM wird durch die neuen Daten ersetzt. Betätigen von  oder  erhöht oder verringert die Adresse jeweils um 1. Solange die Indexpunkte im Datenfeld sind, ist ein erneutes Betätigen der Data-Taste nicht nötig. Nach Betätigen von  kann direkt  oder  verwendet werden.

Falls der Anwender versucht, Inhalte im ROM zu ändern, erlischt das Display, solange die Taste betätigt bleibt. Nach Lösen wird der ursprüngliche Inhalt angezeigt.

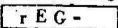
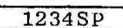
3.1.3 Prüfen und Ändern der Register: REG und DATA Tasten

REG <register name>[DATA [{data}] + [{data}] +

Beispiel: Überprüfe Inhalt der SP,HL,IY Register. Ändere Inhalt von A in 12, und Register F in 34.

Taste	Anzeige	Bemerkung
REG	r E C -	MPF-I in REGister-Anzeige-Art
SP A	x x x x S P [Register Name]	Anzeige von Registername und Speicherinhalt von SP
HL 3	x x x x H L	Anzeige von Registername und Speicherinhalt von HL
IY 9	x x x x I Y	Anzeige von Registername und Speicherinhalt von IY
AF 0	x x x x A F	Anzeige von Registername und Speicherinhalt von AF
DATA	x x x.x.A F	DATA bewirkt 2 Indexpunkte, die Eingabe für Speicher F ermöglichen.
HL 3 AF 4	x x 3.4.A F	
+	x.x.3 4	Indexpunkte für Register A nach links durch + .
BC 1 DE 2	1.2.3 4 A F [F Register] [A Register]	Eingabe von 1 und 2 für Register A.

rkklärung]

⟨Register Name⟩ ist die Darstellung für ein Register. Jedes Register wird durch Betätigen einer Taste angesprochen. Nach Drücken von  erscheint am Display  als Hinweis, den Namen des Registers zu wählen. Nach Eingabe des Namens erscheint im rechten Feld der Registername, im linken Feld der Inhalt. Z.B. bedeutet , daß der Inhalt der Stapelebene (Stack Pointer) 1234 ist. ⟨Register Name⟩ ist ein Befehl, der durch eine einzige Tastenbetätigung ausgeführt wird. Um mehrere Register zu überprüfen, genügt es, jeweils den ⟨Register Name⟩ zu drücken.

Um den Inhalt der Register zu ändern werden ,  oder  benötigt. Jeweils ein Byte wird geändert. ,  oder  bewirkt die zusätzliche Anzeige von jeweils 2 Indexpunkten als Hinweis, welches Byte geändert werden kann.  und  verändern die Index-Punkt-Position wie folgt:

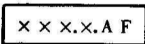
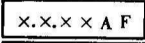
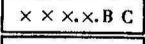
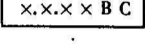
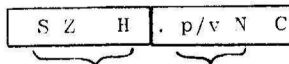
Name	Anzeige
	Flags 
↓	Reg A 
↑	Reg C 
	Reg B 
	⋮
	⋮
	⋮

Bild 3-1 Positionierung der Index-Punkte für Register

Besondere Register Inhalte

1. AF', BC', DE', HL', Flag' wird durch einen Dezimalpunkt an der rechten Seite angezeigt.
2. IX, IY wird mit IF, IY angedeutet.
3. Register I und Unterbrechung IFF 2 wird mit IF angezeigt.

Die Bedeutung jedes Bits wird wie folgt festgelegt:



FH (Flag hoch)

FL (Flag niedrig)

"." = keine Bedeutung

Bild 3-2 Flag Register

MPF-I dekodiert das Flag-Register und zeigt es in vier Bit-Gruppen an. Das Ändern eines dekodierten Flags erfolgt durch Ändern des LSB. Beim nächsten Überprüfen ist eine automatische Änderung von z.B. <AF> feststellbar.

Beispiel: Überprüfe das Flag und verändere es:

Taste	Anzeige	Bemerkung
REG AF	× × 2 9 A F	Inhalt von F ist 29.
PMC D	1 0 0 1 F L	Überprüfe Carry Bit
DATA	1 0 0 . 1 F L	
AF O	1 0 0 0 F L	Setze Carry Flag zurück
REG AF	× × 2 8 A F	F ist automatisch geändert

4 Programm Zähler — Taste

Setze den Programmzähler zurück. Das RAM von MPF-I hat 2 K Bytes und kann auf 4 K Bytes erweitert werden. Durch zurücksetzen des Monitors findet man die niedrigste RAM Adresse (1800) und setzt gleichzeitig den Programmzähler für den Anwender auf diese Adresse. zeigt nach Betätigung im linken Feld die niedrigste RAM-Adresse.

Taste	Anzeige	Bemerkung
RS	UPF -- 1	
PC	1800××	Niedrigste RAM Adresse

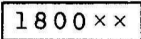
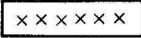
Beispiel

3.2 Programm Ausführung

3.2.1 Programm Start — Taste

 Taste kann nur im Adress-Mode betätigt werden. Nach Betätigung springt der Rechner zu der vom Display angezeigten Adresse. Vor Ausführung des Anwender-Programms werden alle Register überprüft. Eine Register-Voreinstellung kann über  vorgenommen werden.

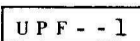
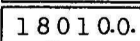
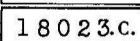
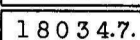
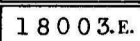
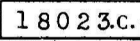
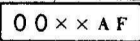
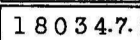
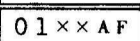
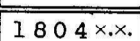
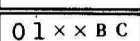
Beispiel: Programm Start

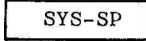
Taste	Anzeige	Bemerkung
		Das linke Feld ist keine Adresse
		Die Anzeige ist leer als Hinweis für Fehlbedienung
		Nach lösen der  -Taste wieder ursprüngliche Anzeige
		Linkes Feld wird eine Adresse
		Rechner startet bei Programm-Ausführung von Adresse 1800 aus.

2.2 Einzelschritt — Taste

 kann nur in Adress-Mode betätigt werden. Hierbei führt der Rechner genau einen Befehl entsprechend dem Anwenderprogramm aus. Nach der Ausführung zeigt die Anzeige den nächsten Befehl mit Dateninhalt an. Nach jedem Einzelschritt kann der Anwender Register oder Speicher ändern.

Beispiel: Programmeingabe in RAM, Ausführung in Einzelschritten

Taste	Anzeige	Bemerkung
		System zurücksetzen
		Start bei Adresse 1800
 		
 		LD A, 0 Lade Speicher A mit Null
  		INC A Erhöhe Speicher A um Eins
  		LD B,A Lade Speicher B mit Inhalt von Speicher A
		Startadresse in Adress-Mode
		1. Schritt: PC wird 1802
 		Register A ist Null (00)
 		2. Schritt: PC wird 1803
 		Register A um Eins erhöht (01)
 		3. Schritt: PC wird 1804
 		Register B wird Eins (01)

Bei Ausführung von Einzelschritten benützt der Monitor den Anwender Stapelpunkt, um eine Rückführadresse zu speichern. Diese Adresse muß auf Anwenderadressen zeigen, andernfalls wird Fehler mit  angezeigt. Führt die Rückführadresse zu einer System-(Monitor-) Adresse erfolgt . Eine Überschneidung von Stapeln verursacht bei "RET"-Befehl zum Fehler; Überprüfen der Rückführadresse, RESET. (siehe auch Abschnitt 4-5!)

3.2.3 Unterbrechung — Taste

Bei sehr langen Programmen ist das Einzelschrittverfahren sehr zeitaufwendig. Das Einfügen von Unterbrechungspunkten verursacht einen sofortigen Programmablauf-Stop bei Erreichen der Unterbrechungsadresse. An diesem Punkt können dann wieder wie bei  Speicher und/oder Register-Inhalte überprüft oder geändert werden.

In Adress-Mode mit einer RAM-Adresse bewirkt das Betätigen von  eine Unterbrechung genau an dieser Adresse.

Beispiel: Lade ein Programm in das RAM. Füge  ein, um das Ergebnis dieses Befehls zu beobachten.

Adresse	Maschinen Code	Befehl
1800	3E00	LD A, 0
1802	3C	INC A
1803	47	LD B, A
1804	04	INC B
1805	48	LD C, B
1806	FB	EI

Taste	Anzeige	Bemerkung
RS	UPF - 1	Zurücksetzen
ADDR BC 1 IX B AF 0 AF 0	1.8.0.0.××	Start-Adresse
DATA + AF 0 AF 0	18010.0.	Mode: Dateneingabe
+ HL 3 SZ-H C	18023.C.	Erhöhe Programmzähler um Eins
+ AF 4 HL 7	18034.7.	Erhöhe Programmzähler um Eins
+ AF 0 AF 4	18040.4.	Erhöhe Programmzähler um Eins
+ AF 4 IX B	18054.8.	Erhöhe Programmzähler um Eins
+ -PNC F I-IF B	1806 F.B.	Erhöhe Programmzähler um Eins
SN	1.8.0.6.F.B.	Füge Unterbrechungspunkt an Adresse 1806 ein
ADDR BC 1 IX B AF 0 AF 0	1.8.0.0.3 E	Programm startet bei 1800
GO	1807 ×.×	Programm wird ausgeführt: 1800 ... 1806. Das Programm bleibt bei Adresse 1807 stehen, das Datenbyte jedoch nicht mehr ausgeführt.
REG	REG -	Überprüfung des Ergebnisses in den Registern
AF 0	01 × × A F A reg F reg	Inhalt von Register A ist korrekt
BC 1	0202 B C B reg C reg	Inhalt von Register B ist korrekt
I-IF B	0001 I F	Flip-Flop für Unterbrechung ist gesetzt als Ergebnis des EI-Befehls (Enable interrupt).
PC -	1.8.0.6.F.B.	Ändere EI in DI (Disable interrupt)
-PNC F HL 3	1.8.0.6.F.3.	Eingabe von F3 für Adresse 1806
ADDR BC 1 IX B AF 0 AF 0	1.8.0.0.3 B	Programm startet bei 1800
GO	1807 ×.×	Programm-Ausführung
REG I-IF B	0000 I F	Überprüfe Register I-IF

Hinweis: Flip-Flop ist zurückgesetzt als Ergebnis des DI-Befehls

[Erklärung]

- (1) Unterbrechungspunkte können nicht im ROM-Bereich gesetzt werden. Monitor löscht Anzeige, nach Lösen der jeweiligen Taste, erscheint ursprünglicher Inhalt.
- (2) Falls ein Befehl aus mehr als 1 Byte besteht, muß der Unterbrechungspunkt zum ersten Byte gesetzt werden, andernfalls entstehen Fehler.
- (3) Beim Setzen eines Unterbrechungspunktes (nur in Adress-Mode möglich!) erscheinen 6 Indexpunkte als Anzeige der Unterbrechung an dieser Adresse.
- (4) Der Dateninhalt einer Unterbrechungsadresse kann wie immer auch geändert werden.
- (5) Bei erfolgter Unterbrechung ist die Anzeige in Adress-Mode.
- (6) Am Unterbrechungspunkt werden alle Inhalte und Register nicht verändert und stehen zur weiteren Verarbeitung zur Verfügung.
- (7) Siehe 3.2.2 für Stapel-Regeln.

Löschen einer Unterbrechung **CBR** Taste

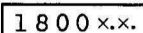
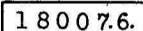
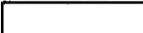
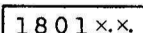
Wenn ein gesetzter Unterbrechungspunkt nicht mehr benötigt wird, wird durch
Betätigen von **CBR** der Unterbrechungsbefehl gelöscht und auf **F.F.F.F.F.F.**
verschoben. Diese Taste wird jederzeit akzeptiert.

3.2.5 Sofortiger Stop – Taste

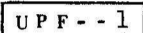
Bei der Ausführung des Programms können diverse Fehler auftreten. Der Rechner verliert die Kontrolle beim Vorhandensein von nichtzulässigen Befehlscodes, oder wenn undefinierte Schleifen enthalten sind.

 , d.h. der Monitor übernimmt die Kontrolle ähnlich wie beim Einzelschrittverfahren. Der Anwenderprogrammschritt wird mit Dateninhalt angezeigt. Nach Ausführung einer "HALT" Instruktion und Eingabe von  , übernimmt der Monitor die Kontrolle und checkt die Stapelebene des Anwenders. Es gelten die gleichen Regeln wie für Einzelschritt und Unterbrechung.

Beispiel: "HALT" und Rückkehr in den Monitor

Taste	Anzeige	Bemerkung
 		
 		Eingabe von "HALT"
		Rechner stoppt, keine Anzeige, LED HALT leuchtet
		Display in Adress-Mode der nächstfolgenden Adresse, alle Register erhalten

Beispiel:  in Monitor-Mode

Taste	Anzeige	Bemerkung
		Der Monitor überprüft die Tastatur. Das System sieht den Monitor als Programm.
		Der Stapelpunkt ist gerade im System: Bedienfehler — Fehleranzeige.

Übertragungsfunktionen

1 Block Transfer — Taste

<Adresse> <Adresse> <Adresse>

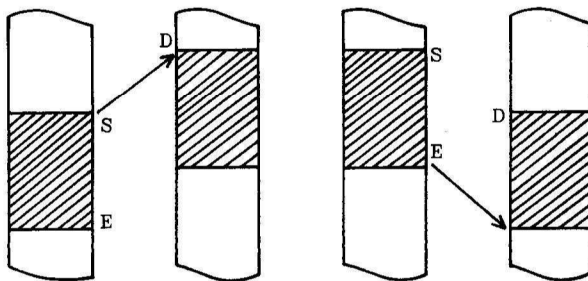
Beispiel: Übertrage die Daten von 1800 — 18FF in 1810 — 190F

Taste	Anzeige	Bemerkung
MOVE	x.x.x.x.- S	S ist Start-Adresse
BC 1	1.8.0.0.- S	Start-Adresse ist 1800
+	x.x.x.x.- E	E ist End-Adresse
BC 1	1.8.F.F.- E	End-Adresse ist 18FF
+	x.x.x.x.- d	D ist Bestimmungs-Adresse
BC 1	1.8.1.0.- d	Bestimmungs-Adresse ist 1810
GO	1 8 1 0 x.x.	Block Transfer ist erfolgt, das zuletzt übertragene Byte gehört zu Adresse 1810.

[Erklärung]

Die Übertragung kann sowohl aufwärts wie abwärts erfolgen: Bei Übertragung aufwärts ist die zuletzt bewegte Adresse die untere Grenze des Bestimmungsblockes. Bei Übertragung abwärts ist die zuletzt bewegte Adresse die obere Grenze des Bestimmungsblockes (Bild 3-3).

Aufgrund der hohen Verarbeitungsgeschwindigkeit dieses Mikrocomputers, ist der Transfer nahezu sofort abgeschlossen. Nach Betätigung der Taste  ist die Anzeige des Ergebnisses fast sofort im Display.



aufwärts

abwärts

Bild 3-3



Funktion

(der Pfeil zeigt das erste zu übertragende Byte an)

Falls sich der Bestimmungsbereich mit dem Systembereich überschneidet, müßte der Systembereich zerstört werden: Fehleranzeige erfolgt.

 -Betätigung führt in den Startzustand zurück.

Daten löschen —  Taste

 wird nur in Adress-Mode betätigt. Die gerade angezeigte Adresse wird mit Inhalt gelöscht. Alle nachfolgenden Adressen werden mit Dateninhalt um 1 Position nach vorne verschoben.

Achtung: Schleifen mit direkten Adressen sind nicht mehr funktionsfähig!

Beispiel: Inhalt des RAM's mit angenommenen Daten:

	Adresse	Alte Daten	Daten nach Löschvorgang
	1800	00	00
	1801	11	11
zu löschende Adresse →	1802	11	22
	1803	22	33
	1804	33	44
	1805	44	XX

Taste Anzeige Bemerkung

     1.8.0.2.1 1

Eingabe der zu löschenden Adresse

 1 8 0 2 2 2

Der alte Inhalt von 1802 ist gelöscht und die nachfolgenden Daten werden um 1 Position nach vorne verschoben. Der neue Inhalt von 1802 ist 22, der ursprüngliche Inhalt von 1803.

     1.8.0.0.0 0
 1 8 0 1 1.1.
 1 8 0 2 2.2.
 1 8 0 3 3.3.
 1 8 0 4 4.4.

Überprüfung

[Erklärung]

Daten im ROM können auf diese Weise nicht gelöscht werden.  ist anwendbar für die Adressen 1800 ... 1DFF. Alle Daten in diesem Bereich werden nach der gelöschten Adresse um 1 Position nach oben verschoben, die letzte Adresse 1DFF wird mit "00" belegt.

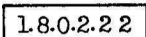
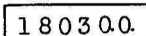
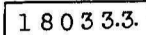
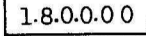
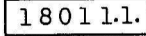
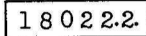
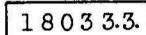
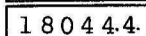
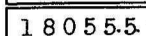
3.3.3 Daten einfügen — Taste

 <data>

Betätigen der  -Taste bewirkt eine Dateneinfügung nach der gerade angezeigten Adresse.

Beispiel: RAM-Inhalt wird wie folgt angenommen:

	Adresse	Alte Daten	Daten nach Einfügung
	1800	00	00
	1801	11	11
	1802	22	22
Füge hier 33 ein →	1803	44	33
	1804	55	44
	1805	66	55

Taste	Anzeige	Bemerkung
    		Adress-Mode 1802
		Füge 1 Byte nach 1802 ein, Adresse wird 1803
 		Gebe Daten 33 ein
    		Überprüfe
		
		
		
		
		

[Erklärung]

Für den anwendbaren Bereich für  gilt das gleiche wie für . Nach der Einfügung von Daten geht das letzte Byte des vorhandenen Datenblocks verloren. (im Beispiel 66).

Die eingefügte Adresse ist eine Position nach der gewählten und angezeigten Position. Alle nachfolgenden Daten werden um eine Position nach unten verschoben. Zu der eingefügten Adresse können die gewünschten Daten hinzugefügt werden.

Relative Adresse — **RELA** Taste

Die Befehle JR und DJNZ erfordern die Eingabe von relativen Adressen.
MPF-I enthält im Monitorprogramm die Kalkulation der relativen Adressen.

Dieses Programm wird mit **RELA** angesprochen.

RELA <Adresse> **+** <Adresse> **GO**

Beispiel: Ein JR-Befehl erfordert einen Sprung von 1800 nach 1804

Taste	Anzeige	Bemerkung
RELA	x.x.x.x - S	S ist Startadresse
BC 1 IX 8 AF 0 AF 0	1.8.0.0 - S	Startadresse ist 1800
+	x.x.x.x - d	D ist Bestimmungsadresse
BC 1 IX 8 AF 0 AF 4	1.8.0.4 - d	Bestimmungsadresse ist 1804
GO	1 8 0 1 0 2.	MPF-I errechnet die relative Adresse, speichert das Ergebnis im nächsten Byte des JR-Befehls und zeigt das Resultat an.

[Erklärung]

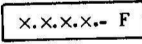
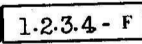
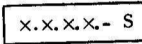
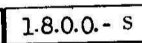
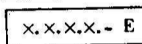
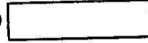
Nach Eingabe von **RELA** wird **x.x.x.x - S** angezeigt. S weist auf die Startadresse von Befehl JR oder DJNZ. Nach **+** wird die Anzeige **x.x.x.x - d**. d weist auf die Bestimmungsadresse von JR oder DJNZ hin. Nach **GO** errechnet MPF-I die relative Adresse und speichert sie im 2. Byte des Befehls. Die Anzeige ist im Adress-Mode. Diese Adresse erhält dann die relative Adressierung. Wenn das Ergebnis dezimal + 127 oder - 128 überschreitet wird **- E r r** angezeigt.

3.3.5 Speichern auf Band — Taste

Ein einfaches Kassetten-Tonbandgerät ist das kostengünstigste Speichermedium mit hoher Kapazität. MPF-I beinhaltet die Anschlußbuchsen sowie die Monitor-Software.

 <File Name>  <Adresse>  <Adresse> 

Beispiel: Speichere die Daten von 1800 ... 18FF auf Band, benütze hierzu File-Name 1234

Taste	Anzeige
	
   	
	
   	
	
   	
(PLAY & REC)	
	

Bemerkung

F ist File-name

File-Name ist 1234

S ist Start-Adresse

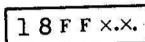
Start-Adresse ist 1800

E ist End-Adresse

End-Adresse ist 18FF

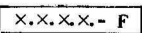
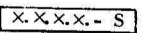
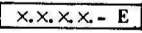
Verbinde Mikrophon-Anschluß des Bandgerätes mit MIC-Buchse von MPF-I
Aufnahmestart durch Betätigung von PLAY- und REC-Taste am Rekorder.

 startet den Datentransfer.
Während der Übertragung bleibt die Anzeige dunkel, TONE-OUT LED leuchtet.



Ende der Übertragung wird angezeigt durch Aufleuchten der End-Adresse.

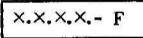
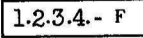
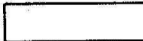
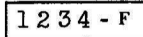
klärung]

Betätigen von  bewirkt die Anzeige . F weist darauf hin, daß jetzt die gewünschte File-Nummer eingegeben werden muß. Verschiedene File-Nummern ermöglichen die Unterscheidung von verschiedenen abgespeicherten Programmen auf einer Kassette. Nach  wird  angezeigt. S stellt die Start-Adresse der zu übertragenden Daten her. Nach erneutem  wird das Display , wobei E die End-Adresse der zu übertragenden Daten bedeutet. Vor Betätigung von  muß das Tonband in Aufnahme-Mode bereits laufen. Andernfalls werden Daten ausgesendet, die dann nicht abgespeichert werden. Während der Übertragung ist das LED-Display dunkel, TONE-OUT LED leuchtet, aus dem Lautsprecher tönt die codierte Information.

3.3.6 Wiedergabe vom Band — Taste

 <File Name> 

Beispiel: Überspiele Daten vom Tonband aus File 1234

Taste	Anzeige	Bemerkung
		F ist File-Name
   		File-Name ist 1234 Verbinde Kopfhöreranschluß vom Tonband mit EAR-Buchse am MPF-I
 (PLAY)		Start: Während der Ausführung keine Anzeige am Display, solange MPF-I den File-Namen sucht
		Rekorder-Start durch Drücken von PLAY, maxi- male Lautstärke eingestellt. MPF-I Lautsprecher liest die abgespeicherten Daten. Bei zu kleiner Lautstärke ertönt kein Ton! Jeder erfaßte File- Name erscheint ca. 1,5 Sek. am Display. Erkennen des gesuchten File's wird mit einem Anzeige-Wechs-
		von in — — — — bestätigt. Nach Abschluß der gewünschten Übertragung wird die letzte übertragene Adresse mit Daten angezeigt.
		

lärung]

Start- und End-Adresse müssen hier nicht eingegeben werden, da diese Daten bereits auf der Kassette gespeichert sind. Durch das gleichzeitige Lesen der Daten und Tonausgabe über Lautsprecher wird nach etwas Übung eine richtige Übertragung leicht erkannt. Gleichzeitig wird über eine entsprechende Codierung die Übertragung vom Monitor kontrolliert. Eine fehlerhafte Übertragung wird erkannt und mit **- E r r** angezeigt. Dann Übertragung wiederholen, eventuell Lautstärke am Tonbandgerät anders einstellen.

4. Software und Hardware Beschreibung

4.1 Speicher Adresse

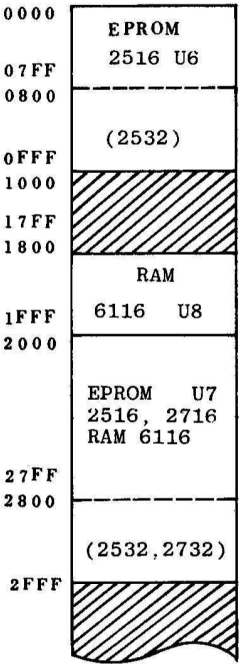


Bild 4-1 Speicher-Übersicht

klärung]

(1) U 6 EPROM: Monitor-Programm

U 7 RAM oder EPROM: vorgesehen für Erweiterung

U 8 RAM: Achtung! 1F9F bis 1FF3 werden vom Monitor benutzt!

(2) Alle Adress-Leitungen sind im MPF-I bereits vorhanden. Bei Verwendung folgender Bausteine sind keine Änderungen auf der gedruckten Schaltung notwendig: U 7: 2516, 2716, 2532

a) U 7: 2732
folgende Trennungen oder Verbindungen sind nötig

Trennungen	Verbindungen
Pin 1 und 2	Pin 2 und 3
Pin 3 und 4	Pin 4 und 5
Pin 5 und 6	Pin 6 und 7

b) U 7: 6116

Trennungen	Verbindungen
3 und 4	4 und 5

Siehe Seite I - 4: Jumper 1

4.2 Adresse Eingabe/Ausgabe

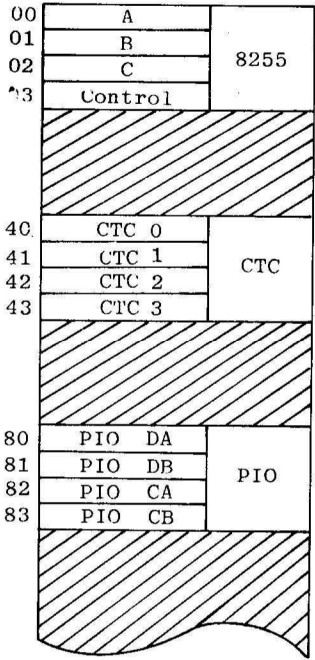


Bild 4-2 Adressen-Übersicht

Klärung]

- (1) Der Baustein 8255 hat eine programmierbare Interface-Funktion.
24 I/O-(Ein-/Ausgabe-)Linien sind in drei 8-Bit-Blöcke aufgeteilt.
(Siehe Datenblatt 8255.)
- (2) Das Kontrollwort für 8255 ist 90H. Block A ist ein Eingabe-Block,
Block B und C sind Ausgabe-Blöcke.

a) Block A (Adresse 00):

Bit 7: Tonbandeingang

Bit 6:  , aktiv-Pegel = 0

Bit 5 ... 0: verbunden mit 6 Tastatur-Reihen
Eingangssignal wird 0, wenn Tasten der aktiven
Reihen betätigt werden.

b) Block B (Adresse 01):

Kontrolle der sieben Segmente und des Dezimal-Punktes der LED-
Anzeigen. Bild 4-3 zeigt die entsprechenden Segment-Bezeichnungen.
Ausgangs-Bits sind aktive Pegel = L.

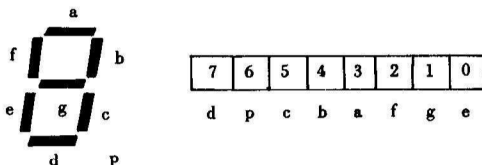


Bild 4-3 Segment-Bezeichnung

c) Block C (Adresse 02):

Bit 7: Tonbandausgang, Lautsprecher, LED TONE-OUT.

LED leuchtet, wenn Ausgang = 0

Bit 6: Unterbrechkungskontrolle für Monitor, Jeglicher Versuch dieses Bit zu ändern ist nicht erlaubt!

Bit 5 ... 0: Verbunden mit 6 Reihen von Display- und Tastatur-Matrix.

Bit 0 ist ganz rechts, Bit 5 ist ganz links. Diese Bits sind alle aktiv auf L.

- (3) Die Z-80 CTC (Option) Zähl-, Zeitablauf-Steuerung gibt die Möglichkeit 4 voneinander unabhängige Zähl- oder Zeitsteuerungen zu programmieren. Die Adressen für CTC sind 40H ... 43H.
- (4) Die Z-80 PIO (Option) Parallel-Ein-/Ausgabe enthält 2 Blöcke um TTL-kompatible Peripherie mit dem Z 80 Rechner anzusprechen. Die Adressen für PIO sind 80H ... 83H.
- (5) Vollständig dekodierte Adressen: A0, A1, A6, A7
Nicht dekodierte Adressen: A2 ... A5

Programm Unterbrechung

Eine nicht änderbare Unterbrechung wird ausschließlich vom Monitor benützt. Pin 16 vom Rechner CPU ist mit  Taste verbunden. Wenn 0038 vom Monitor ausgeführt wird, übernimmt die Adresse, die in 1FEE und 1FEF gespeichert ist, die Kontrolle. Während dieses Vorganges haben alle CPU-Inhalte keinen Einfluß. Die Inhalte von 1FEE und 1FEF können mit 0066 angenommen werden. An dieser Stelle startet dann der Serviceablauf. 0038 wird in folgenden Fällen ausgeführt:

- (1) Art 1 Unterbrechung ist erkannt
- (2) Befehl "RST 38H" mit Befehlscode FF wird ausgeführt
- (3) Die Leitungen des Daten-Bus' liegen hoch. Art 0 Unterbrechung ist erkannt, jedoch ohne Unterbrechungs-Hinweis. Dann wird RST 38H ausgeführt.
- (4) Falls durch ein fehlerhaftes Programm in eine nicht vorhandene Adresse führt, wird vom CPU der Befehl FF ausgeführt.

Wenn die Inhalte von 1FEE und 1FEF nach dem Einschalten der Stromversorgung nicht geändert werden, ist die Ausführung des Befehls 0038 das gleiche, wie das Betätigen von  oder Setzen eines Unterbrechungspunktes. Der Anwender kann eigene Service-Abläufe durch Verändern der Daten von Service-Zieladressen 1FEE und 1FEF definieren.

4.4 Software Stop-Instruktion RST 30H (Opcode F7)

RST 30H mit Befehls-Code F7 hat den gleichen Effekt wie eine Unterbrechung, wird jedoch Software-Unterbrechung genannt, da ein externer Einfluß, wie Betätigen einer Taste o.ä., entfällt.

Anwendung: Vorgesehener Programmablauf-Stop, oder Vielfach-Unterbrechung in der Programm-Entwicklung.

Stapelblock (Stack)

Bild 4-4 zeigt den Aufbau eines Stapels.

Der ursprüngliche Wert des Anwender-Stapel-Hinweises wird mit 1F9F angenommen. Bei jeder Unterbrechung des Anwenderprogrammes überprüft der

Monitor den Stapelhinweis. **SYS-SP** wird angezeigt, wenn dieser Hin-

weis auf den System-Stapel hinführt. Ist eine stapelbezogene Instruktion (z.B. RET) im Anwenderprogramm, kann ein Fehler auftreten, wenn sich Anwenderstapel und System-Stapel überschneiden.

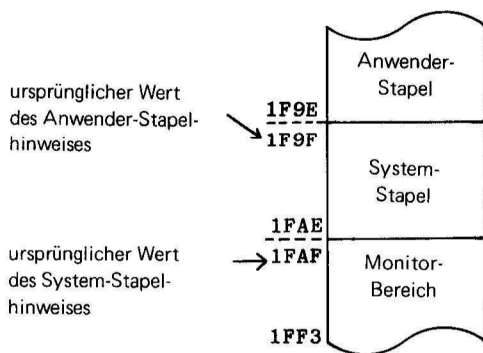


Bild 4-4 Darstellung Stapel-Hinweise

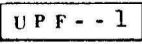
SYS-SP kann in folgenden Fällen auftreten:

- (1) -Taste drücken, solange der Monitor den CPU überprüft. Hierbei werden alle Register-Inhalte zerstört.
- (2) Ausführen des Monitorprogrammes mit Taste.

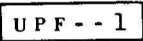
4.6 Reset

Hierbei gibt es 2 mögliche Arten, nachdem der Monitor zurückgesetzt ist:

(1) Spannungsversorgung EIN

- a) Unterbrechung abschalten (IFF = 0)
- b) I-Register = 0
- c) Unterbrechungsart = 0
- d) Anwenderprogrammzähler = 1800
- e) Anwenderstapelhinweis = 1F9F
- f) Unterbrechungspunkt abgeschaltet
- g) Setze Inhalt von 1FEE auf 66, und 1FEF auf 00.
Bei Ausführung von 0038 springt der Rechner zu Adresse 0066,
gleichbedeutend mit Betätigung der  Taste.
- h)  wird angezeigt, startend mit dem 1. Buchstaben,
der von rechts nach links wandert.

(2) Betätigen der Taste

- (a) — (e) wie unter (1)
- (f) u. (g) ohne Einfluß
- h) sofortige Anzeige von 

Standaufzeichnungs-Format

(1) Bit-Format:

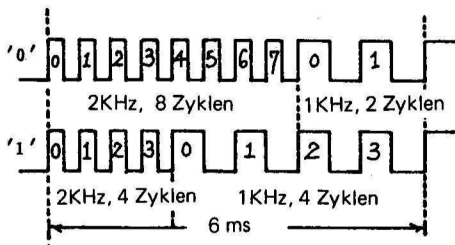


Bild 4-5 Bit-Format

(2) Byte-Format:

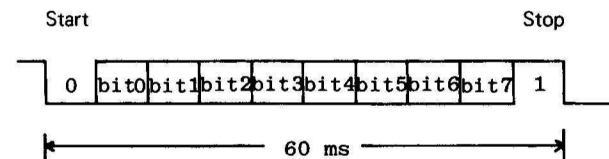


Bild 4-6 Byte-Format

(3) File-Format:

Start-Sync	File Name	Start Adr.	End Adr.	Chk Sum	Mid sync	Daten	End-Sync
1KHz	2	2	2	1	2KHz	Variable	2KHz
4sec	Byte	Byte	Byte	Byte	2sec	Länge je nach Inhalt	2sec

Bild 4-7 File-Format

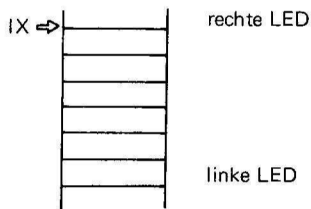
5. Monitor Unterroutinen

5.1 Überblick

Adresse	Bezeichnung	Funktion
0624	SCAN1	Überprüfe Tastatur und zeige einen Zyklus an.
05FE	SCAN	Überprüfe Tastatur und zeige solange an bis eine neue Eingabe folgt.
0689	HEX7	Wandle eine hexadezimale Zahl in das 7-Segment Anzeige-Format
0678	HEX7SG	Wandle zwei hexadezimale Zahlen in die 7-Segment-Anzeigen-Formate
05F6	RAMCHK	Überprüfe, ob die eingegebene Adresse im RAM enthalten ist.
05E4	RONE	Erzeuge einen Ton
05DE	TONE1K	Erzeuge einen Ton von 1 KHz
05E2	TONE2K	Erzeuge einen Ton von 2 KHz

CAN1

- [Adresse]: 0624
- [Funktion]: Überprüfe Tastatur und zeige einen Zyklus am Display von rechts nach links an. Die Ausführungszeit beträgt ~ 10 msec (exakt 9.97 msec)
- [Eingabe]: IX zeigt zum Anzeige-Buffer
- [Ausgabe]: (1) keine Tastatureingabe: Carry Flag = 1
(2) Tastatureingabe: Carry Flag = 0 und der Positions-Code der Taste ist in Register A gespeichert (siehe Anhang A).
- [Register]: Zerstöre Inhalt von AF, AF', BC', DE'
- [Ergänzung]: (1) 6 Bytes sind für 6 LED's erforderlich
(2) IX weist auf die ganz rechte LED
IX + 5 weist auf die ganz linke LED



- (3) Siehe Anhang A für die Zuordnung jedes Bits und den 7 Segmenten.

5.3 SCAN

[Adresse]: 05FE

[Funktion]: Ähnlich SCAN1 außer:

- (1) Überprüfung und Abtasten bis neue Eingabe erfolgt.
- (2) SCAN führt zum Positions Code der betätigten Taste zurück.

[Eingabe]: IX zeigt zum Anzeige-Buffer

[Ausgabe]: Register A enthält den internen Tasten-Code.

[Register]: Zerstöre Inhalt von AF, B, HL, AF', BC', DE'.

IE X7

- [Adresse]: 0689
- [Funktion]: Dekodiere eine hexadezimale Zahl in das 7-Segment-Format
- [Eingabe]: Die letzten 4 Bits von Register A enthalten die Hexadezimalziffer (0 ... F).
- [Ausgabe]: Das Ergebnis wird auch in Register A gespeichert.
- [Register]: Zerstöre Inhalt von AF

5.5 HEX7SG

[Adresse]: 0678

[Funktion]: Wandle zwei hexadezimale Ziffern in das 7-Segment-Format.

[Eingabe]: Die 1. Zahl den rechten 4 Bits von A.
Die 2. Zahl den linken 4 Bits von A.

[Ausgabe]: Das 1. Anzeigemuster ist in (HL)
Das 2. Anzeigemuster ist in (HL+1)
HL wird um 2 erhöht.

[Register]: Zerstöre Inhalt von AF, HL.

AMCHK

- [Adresse]: 05F6
- [Funktion]: Überprüfe, ob die eingegebene Adresse im RAM enthalten ist.
- [Eingabe]: Die zu überprüfende Adresse in HL eingeben.
- [Ausgabe]: Falls in RAM enthalten, Zero-Flag = 1.
Falls in RAM nicht enthalten, Zero-Flag = 0.
- [Register]: Zerstöre Inhalt von AF.

5.7 TONE

[Adresse]: 05E4

[Funktion]: Erzeuge einen Ton

[Eingabe]: (1) C ergibt die Tonfrequenz:

$$\text{Frequenz} = \frac{200}{(10 + 3C)} \text{ KHz}$$

$$\text{Periode} = (44 + 13C) \times 2 \times 0,56 \mu\text{sec}$$

(2) HL enthält die Anzahl der Schwingungen, maximal 32768

[Ausgabe]: keine

[Register]: Zerstöre Inhalt von AF, B, DE, HL.

ONE1K

Adresse]: 05DE

Funktion]: Erzeuge Ton von 1KHz.

Eingabe]: Anzahl der Schwingungen ist in HL.

Ausgabe]: Keine

Register]: Zerstöre Inhalt von AF, BC, DE, HL.

5.9 TONE2K

- [Adresse]: 05E2
- [Funktion]: Erzeuge Ton von 2KHz.
- [Eingabe]: Anzahl der Schwingungen ist in HL.
- [Ausgabe]: Keine
- [Register]: Zerstöre Inhalt von AF, BC, HL, DE.

rogramm-Beispiele

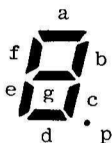
Beispiel 1: Zeige "HELPUS", HALT wenn STEP betätigt wird.

```

1800          1  ;DISPLAY 'HELP US' UNTIL STEP-KEY PUSHED:
1800 DD212018 2          ORG      1800H
1804 CDFE05    3          LD      IX,HELP
1807 FE13      4  DISP    CALL    SCAN
1809 20F9      5          CP      13H          ;KEY-STEP
180B 76        6          JR      NZ,DISP
1820          7          HALT
1820          8  ;
1820          9          ORG      1820H
1820 AE        10  HELP    DEFB    0AEH          ;'S'
1821 B5        11          DEFB    0B5H          ;'U'
1822 1F        12          DEFB    01FH          ;'P'
1823 85        13          DEFB    085H          ;'L'
1824 8F        14          DEFB    08FH          ;'E'
1825 37        15          DEFB    037H          ;'H'
1825          16  ;
1825          17  SCAN    EQU     05FEH
1825          18  END

```

Details des Display Buffers sind hier gezeigt:



Position	Display Format	Beleuchtetes Segment	d p c b a f g e	Daten	Adr.
Rechts ↑ Links	S	a,c,d,f,g,	1 0 1 0 1 1 1 0	AE	1820
	U	a,b,c,d,e,f,	1 0 1 1 0 1 0 1	B5	1821
	P	a,b,e,f,g,	0 0 0 1 1 1 1 1	1F	1822
	L	d,e,f,	1 0 0 0 0 1 0 1	85	1823
	E	a,d,e,f,g,	1 0 0 0 1 1 1 1	8F	1824
	H	b,c,e,f,g,	0 0 1 1 0 1 1 1	37	1825

— siehe Anhang A —

Beispiel 2: "HELP US" blinkend

Benütze Routine SCAN1 zur blinkenden Anzeige von "HELP US". 50 Schleifen
SCAN1 ergeben ca. 0,5 sec. Leuchtdauer.

```

1  ;FLASH 'HELP US':
2      ORG      1800H
1800 212618 3      LD      HL,BLANK
1803 E5      4      PUSH   HL
1804 DD212018 5      LD      IX,HELP
1808 DDE3     6  LOOP   EX      (SP), IX
180A 0632     7      LD      B,50
180C CD2406   8  HELFSEG CALL  SCAN1
180F 10FB     9      DJNZ   HELFSEG
1811 18F5    10      JR      LOOP
11 ;
12      ORG      1820H
1820 AE      13  HELP   DEFB   0AEH      ;'S'
1821 B5      14      DEFB   0B5H      ;'U'
1822 1F      15      DEFB   01FH      ;'P'
1823 85      16      DEFB   085H      ;'L'
1824 8F      17      DEFB   08FH      ;'E'
1825 37      18      DEFB   -037H     ;'H'
1826 00      19  BLANK  DEFB   0
1827 00      20      DEFB   0
1828 00      21      DEFB   0
1829 00      22      DEFB   0
182A 00      23      DEFB   0
182B 00      24      DEFB   0
25 ;
26 SCAN1 EQU 0624H
27      END

```

Der Inhalt von Adresse 180B bestimmt die Blinkfrequenz.

Beispiel 3: Zeige den Positions-Code der gerade betätigten Taste an.

```

1      ;DISPLAY INTERNAL CODE
1800      2      ORG      1800H
1800 DD210019      3      LD      IX,OUTBF
1804 CDFE05      4      LOOP   CALL   SCAN
1807 210019      5      LD      HL,OUTBF
180A CD7806      6      CALL   HEX7SG
180D 18F5      7      JR      LOOP
      8      ;
1900      9      ORG      1900H
1900 00      10     OUTBF   DEFB   0
1901 00      11           DEFB   0
1902 00      12           DEFB   0
1903 00      13           DEFB   0
1904 00      14           DEFB   0
1905 00      15           DEFB   0
      16      ;
      17     SCAN   EQU      05FEH
      18     HEX7SG EQU      0678H
      19      END
```

Sobald eine Taste betätigt wird, erscheint der interne Befehl im Display.
Der Anwender kann diese Anzeige mit dem Anhang A vergleichen.

Zur Anzeige des Positions-Codes der betätigten Taste wird das Programm wie folgt geändert:

	1	;DISPLAY POSITION CODE	
	2	ORG	1800H
1800		LD	IX,OUTBF
1800 DD210019	3	CALL	SCAN1
1804 CD2406	4	JR	C,LOOP
1807 38FB	5	LD	HL,OUTBF
1809 210019	6	CALL	HEX7SG
180C CD7806	7	JR	LOOP
180F 18F3	8		
	9		

Beispiel 4: Wandle 3 fortlaufende Bytes in 7-Segment-Format, speichere Ergebnis auf Adresse 1903 ... 1908 und zeige sie im Display an.

	1	;DISPLAY 3 BYTES IN RAM TO 6 HEXA-DIGITS	
	2	ORG	1800H
1800		LD	DE,BYTE0
1800 110019	3	LD	HL,OUTBF
1803 210319	4	LD	B,3
1806 0603	5	LD	A,(DE)
1808 1A	6	CALL	HEX7SG
1809 CD7806	7	INC	DE
180C 13	8	DJNZ	LOOP
180D 10F9	9		
	10	;CONVERSION COMPLETE, BREAK FOR CHECK	
180F DD210319	11	LD	IX,OUTBF
1813 CDFE05	12	CALL	SCAN
1816 76	13	HALT	
	14	;	
	15	ORG	1900H
1900		DEFB	10H
1900 10	16	DEFB	32H
1901 32	17	DEFB	54H
1902 54	18	OUTBF	DEFS 6
1903	19		
	20	;	
	21	HEX7SG EQU	0678H
	22	SCAN EQU	05FEH
	23	END	

Die 3 Bytes der binären Daten sind in 1900 ... 1902 gespeichert. Der Anwender kann einen Unterbrechungspunkt auf 180F legen, um die Wandlung vor der Ergebnisanzeige zu überprüfen.

Beispiel 5: Erzeugung einer Polizei-Sirene

Der Ton einer Polizei-Sirene wird durch abwechselndes Generieren zweier verschiedener Frequenzen simuliert. Register C kontrolliert die Tonfrequenz, Registerpaar HL kontrolliert die Tondauer.

	1	; POLICE CAR SIREN:		
1800		2	ORG	1800H
1800	0E00	3	LOOP	LD C, 0
1802	21C000	4		LD HL, 0C0H
1805	CDE405	5		CALL TONE
1808	0EC0	6		LD C, 0C0H
180A	210001	7		LD HL, 100H
180D	CDE405	8		CALL TONE
1810	18EE	9		JR LOOP
		10	;	
		11	TONE	EQU 05E4
		12	END	

- (1) Niedrige Frequenz: C = 0 (≙ 256)
 HL = C0H (≙ 192)

Periodendauer = (44 + 13 x 256) x 2 x 0,56 = 3777 μsec

Frequenz = $\frac{1}{3777 \mu\text{sec}}$ = 265 Hz

Tondauer = 3777 μsec x 192 = 0,73 sec.

- (2) Hohe Frequenz: C = C0H (≙ 192)
 HL = 100H (≙ 256)

Periodendauer = (44 + 13 x 192) x 2 x 0,56 = 2845 μsec

Frequenz = $\frac{1}{2845 \mu\text{sec}}$ = 352 Hz

Tondauer = 2845 μsec x 256 = 0,73 sec.

6. Überprüfen der Speicher

6.1 EPROM 0000-07FF

Die Summe aller Monitor-Codes ergibt 0. Routine ROMTEST mit Adresse 06A6 überprüft Monitor-EPROM.

```
ROMTEST:
06A6      210000      LD      HL,0
06A9      010008      LD      BC,800H
06AC      CD3105      CALL    SUM
06AF      2801        JR      Z,SUMOK
06B1      76          HALT     ;IF ERROR
06B2      C7          SMUOK    RST     0      ;DISPLAY 'UPF--1'
;

0531      AF          SUM      XOR      A
0532      86          SUMCAL   ADD      A,(HL)
0533      EDA1        CPI
0535      EA3205      JP      PE,SUMCAL
0538      B7          OR      A
0539      C9          RET
```

Dieses Programm errechnet die Summe aller EPROM-Codes. Bei Ergebnis = 00 wird "UPF--1" angezeigt.

Die Eingabefolge lautet:



(1) Richtig: Die Anzeige ist U P F - - 1

(2) Fehler: HALT LED leuchtet.

6. Überprüfen der Speicher

6.1 EPROM 0000-07FF

Die Summe aller Monitor-Codes ergibt 0. Routine ROMTEST mit Adresse 06A6 überprüft Monitor-EPROM.

```
ROMTEST:
06A6      210000      LD      HL,0
06A9      010008      LD      BC,800H
06AC      CD3105      CALL    SUM
06AF      2801        JR      Z,SUMOK
06B1      76          HALT     ;IF ERROR
06B2      C7          SMUOK    RST     0      ;DISPLAY 'UPF--1'
;

0531      AF          SUM      XOR      A
0532      86          SUMCAL   ADD      A,(HL)
0533      EDA1        CPI
0535      EA3205      JP      PE,SUMCAL
0538      B7          OR      A
0539      C9          RET
```

Dieses Programm errechnet die Summe aller EPROM-Codes. Bei Ergebnis = 00 wird "UPF--1" angezeigt.

Die Eingabefolge lautet:



(1) Richtig: Die Anzeige ist U P F - - 1

(2) Fehler: HALT LED leuchtet.

2 RAM Region 1800-1FFF

```

RAMTEST:
0694      210018      LD      HL,1800H
0697      010008      LD      BC,800H
069A      CDF605      RAMT    CALL   RAMCHK
069D      2801        JR      Z,TNEXT
069F      76          HALT                    ;IF ERROR
06A0      EDA1        TNEXT  CPI
06A2      EA1E07      JP      PE,RAMT
06A5      C7          RST      0          ;DISPLAY 'UPF--1'
;

```

```

RAMCHK:
05F6      7E          LD      A,(HL)
05F7      2F          CPL
05F8      77          LD      (HL),A
05F9      7E          LD      A,(HL)
05FA      2F          CPL
05FB      77          LD      (HL),A
05FC      BE          CP      (HL)
05FD      C9          RET

```

Dieses Programm überprüft jedes Byte im Bereich 1800-1FFF. Nach jedem Byte = GUT wird der Test weitergeführt, bis alle Bytes überprüft sind. Falls ein nichtfunktionstüchtiges Byte vorkommt, stoppt das Programm, die HALT LED leuchtet.

Nach ☐ ☐ und ☐ wird die fehlerhafte Adresse angezeigt, anschließendes ☐ ergibt auch den Dateninhalt.

☐ ☐ fährt mit dem kontinuierlichen Text fort.

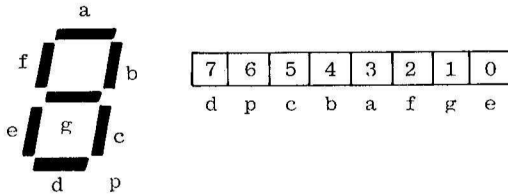
RS ADDR DE 6 IV 9 AF 4 GO

(1) Richtig: ist angezeigt

(2) Fehler: HALT LED leuchtet.

ANHANG

A-1. Display Format, Positions-Code und Interner-Code



DISPLAY FORMAT :

CODE	BD	30	9B	BA	36	AE	AF	38	BF	FE	3F	A7	BD	B3
DATA	0	1	2	3	4	5	6	7	8	9	A	B	C	D
DISP	0	1	2	3	4	5	6	7	8	9	A	b	c	d

CODE	8F	0F	AD	37	89	B1	97	85	2B	23	A3	1F	3E	03
DATA	E	F	C	H	I	J	K	L	M	N	G	P	Q	R
DISP	E	F	G	H	I	J	K	L	M	N	O	P	Q	R

CODE	A6	87	B5	B7	A9	07	B6	8A	83	A2	32	02	C0	00
DATA	S	T	U	V	W	X	Y	Z	(	)	+	-	,	
DISP	S	T	U	V	W	X	Y	Z	(	)	+	-	,	

1. Positions-Code (CALL SCANI):

1E SBR	18 CBR	12 '0'	0C '1'	06 '2'	00 '3'
1F '-'	19 PC	13 '4'	0D '5'	07 '6'	01 '7'
20 DATA	1A REG	14 '8'	0E '9'	08 'A'	02 'B'
21 '+'	1B ADDR	15 'C'	0F 'D'	09 'E'	03 'F'
22 INS	1C DEL	16 GO	10 STEP	0A	04
23 MOVE	1D RELA	17 TPWR	11 TPRD	0B	05

2. Interner Code (CALL SCAN):

15 SBR	1A CBR	00 '0'	01 '1'	02 '2'	03 '3'
11	18 PC	04 '4'	05 '5'	06 '6'	07 '7'
14 DATA	1B REG	08 '8'	09 '9'	0A 'A'	0B 'B'
10 '+'	19 ADDR	0C 'C'	0D 'D'	0E 'E'	0F 'F'
16 INS	17 DEL	12 GO	13 STEP	22	20
1C MOVE	1D RELA	1E TPWR	1F TPRD	23	21

ANGABE

Theorie der Schaltkreise

A. System Frequenz

U3a, U3b und der 3,58 MHz Quarz produzieren ein 3,58 MHz Signal. Dieses Signal wird an U2a Pin 3 gelegt und nach Teilung durch 2 in das System mit 1,79 MHz Clock-Frequenz geführt.

B. Reset Signal

U2b erzeugt das Reset-Signal entweder durch Einschalten der Versorgungsspannung, oder durch Betätigen von ☐. Die aufbereiteten RST- und $\overline{\text{RST}}$ -Signale werden zum CPU, zur CTC und zum 8255 als Reset-Signale geleitet.

C. Speicher Adressierung

MREQ	A15	A14	A13	A12	A11	A10	---	A0	Chip	Adresse
0	0	0	0	0	X	X	---	X	U6	0000-0FFF
0	0	0	1	0	X	X	---	X	U7	2000-2FFF
0	0	0	0	1	1	X	---	X	U8	1800-1FFF

U6 enthält MPF-I Monitor Programm, EPROM Typ TMS 2516, Intel 2716, o.ä. —

U7 ist für spätere Erweiterung als RAM oder ROM gedacht, Typen HM 6116, 2716, 2516, 2532, 2732 unter Beachtung der Hinweise auf Seite 4 der Schaltplandarstellung.

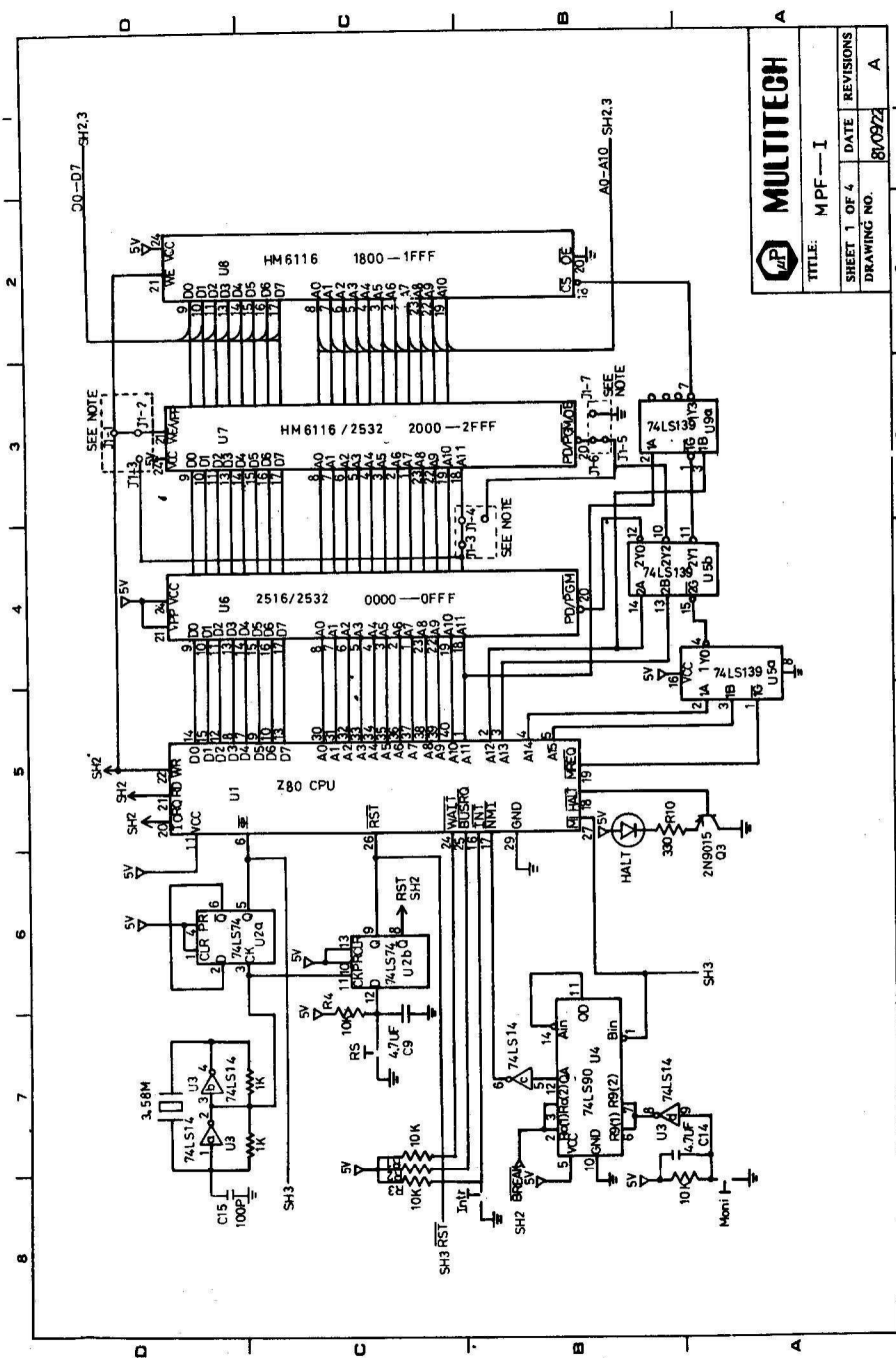
U8 ist das System RAM mit 2 K Byte Speicherinhalt.

D. Adressierung der Ein-/Ausgabe Blöcke

U9b (74LS139) ist der Block-Dekoder

IORQ	A7	A6	I/O	Block Adresse
0	0	0	8255	00 - 03
0	0	1	CTC	40 - 43
0	1	0	PIO	80 - 83

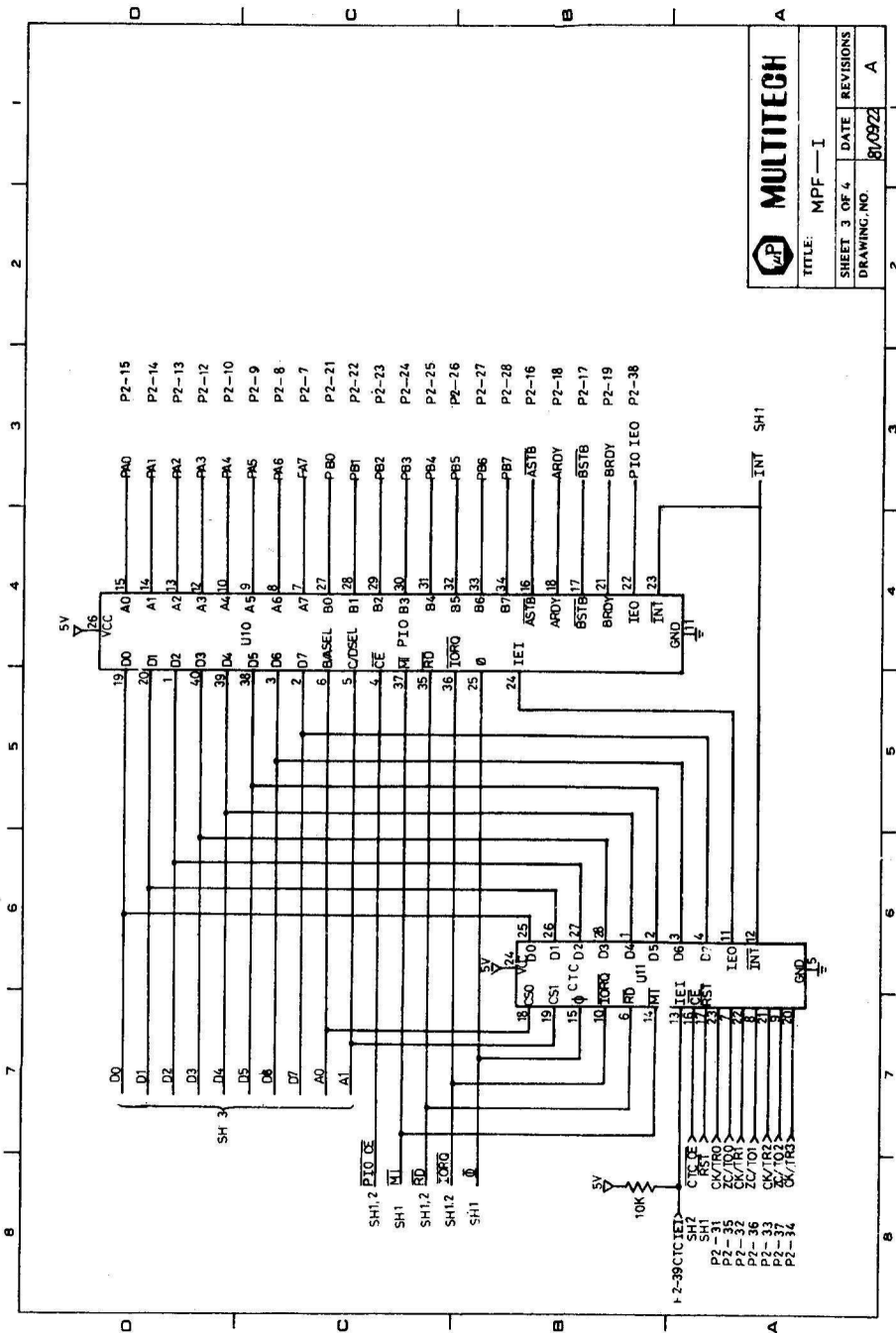
Hinweis: Ein-/Ausgabe Blöcke sind nicht alle dekodiert, z.B. 00-03, 04-07, 08-0B, ..., 3C-3F; es gibt 16 verschiedene Kombinationen den 8255 anzusprechen; ähnliches gilt für CTC und PIO.



MULTITECH

TITLE: MPF-1

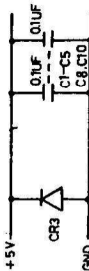
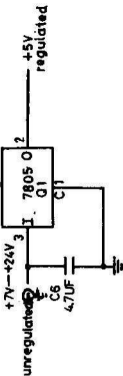
SHEET 1 OF 4	DATE	REVISIONS
DRAWING NO.	8/09/22	A



P2 PIN FUNCTION

PIN NO	SIGNAL	PIN NO	SIGNAL
1	NC	21	PB0
2	NC	22	PB1
3	NC	23	PB2
4	NC	24	PB3
5	NC	25	PB4
6	NC	26	PB5
7	PA7	27	PB6
8	PA6	28	PB7
9	PA5	29	NC
10	PA4	30	GND
11	GND	31	CK/TR0
12	PA3	32	CK/TR1
13	PA2	33	CK/TR2
14	PA1	34	CK/TR3
15	PA0	35	ZC/T00
16	A5TB	36	ZC/T01
17	B5TB	37	ZC/T02
18	ARDY	38	PT0 IEO
19	3RDY	39	CTC IEI
20	NC	40	NC

PIN NO	SIGNAL	PIN NO	SIGNAL
1	A11	21	A10
2	A12	22	A9
3	A13	23	A8
4	A14	24	A7
5	A15	25	A6
6	A16	26	A5
7	D4	27	A4
8	D3	28	A3
9	D5	29	A2
10	D6	30	A1
11	NC	31	A0
12	D2	32	GND
13	D7	33	RF SH
14	D0	34	MI
15	D1	35	RESET
16	INT	36	BUSRO
17	NMI	37	WALT
18	HALT	38	BUSAK
19	HFREQ	39	WR
20	IORQ	40	RD



Hinweis: 1. U7 ist für Erweiterung vorgesehen:
2516, 2716, 2532, 2732, 6116
2. Lötbrücken (Jumper) 1 sind zur Auswahl
verschiedener Speichertypen vorgesehen:

Speicher Typ	Speicher Adresse	Trennen	Verbinden
2516 2716	2000-27FF	1) NIL	1a) NIL
2532	2000-2FFF	NIL	NIL
2732	2000-2FFF	PTN3 TO PTN4	PTN2 TO PTN3 PTN4 TO PTN5 PTN5 TO PTN6 PTN6 TO PTN7 PTN7 TO PTN8 PTN8 TO PTN9 PTN9 TO PTN10 PTN10 TO PTN11 PTN11 TO PTN12 PTN12 TO PTN13 PTN13 TO PTN14 PTN14 TO PTN15
6116	2000-27FF	PTN3 TO PTN4	PTN4 TO PTN5 PTN5 TO PTN6 PTN6 TO PTN7 PTN7 TO PTN8 PTN8 TO PTN9 PTN9 TO PTN10 PTN10 TO PTN11 PTN11 TO PTN12 PTN12 TO PTN13 PTN13 TO PTN14 PTN14 TO PTN15

E. Tastatur-Matrix und Anzeige

U14 (8255) hat 3 Ein-/Ausgabe (I/O-) Blöcke:

Pins PB0 ... PB7 kontrollieren die Segment-Anzeige

U15 und U17 sind die Segment-Treiber

Pins PC0 ... PC7 treffen die Digit-Auswahl für die Anzeige

U13 (75492) ist der 6-Digit-Treiber.

Die LED-Anzeige wird mit Multiplex-Verfahren betrieben, d.h. es leuchtet jeweils nur ein Digit, startend von rechts nach links. Auf Grund der hohen Multiplex-Frequenz erscheint die Anzeige als stetig leuchtend.

Während der LED-Anzeige wird die Tastaturbetätigung von U14 (8255) Pins PA0 ... PA5 überprüft. Ohne Tasteneingabe liegen diese Pins hoch, L. Jede betätigte Taste wird über die Tastatur-Abstroutine vom CPU erkannt. Von den 36 Tasten werden jedoch nur 32 über dieses Verfahren abgetastet.

F. Anwender-Taste

Diese Taste, Pin PA6 von 8255, ist anwenderorientiert belegbar.

G. Tonband Interface

Das Programm, bzw. alle Daten, die auf Tonband gespeichert werden, laufen über Pin PC7 vom 8255. Eine Filterung und Anpassung erfolgt über C13, R11, C12, R12 und R13. Das angepaßte Signal geht zu MIC (= Mikrophon)-Anschluß. Q2 betreibt LED und Lautsprecher. PC7 ist ebenso der Anschluß für Tonband-Ausgang.

Die Daten werden vom Tonband über EAR (Kopfhörer)-Anschluß in den RAM-Speicher zurückgelesen. Über R14, CR2, CR1 und C11 erfolgt wieder die entsprechende Anpassung auf TTL-Pegel und wird vom CPU Pin PA7 vom 8255 weiterverarbeitet.

H. Einzelschritt, Unterbrechungspunkt und Monitor-Unterbrechung

PC6 ist normal hoher Pegel, L. Dieses Signal wird zu RO-Pin von U4 (7490) geleitet, der Ausgang von U4 wird auf 0000 gesetzt, Pin NMI vom Z-80 hochgelegt. Sobald der Programmzähler einen Unterbrechungspunkt erreicht, oder MPF-I Einzelschritte ausführt, wird PC6 Ausgang niedrig, 0 – U4 startet zu zählen, nach dem 5. Befehls-Code wird NMI niedrig, 0. Dieser Vorgang stoppt die Programm-Ausführung und führt in den Monitor zurück.

Logik Tabelle für U4 (74LS90)

	R9	R0	Qa	Q0	Qc	Qb	NMI	Bemerkung
Normal State	0	1	0	0	0	0	1	U4 preset to 0000
BREAK wird low	0	0	0	0	0	0	1	$R0 = \overline{BREAK} = 0$
1st MI	0	0	0	0	0	1	1	7490 Start counting
2nd MI	0	0	0	0	1	0	1	Qd, Qc, Qb is Mod, 5 Counter
3rd MI	0	0	0	0	1	1	1	
4th MI	0	0	0	1	0	0	1	
5th MI	0	0	1	0	0	0	0	Qa from 0-1 when Qd from 1-0
Pressing Key 	1	0	1	1	0	0	0	U4 Preset to 1001

Nach Betätigen von  wird R9 von U4 hoch, Qa wird hoch, NMI wird niedrig. Somit springt der CPU auf Grund der Unterbrechung zurück in das Monitor-Programm.

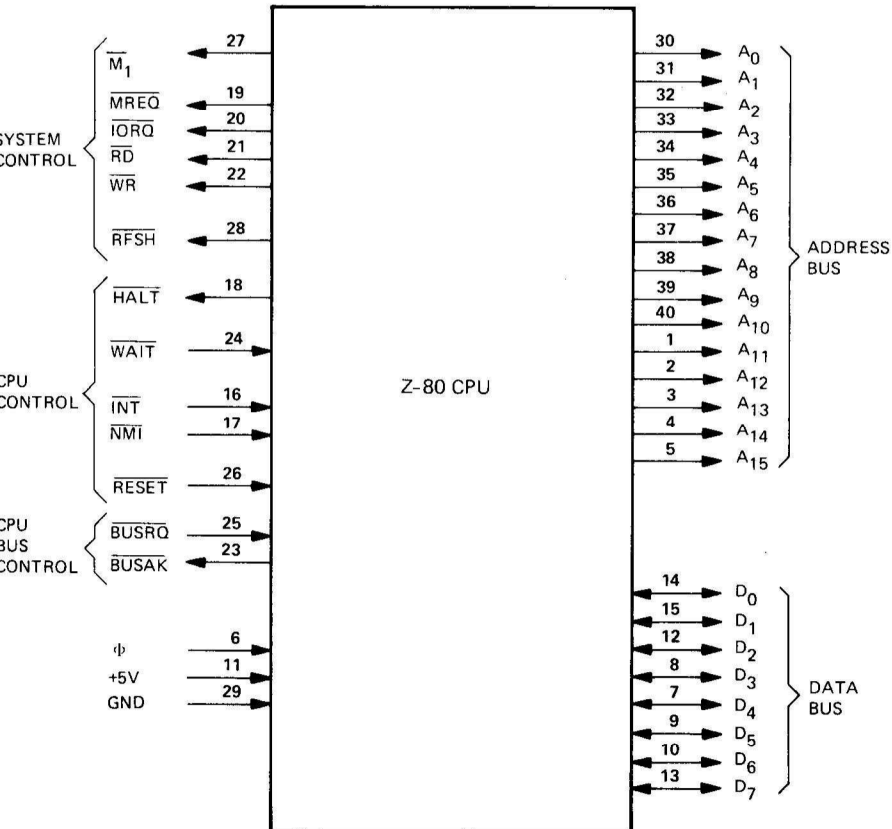
I. PIO und CTC

U11 (CTC) und U10 (PIO) sind hintereinander anzusprechen: CTC hat Unterbrechungs-Vorrang. CTC-IE1, PIO-IE0, CTC Kanal-Signale und PIO-I/O-Port sind auf den Stecker P2 für spätere Anwender-Erweiterungen geführt.

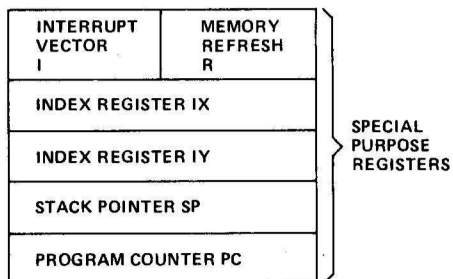
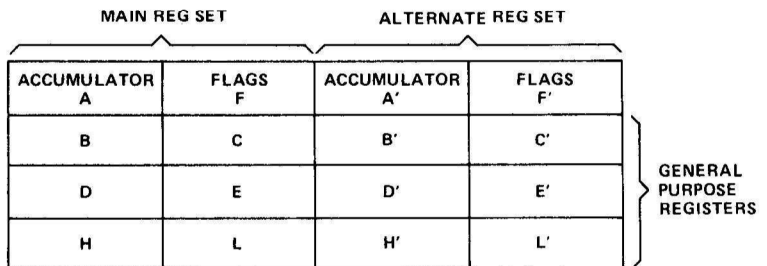
ANHANG C

Z80-CPU Programme, Befehle

Z-80 PIN CONFIGURATION



Z-80 CPU REGISTER CONFIGURATION



SUMMARY OF FLAG OPERATION

Instruction	C	Z	P/V	S	N	H	Comments
ADD A, s; ADC A, s	†	†	V	†	0	†	8-bit add or add with carry
SUB s; SBC A, s, CP s, NEG	†	†	V	†	1	†	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	†	P	†	0	1	Logical operations
OR s; XOR s	0	†	P	†	0	0	
INC s	0	†	V	†	0	†	8-bit increment
DEC m	0	†	V	†	1	†	8-bit decrement
ADD DD, ss	†	0	0	0	X		16-bit add
ADC HL, ss	†	†	V	†	0	X	16-bit add with carry
SBC HL, ss	†	†	V	†	1	X	16-bit subtract with carry
RLA; RLCA, RRA, RRCA	†	0	0	0	0		Rotate accumulator
RL m; RLC m; RR m; RRC m	†	†	P	†	0	0	Rotate and shift location m
SLA m; SRA m; SRL m							
RLD, RRD	0	†	P	†	0	0	Rotate digit left and right
DAA	†	†	P	†	0	†	Decimal adjust accumulator
CPL	0	0	0	1	1		Complement accumulator
SCF	1	0	0	0	0		Set carry
CCF	†	0	0	0	X		Complement carry
IN r, (C)	0	†	P	†	0	0	Input register indirect
INI; IND; OUTI; OUTD	0	†	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	0	1	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI, LDD	0	X	†	X	0	0	Block transfer instructions
LDIR, LDDR	0	X	0	X	0	0	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	0	†	†	†	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, i; LD A, R	0	†	IFF	†	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	0	†	X	X	0	1	The state of bit b of location s is copied into the Z flag
NEG	†	†	V	†	1	†	Negate accumulator

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C=1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z=1 if the result of the operation is zero.
S	Sign flag. S=1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V=1 if the result of the operation is even, P/V=0 if result is odd. If P/V holds overflow, P/V=1 if the result of the operation produced an overflow.
H	Half-carry flag. H=1 if the add or subtract operation produced a carry into or borrow from into bit 4 of the accumulator.
N	Add/Subtract flag. N=1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
†	The flag is affected according to the result of the operation.
0	The flag is unchanged by the operation.
1	The flag is reset by the operation.
X	The flag is set by the operation.
	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ix	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>
m	Any 8-bit location for all the addressing modes allowed for the particular instruction.

8 BIT LOAD GROUP 'LD'

		SOURCE															
		IMPLIED		REGISTER								REG INDIRECT			INDEXED		EXT. ADDR
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	
DESTINATION	REGISTER	A	ED 57	ED 5F	7F	7B	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n
		B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d	
		C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d	
		D			57	50	51	52	53	54	56	56			DD 56 d	FD 56 d	
		E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d	
		H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d	
		L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d	
	REG INDIRECT	(HL)			77	70	71	72	73	74	75						
		(BC)			02												
		(DE)			12												
	INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d						
		(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d						
	EXT. ADDR	(nn)			32 n n												
	IMPLIED	I			ED 47												
		R			ED 4F												

8-BIT LOAD GROUP

Mnemonic	Symbolic Operation	Flags						OP-Code			No. of Bytes	No. of M Cycles	No. of T Cycles	Comments	
		C	Z	P/V	S	N	H	76	543	210					
LD r, r'	r ← r'	•	•	•	•	•	•	01	r	r'	1	1	4	r, r'	Reg.
LD r, n	r ← n	•	•	•	•	•	•	00	r	110	2	2	7	000	B
									←	n	→			001	C
LD r, (HL)	r ← (HL)	•	•	•	•	•	•	01	r	110	1	2	7	010	D
LD r, (IX+d)	r ← (IX+d)	•	•	•	•	•	•	11	011	101	3	5	19	011	E
									01	r	110			100	H
									←	d	→			101	L
LD r, (IY+d)	r ← (IY+d)	•	•	•	•	•	•	11	111	101	3	5	19	111	A
									01	r	110				
									←	d	→				
LD (HL), r	(HL) ← r	•	•	•	•	•	•	01	110	r	1	2	7		
LD (IX+d), r	(IX+d) ← r	•	•	•	•	•	•	11	011	101	3	5	19		
									01	110	r				
									←	d	→				
LD (IY+d), r	(IY+d) ← r	•	•	•	•	•	•	11	111	101	3	5	19		
									01	110	r				
									←	d	→				
LD (HL), n	(HL) ← n	•	•	•	•	•	•	00	110	110	2	3	10		
									←	n	→				
LD (IX+d), n	(IX+d) ← n	•	•	•	•	•	•	11	011	101	4	5	19		
									00	110	110				
									←	d	→				
									←	n	→				
LD (IY+d), n	(IY+d) ← n	•	•	•	•	•	•	11	111	101	4	5	19		
									00	110	110				
									←	d	→				
									←	n	→				
LD A, (BC)	A ← (BC)	•	•	•	•	•	•	00	001	010	1	2	7		
LD A, (DE)	A ← (DE)	•	•	•	•	•	•	00	011	010	1	2	7		
LD A, (nn)	A ← (nn)	•	•	•	•	•	•	00	111	010	3	4	13		
									←	n	→				
									←	n	→				
LD (BC), A	(BC) ← A	•	•	•	•	•	•	00	000	010	1	2	7		
LD (DE), A	(DE) ← A	•	•	•	•	•	•	00	010	010	1	2	7		
LD (nn), A	(nn) ← A	•	•	•	•	•	•	00	110	010	3	4	13		
									←	n	→				
									←	n	→				
LD A, I	A ← I	•	‡	IFF	‡	0	0	11	101	101	2	2	9		
									01	010	111				
LD A, R	A ← R	•	‡	IFF	‡	0	0	11	101	101	2	2	9		
									01	011	111				
LD I, A	I ← A	•	•	•	•	•	•	11	101	101	2	2	9		
									01	000	111				
LD R, A	R ← A	•	•	•	•	•	•	11	101	101	2	2	9		
									01	001	111				

Notes: r, r' means any of the registers A, B, C, D, E, H, L

IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,

‡ = flag is affected according to the result of the operation.

16 BIT LOAD GROUP
'LD'
'PUSH' AND 'POP'

		SOURCE								IMM. EXT.	EXT. ADDR.	REG INDI
		REGISTER										
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)	
DESTINATION	REGISTER	AF									F1	
		BC							01 n n	ED 4B n n	C1	
		DE							11 n n	ED 5B n n	D1	
		HL							21 n n	2A n n	E1	
		SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
		IX								DD 21 n n	DD 2A n n	DD E1
		IY								FD 21 n n	FD 2A n n	FD E1
	EXT. ADDR.	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS →		REG. IND.	(SP)	F5	C5	D5	E5		DD E5	FD E5		

NOTE: The Push & Pop Instructions adjust the SP after every execution

POP INSTRUCTION

16-BIT LOAD GROUP

Mnemonic	Symbolic Operation	Flags					Op-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	V	S	N					
LD dd, nn	dd ← nn	*	*	*	*	*	00 dd0 001	3	3	10	dd Pair
							← n →				00 BC
							← n →				01 DE
LD IX, nn	IX ← nn	*	*	*	*	*	11 011 101	4	4	14	10 HL
							00 100 001				11 SP
							← n →				
							← n →				
LD IY, nn	IY ← nn	*	*	*	*	*	11 111 101	4	4	14	
							00 100 001				
							← n →				
							← n →				
LD HL, (nn)	H ← (nn+1) L ← (nn)	*	*	*	*	*	00 101 010	3	5	16	
							← n →				
							← n →				
LD dd, (nn)	dd _H ← (nn+1) dd _L ← (nn)	*	*	*	*	*	11 101 101	4	6	20	
							01 dd1 011				
							← n →				
							← n →				
LD IX, (nn)	IX _H ← (nn+1) IX _L ← (nn)	*	*	*	*	*	11 011 101	4	6	20	
							00 101 010				
							← n →				
							← n →				
LD IY, (nn)	IY _H ← (nn+1) IY _L ← (nn)	*	*	*	*	*	11 111 101	4	6	20	
							00 101 010				
							← n →				
							← n →				
LD (nn), HL	(nn+1) ← H (nn) ← L	*	*	*	*	*	00 100 010	3	5	16	
							← n →				
							← n →				
LD (nn), dd	(nn+1) ← dd _H (nn) ← dd _L	*	*	*	*	*	11 101 101	4	6	20	
							01 dd0 011				
							← n →				
							← n →				
LD (nn), IX	(nn+1) ← IX _H (nn) ← IX _L	*	*	*	*	*	11 011 101	4	6	20	
							00 100 010				
							← n →				
							← n →				
LD (nn), IY	(nn+1) ← IY _H (nn) ← IY _L	*	*	*	*	*	11 111 101	4	6	20	
							00 100 010				
							← n →				
							← n →				
LD SP, HL	SP ← HL	*	*	*	*	*	11 111 001	1	1	6	
LD SP, IX	SP ← IX	*	*	*	*	*	11 011 101	2	2	10	
							11 111 001				
LD SP, IY	SP ← IY	*	*	*	*	*	11 111 101	2	2	10	
							11 111 001				qq Pair
PUSH qq	(SP-2) ← qq _L (SP-1) ← qq _H	*	*	*	*	*	11 qq0 101	1	3	11	00 BC
											01 DE
PUSH IX	(SP-2) ← IX _L (SP-1) ← IX _H	*	*	*	*	*	11 011 101	2	4	15	10 HL
							11 100 101				11 AF
PUSH IY	(SP-2) ← IY _L (SP-1) ← IY _H	*	*	*	*	*	11 111 101	2	4	15	
							11 100 101				
POP qq	qq _H ← (SP+1) qq _L ← (SP)	*	*	*	*	*	11 qq0 001	1	3	10	
POP IX	IX _H ← (SP+1) IX _L ← (SP)	*	*	*	*	*	11 011 101	2	4	14	
							11 100 001				
POP IY	IY _H ← (SP+1) IY _L ← (SP)	*	*	*	*	*	11 111 101	2	4	14	
							11 100 001				

Notes: dd is any of the register pairs BC, DE, HL, SP
 qq is any of the register pairs AF, BC, DE, HL
 (PAIR)_H, (PAIR)_L refer to high order and low order eight bits of the register pair respectively.
 E.g. BC_L = C, AF_H = A

Flag Notation: * = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
 † flag is affected according to the result of the operation.

EXCHANGES 'EX' AND 'EXX'

		IMPLIED ADDRESSING				
		AF	BC, DE & HL	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REG. INDIR.	(SP)			E3	DD E3	FD E3

BLOCK SEARCH GROUP

SEARCH LOCATION

REG. INDIR.	
(HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR', Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD' Dec HL & BC
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match

HL points to location in memory
to be compared with accumulator
contents
BC is byte counter

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP

Mnemonic	Symbolic Operation	Flags					Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P	V	S	N	H	76 543 210				
EX DE, HL	DE ← HL	•	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF'	AF ← AF'	•	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	$\begin{pmatrix} BC \\ DE \end{pmatrix} \leftrightarrow \begin{pmatrix} BC' \\ DE' \end{pmatrix}$ $\begin{pmatrix} HL \\ HL' \end{pmatrix}$	•	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ← (SP+1) L ← (SP)	•	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	IX _H ← (SP+1) IX _L ← (SP)	•	•	•	•	•	•	•	11 011 101	2	6	23	
EX (SP), IY	IY _H ← (SP+1) IY _L ← (SP)	•	•	•	•	•	•	•	11 111 101	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	•	•	1	•	•	•	0	11 101 101 10 100 000	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Repeat until BC = 0	•	•	0	•	•	•	0	11 101 101 10 110 000	2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	1	•	•	•	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Repeat until BC = 0	•	•	0	•	•	•	0	11 101 101 10 111 000	2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A ← (HL) HL ← HL+1 BC ← BC-1	•	1	1	1	1	1	1	11 101 101 10 100 001	2	4	16	
CPIR	A ← (HL) HL ← HL+1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	1	1	1	1	1	1	11 101 101 10 110 001	2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A ← (HL) HL ← HL-1 BC ← BC-1	•	1	1	1	1	1	1	11 101 101 10 101 001	2	4	16	
CPDR	A ← (HL) HL ← HL-1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	1	1	1	1	1	1	11 101 101 10 111 001	2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: ① P/V flag is 0 if the result of BC-1 ≠ 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
! = flag is affected according to the result of the operation.

8 BIT ARITHMETIC AND LOGIC

SOURCE

	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	CS n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

8-BIT ARITHMETIC AND LOGICAL GROUP

Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76	543	210				
A	Converts acc. content into packed BCD following add or subtract with packed BCD operands	‡	‡	P	‡	•	‡	00	100	111	1	1	4	Decimal adjust accumulator
	$A \leftarrow \bar{A}$	•	•	•	•	1	1	00	101	111	1	1	4	Complement accumulator (one's complement)
G	$A \leftarrow 0 - A$	‡	‡	V	‡	1	‡	11	101	101	2	2	8	Negate acc. (two's complement)
								01	000	100				
F	$CY \leftarrow \bar{CY}$	‡	•	•	•	0	X	00	111	111	1	1	4	Complement carry flag
F	$CY \leftarrow 1$	1	•	•	•	0	0	00	110	111	1	1	4	Set carry flag
P	No operation	•	•	•	•	•	•	00	000	000	1	1	4	
LT	CPU halted	•	•	•	•	•	•	01	110	110	1	1	4	
	$IFF \leftarrow 0$	•	•	•	•	•	•	11	110	011	1	1	4	
	$IFF \leftarrow 1$	•	•	•	•	•	•	11	111	011	1	1	4	
0	Set interrupt mode 0	•	•	•	•	•	•	11	101	101	2	2	8	
								01	000	110				
1	Set interrupt mode 1	•	•	•	•	•	•	11	101	101	2	2	8	
								01	010	110				
2	Set interrupt mode 2	•	•	•	•	•	•	11	101	101	2	2	8	
								01	011	110				

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

GENERAL PURPOSE AF OPERATIONS

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

MISCELLANEOUS CPU CONTROL

'NOP'	00
'HALT'	76
DISABLE INT '(DI)'	F3
ENABLE INT '(EI)'	FB
SET INT MODE 0 'IM0'	ED 46
SET INT MODE 1 'IM1'	ED 56
SET INT MODE 2 'IM2'	ED 5E

8080A MODE

CALL TO LOCATION 0038_H

INDIRECT CALL USING REGISTER
I AND 8 BITS FROM INTERRUPTING
DEVICE AS A POINTER.

GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS

Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76	543	210				
	Converts acc. content into packed BCD following add or subtract with packed BCD operands	‡	‡	P	‡	•	‡	00	100	111	1	1	4	Decimal adjust accumulator
	$A \leftarrow \bar{A}$	•	•	•	•	1	1	00	101	111	1	1	4	Complement accumulator (one's complement)
	$A \leftarrow 0 - A$	‡	‡	V	‡	1	‡	11	101	101	2	2	8	Negate acc. (two's complement)
	$CY \leftarrow \overline{CY}$	‡	•	•	•	0	X	00	111	111	1	1	4	Complement carry flag
	$CY \leftarrow 1$	1	•	•	•	0	0	00	110	111	1	1	4	Set carry flag
	No operation	•	•	•	•	•	•	00	000	000	1	1	4	
	CPU halted	•	•	•	•	•	•	01	110	110	1	1	4	
	$IFF \leftarrow 0$	•	•	•	•	•	•	11	110	011	1	1	4	
	$IFF \leftarrow 1$	•	•	•	•	•	•	11	111	011	1	1	4	
	Set interrupt mode 0	•	•	•	•	•	•	11	101	101	2	2	8	
	Set interrupt mode 1	•	•	•	•	•	•	11	101	101	2	2	8	
	Set interrupt mode 2	•	•	•	•	•	•	11	101	101	2	2	8	
								01	010	110				
								01	011	110				

es: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

16 BIT ARITHMETIC

		SOURCE					
		BC	DE	HL	SP	IX	IY
DESTINATION	'ADD'	HL	08	10	20	30	
		IX	DD 09	DD 19		DD 39	DD 29
		IY	FD 09	FD 19		FD 39	FD 29
	ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A	
	SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72	
	INCREMENT 'INC'		08	10	20	30	DD 23
DECREMENT 'DEC'			0C	1C	2C	3C	DD 2B

16-BIT ARITHMETIC GROUP

Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76	543	210				
ADD HL, ss	HL ← HL + ss	‡	•	•	•	0	X	00	ss1	001	1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	‡	•	V	‡	0	X	11	101	101	2	4	15	
SBC HL, ss	HL ← HL - ss - CY	‡	‡	V	‡	1	X	11	101	101	2	4	15	
ADD IX, pp	IX ← IX + pp	‡	•	•	•	0	X	11	011	101	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	‡	•	•	•	0	X	11	111	101	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	•	•	•	•	00	ss0	011	1	1	6	
INC IX	IX ← IX + 1	•	•	•	•	•	•	11	011	101	2	2	10	
INC IY	IY ← IY + 1	•	•	•	•	•	•	11	111	101	2	2	10	
DEC ss	ss ← ss - 1	•	•	•	•	•	•	00	ss1	011	1	1	6	
DEC IX	IX ← IX - 1	•	•	•	•	•	•	11	011	101	2	2	10	
DEC IY	IY ← IY - 1	•	•	•	•	•	•	11	111	101	2	2	10	

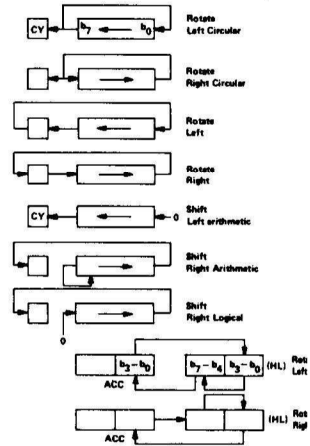
Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

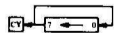
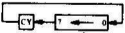
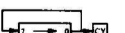
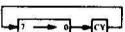
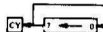
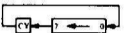
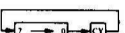
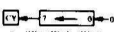
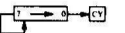
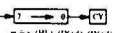
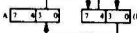
ROTATES AND SHIFTS

		Source and Destination									
		A	B	C	D	E	H	L	(HL)	(IX + d)	(IV + d)
TYPE OF ROTATE OR SHIFT	'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
	'RRC'	CB 0F	CB 06	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
	'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
	'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
	'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
	'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
	'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
	'RLD'								ED BF		
	'RRD'								ED 67		

	A
RLCA	67
RRCA	6F
RLA	17
RRA	1F



ROTATE AND SHIFT GROUP

Mnemonic	Symbolic Operation	Flags					Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	76	543	210				
RLCA		†	•	•	•	0	00	000	111	1	1	4	Rotate left circular accumulator
RLA		†	•	•	•	0	00	010	111	1	1	4	Rotate left accumulator
RRCA		†	•	•	•	0	00	001	111	1	1	4	Rotate right circular accumulator
RRA		†	•	•	•	0	00	011	111	1	1	4	Rotate right accumulator
RLC r		†	†	P	†	0	11	001	011	2	2	8	Rotate left circular register r
RLC (HL)		†	†	P	†	0	11	001	011	2	4	15	r Reg.
RLC (IX+d)		†	†	P	†	0	00	000	110	4	6	23	000 B
		†	†	P	†	0	11	011	101				010 C
		†	†	P	†	0	11	001	011				011 E
		†	†	P	†	0	11	111	101				100 H
		†	†	P	†	0	00	000	110				101 L
RLC (IY+d)	†	†	P	†	0	11	111	101	4	6	23	111 A	
RL m	 m = r, (HL), (IX+d), (IY+d)	†	†	P	†	0	01	0					Instruction format and states are as shown for RLC m. To form new OP-code replace 000 of RLC m with shown code
RRC m	 m = r, (HL), (IX+d), (IY+d)	†	†	P	†	0	00	1					
RR m	 m = r, (HL), (IX+d), (IY+d)	†	†	P	†	0	01	1					
SLA m	 m = r, (HL), (IX+d), (IY+d)	†	†	P	†	0	10	0					
SRA m	 m = r, (HL), (IX+d), (IY+d)	†	†	P	†	0	10	1					
SRL m	 m = r, (HL), (IX+d), (IY+d)	†	†	P	†	0	11	1					
RLD		•	†	P	†	0	11	101	101	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		•	†	P	†	0	11	101	101	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
† = flag is affected according to the result of the operation.

BIT MANIPULATION GROUP

		REGISTER ADDRESSING							REG. INDIR.	INDEXED		
		A	B	C	D	E	H	L	(HL)	((X+d)	((Y+d)	
TEST 'BIT'	BIT	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E	
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56	
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E	
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66	
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E	
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76	
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E	
RESET 'BIT' 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86	
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E	
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96	
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E	
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6	
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE	
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6	
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE	
SET 'BIT' 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6	
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE	
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6	
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE	
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6	
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE	
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6	
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE	

BIT SET, RESET AND TEST GROUP

Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments	
		C	Z	V	S	N	H	76	543	210					
b, r	$Z \leftarrow \bar{T}_b$	•	‡	X	X	0	1	11	001	011	2	2	8	r	Reg.
								01	b	r				000	B
b, (HL)	$Z \leftarrow \overline{(HL)}_b$	•	‡	X	X	0	1	11	001	011	2	3	12	001	C
								01	b	110				010	D
b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	•	‡	X	X	0	1	11	011	101	4	5	20	011	E
								11	001	011				100	H
								←	d	→				101	L
								01	b	110				111	A
b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	•	‡	X	X	0	1	11	111	101	4	5	20	b	Bit Tested
								11	001	011				000	0
								←	d	→				001	1
								01	b	110				010	2
														011	3
														100	4
														101	5
														110	6
														111	7
b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	11	001	011	2	2	8		
								11	b	r					
b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11	001	011	2	4	15		
								11	b	110					
b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11	011	101	4	6	23		
								11	001	011					
								←	d	→					
								11	b	110					
b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11	111	101	4	6	23		
								11	001	011					
								←	d	→					
								11	b	110					
s b, m	$s_b \leftarrow 0$ $m \equiv r, (HL),$ $(IX+d),$ $(IY+d)$							10						To form new OP-code replace 11 of SET b,m with 10. Flags and time states for SET instruction	

s: The notation s_b indicates bit b (0 to 7) or location s.

Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

JUMP, CALL and RETURN GROUP

CONDITION

			UN- COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B=0
JUMP 'JP'	IMMED. EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	RELATIVE	PC+6	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	REG. INDIR.	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC+6										10 e-2
RETURN 'RET'	REGISTER INDIR.	(SP) (SP+1)	C9	D6	D0	C8	C0	E8	E0	F6	F0	
RETURN FROM INT 'RETI'	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REG. INDIR.	(SP) (SP+1)	ED 45									

NOTE—CERTAIN
FLAGS HAVE MORE
THAN ONE PURPOSE.
REFER TO SECTION
6.0 FOR DETAILS

JUMP GROUP

Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments	
		C	Z	P/V	S	N	H	76	543	210					
PC ← nn		•	•	•	•	•	•	11	000	011	3	3	10		
								←	n	→					
								←	n	→					
cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	•	•	•	•	11	cc	010	3	3	10	cc	Condition
								←	n	→				000	NZ non zero
								←	n	→				001	Z zero
								←	n	→				010	NC non carry
								←	n	→				011	C carry
														100	PO parity odd
														101	PE parity even
														110	P sign positive
														111	M sign negative
PC ← PC + e		•	•	•	•	•	•	00	011	000	2	3	12		
								←	e-2	→					
C, e	If C = 0, continue	•	•	•	•	•	•	00	111	000	2	2	7	If condition not met	
								←	e-2	→					
											2	3	12	If condition is met	
NC, e	If C = 1, PC ← PC + e	•	•	•	•	•	•	00	110	000	2	2	7	If condition not met	
								←	e-2	→					
											2	3	12	If condition is met	
Z, e	If Z = 0 continue	•	•	•	•	•	•	00	101	000	2	2	7	If condition not met	
								←	e-2	→					
											2	3	12	If condition is met	
NZ, e	If Z = 1, continue	•	•	•	•	•	•	00	100	000	2	2	7	If condition not met	
								←	e-2	→					
											2	3	12	If condition met	
	If Z = 0, PC ← PC + e														
HL)	PC ← HL	•	•	•	•	•	•	11	101	001	1	1	4		
IX)	PC ← IX	•	•	•	•	•	•	11	011	101	2	2	8		
								11	101	001					
IY)	PC ← IY	•	•	•	•	•	•	11	111	101	2	2	8		
								11	101	001					
Z, e	B ← B-1	•	•	•	•	•	•	00	010	000	2	2	8	If B = 0	
	If B = 0, continue							←	e-2	→					
											2	3	13	If B ≠ 0	
	If B ≠ 0, PC ← PC + e														

es: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <-126, 129>

e-2 in the op-code provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e.

Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,

‡ = flag is affected according to the result of the operation.

JUMP, CALL and RETURN GROUP

			CONDITION									
			UN- COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B=0
JUMP 'JP'	IMMED. EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	RELATIVE	PC+6	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	REG. INDIR.	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC+6										10 e-2
RETURN 'RET'	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0.	
RETURN FROM INT 'RETI'	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REG. INDIR.	(SP) (SP+1)	ED 45									

NOTE—CERTAIN
FLAGS HAVE MORE
THAN ONE PURPOSE.
REFER TO SECTION
6.0 FOR DETAILS

RESTART GROUP

		OP CODE	
C A L L A D D R E S S	0000 _H	C7	'RST 0'
	0008 _H	CF	'RST 8'
	0010 _H	D7	'RST 16'
	0018 _H	DF	'RST 24'
	0020 _H	E7	'RST 32'
	0028 _H	EF	'RST 40'
	0030 _H	F7	'RST 48'
	0038 _H	FF	'RST 56'

CALL AND RETURN GROUP

Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76	543	210				
CALL nn	(SP-1) ← PC _H (SP-2) ← PC _L PC ← nn	•	•	•	•	•	•	11	001	101	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11	cc	100	3	3	10	If cc is false
								←	n	→	3	5	17	If cc is true
								←	n	→				
	PC _L ← (SP) PC _H ← (SP+1)	•	•	•	•	•	•	11	001	001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11	cc	000	1	1	5	If cc is false
											1	3	11	If cc is true
RETURN	Return from interrupt	•	•	•	•	•	•	11	101	101	2	4	14	
								01	001	101				
								11	101	101	2	4	14	
								01	000	101				
								11	t	111	1	3	11	
RETURN p	(SP-1) ← PC _H (SP-2) ← PC _L PC _H ← 0 PC _L ← P	•	•	•	•	•	•	11	t	111	1	3	11	

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
‡ = flag is affected according to the result of the operation.

INPUT GROUP

		SOURCE PORT ADDRESS			
		IMMED.	REG.	INDIR.	
		(n)	(C)		
INPUT DESTINATION	INPUT 'IN'	REG ADDRESSING	A	D8 n	ED 78
			B		ED 40
			C		ED 48
			D		ED 50
			E		ED 58
			H		ED 80
			L		ED 88
			'INI' – INPUT & Inc HL, Dec B	REG. INDIR	(HL)
	'INIR' – INP, Inc HL, Dec B, REPEAT IF B≠0		ED B2		
	'IND' – INPUT & Dec HL, Dec B		ED AA		
	'INDR' – INPUT, Dec HL, Dec B, REPEAT IF B≠0		ED BA		

BLOCK INPUT
COMMANDS

BLOCK INPUT
COMMANDS

OUTPUT GROUP

		SOURCE							
		REGISTER							REG. IND.
		A	B	C	D	E	H	L	(HL)
'OUT'	IMMED.	(n)	D3 n						
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69
'OUTI' – OUTPUT Inc HL, Dec b	REG. IND.	(C)							ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	REG. IND.	(C)							ED B3
'OUTD' – OUTPUT Dec HL & B	REG. IND.	(C)							ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	REG. IND.	(C)							ED BB

BLOCK
OUTPUT
COMMANDS

PORT
DESTINATION
ADDRESS

INPUT AND OUTPUT GROUP

Mnemonic	Symbolic Operation	Flags						Op-Code			No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76	543	210				
IN A, (n)	A ← (n)	•	•	•	•	•	•	11	011	011	2	3	11	n to A ₀ ~ A ₇ Acc to A ₈ ~ A ₁₅
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	•	‡	P	‡	0	‡	11	101	101	2	3	12	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	X	①	X	X	1	X	11	101	101	2	4	16	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	1	X	11	101	101	2	5 (If B ≠ 0)	21	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
								10	110	010	2	4 (If B = 0)	16	
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	X	①	X	X	1	X	11	101	101	2	4	16	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
INDR	(HL) ← (C) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	1	X	11	101	101	2	5 (If B ≠ 0)	21	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
								10	111	010	2	4 (If B = 0)	16	
OUT (n), A	(n) → A	•	•	•	•	•	•	11	010	011	2	3	11	n to A ₀ ~ A ₇ Acc to A ₈ ~ A ₁₅
OUT (C), r	(C) → r	•	•	•	•	•	•	11	101	101	2	3	12	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	X	①	X	X	1	X	11	101	101	2	4	16	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	1	X	11	101	101	2	5 (If B ≠ 0)	21	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
								10	110	011	2	4 (If B = 0)	16	
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	X	①	X	X	1	X	11	101	101	2	4	16	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
OTDR	(C) ← (HL) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	1	X	11	101	101	2	5 (If B ≠ 0)	21	C to A ₀ ~ A ₇ B to A ₈ ~ A ₁₅
								10	111	011	2	4 (If B = 0)	16	

Notes: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ‡ = flag is affected according to the result of the operation.

Der Befehlssatz des Z80

OBJ CODE	QUELLBEFEHL	
8E	ADC	A,(HL)
DD8E05	ADC	A,(IX+d)
FD8E05	ADC	A,(IY+d)
8F	ADC	A,A
88	ADC	A,B
89	ADC	A,C
8A	ADC	A,D
8B	ADC	A,E
8C	ADC	A,H
8D	ADC	A,L
CE20	ADC	A,n
ED4A	ADC	HL,BC
ED5A	ADC	HL,DE
ED6A	ADC	HL,HL
ED7A	ADC	HL,SP
86	ADD	A,(HL)
DD8605	ADD	A,(IX+d)
FD8605	ADD	A,(IY+d)
87	ADD	A,A
80	ADD	A,B
81	ADD	A,C
82	ADD	A,D
83	ADD	A,E
84	ADD	A,H
85	ADD	A,L
C620	ADD	A,n
09	ADD	HL,BC
19	ADD	HL,DE
29	ADD	HL,HL
39	ADD	HL,SP
DD09	ADD	IX,BC
DD19	ADD	IX,DE
DD29	ADD	IX,IX
DD39	ADD	IX,SP
FD09	ADD	IY,BC
FD19	ADD	IY,DE
FD29	ADD	IY,IY
FD39	ADD	IY,SP
A6	AND	(HL)
DDA605	AND	(IX+d)
FDA605	AND	(IY+d)
A7	AND	A
A0	AND	B
A1	AND	C
A2	AND	D
A3	AND	E
A4	AND	H
A5	AND	L

OBJ CODE	QUELLBEFEHL	
E620	AND	n
CB46	BIT	0,(HL)
DDCB0546	BIT	0,(IX+d)
FDCB0546	BIT	0,(IY+d)
CB47	BIT	0,A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB4E	BIT	1,(HL)
DDCB054E	BIT	1,(IX+d)
FDCB054E	BIT	1,(IY+d)
CB4F	BIT	1,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB56	BIT	2,(HL)
DDCB0556	BIT	2,(IX+d)
FDCB0556	BIT	2,(IY+d)
CB57	BIT	2,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB5E	BIT	3,(HL)
DDCB055E	BIT	3,(IX+d)
FDCB055E	BIT	3,(IY+d)
CB5F	BIT	3,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB66	BIT	4,(HL)
DDCB0566	BIT	4,(IX+d)
FDCB0566	BIT	4,(IY+d)
CB67	BIT	4,A
CB60	BIT	4,B
CB61	BIT	4,C
CB62	BIT	4,D

OBJ CODE	QUELLBEFEHL	
CB63	BIT	4,E
CB64	BIT	4,H
CB65	BIT	4,L
CB6E	BIT	5,(HL)
DDCB056E	BIT	5,(IX+d)
FDCB056E	BIT	5,(IY+d)
CB6F	BIT	5,A
CB68	BIT	5,B
CB69	BIT	5,C
CB6A	BIT	5,D
CB6B	BIT	5,E
CB6C	BIT	5,H
CB6D	BIT	5,L
CB76	BIT	6,(HL)
DDCB0576	BIT	6,(IX+d)
FDCB0576	BIT	6,(IY+d)
CB77	BIT	6,A
CB70	BIT	6,B
CB71	BIT	6,C
CB72	BIT	6,D
CB73	BIT	6,E
CB74	BIT	6,H
CB75	BIT	6,L
CB7E	BIT	7,(HL)
DDCB057E	BIT	7,(IX+d)
FDCB057E	BIT	7,(IY+d)
CB7F	BIT	7,A
CB78	BIT	7,B
CB79	BIT	7,C
CB7A	BIT	7,D
CB7B	BIT	7,E
CB7C	BIT	7,H
CB7D	BIT	7,L
DC8405	CALL	C,nn
FC8405	CALL	M,nn
D48405	CALL	NC,nn
C48405	CALL	NZ,nn
F48405	CALL	P,nn
EC8405	CALL	PE,nn
E48405	CALL	PO,nn
CC8405	CALL	Z,nn
CD8405	CALL	nn
3F	CCF	
BE	CP	(HL)
DDBE05	CP	(IX+d)
FDBE05	CP	(IY+d)
BF	CP	A
BB	CP	B
B9	CP	C
BA	CP	D
BB	CP	E
BC	CP	H
BD	CP	L
FE20	CP	n
EDA9	CPD	
EDB9	CPDR	

OBJ CODE	QUELLBEFEHL	
EDB1	CPIR	
EDA1	CPI	
2F	CPL	
27	DAA	
35	DEC	(HL)
DD3505	DEC	(IX+d)
FD3505	DEC	(IY+d)
3D	DEC	A
05	DEC	B
08	DEC	BC
0D	DEC	C
15	DEC	D
18	DEC	DE
1D	DEC	E
25	DEC	H
28	DEC	HL
DD2B	DEC	IX
FD2B	DEC	IY
2D	DEC	L
3B	DEC	SP
F3	DI	
102E	DJNZ	
FB	EI	
E3	EX	(SP),HL
DDE3	EX	(SP),IX
FDE3	EX	(SP),IY
08	EX	AF,AF
EB	EX	DE,HL
09	FXX	
76	HALT	
ED46	IM	0
ED56	IM	1
ED5E	IM	2
ED78	IN	A,C
ED40	IN	B,C
ED48	IN	C,C
ED50	IN	D,C
ED58	IN	E,C
ED60	IN	H,C
ED68	IN	L,C
34	INC	(HL)
DD3405	INC	(IX+d)
FD3405	INC	(IY+d)
3C	INC	A
04	INC	B
03	INC	BC
0C	INC	C
14	INC	D
13	INC	DE
1C	INC	E
24	INC	H
23	INC	HL
DD23	INC	IX
FD23	INC	IY
2C	INC	L
33	INC	SP
DB20	IN	A (n)

OBJ CODE	QUELLBEFEHL	
EDAA	IND	
EDBA	INDR	
EDA2	INI	
EDB2	INIR	
C38405	JP	nn
E9	JP	(HL)
DDE9	JP	(IX)
FDE9	JP	(IY)
DA8405	JP	C,nn
FA8405	JP	M,nn
D28405	JP	NC,nn
C28405	JP	NZ,nn
F28405	JP	P,nn
EA8405	JP	PE,nn
E28405	JP	PO,nn
CA8405	JP	Z,nn
382E	JR	C,e
302E	JR	NC,e
202E	JR	NZ,e
282E	JR	Z,e
182E	JR	e
02	LD	(BC),A
12	LD	(DE),A
77	LD	(HL),A
70	LD	(HL),B
71	LD	(HL),C
72	LD	(HL),D
73	LD	(HL),E
74	LD	(HL),H
75	LD	(HL),L
3620	LD	(HL),n
DD7705	LD	(IX+d),A
DD7005	LD	(IX+d),B
DD7105	LD	(IX+d),C
DD7205	LD	(IX+d),D
DD7305	LD	(IX+d),E
DD7405	LD	(IX+d),H
DD7505	LD	(IX+d),L
DD360520	LD	(IX+d),n
FD7705	LD	(IY+d),A
FD7005	LD	(IY+d),B
FD7105	LD	(IY+d),C
FD7205	LD	(IY+d),D
FD7305	LD	(IY+d),E
FD7405	LD	(IY+d),H
FD7505	LD	(IY+d),L
FD360520	LD	(IY+d),n
328405	LD	(nn),A
ED438405	LD	(nn),BC
ED538405	LD	(nn),DE
228405	LD	(nn),HL
DD228405	LD	(nn),IX
FD228405	LD	(nn),IY
ED738405	LD	(nn),SP
0A	LD	A,(BC)
1A	LD	A,(DE)
7E	LD	A,(HL)

OBJ CODE	QUELLBEFEHL	
DD7E05	LD	A,(IX+d)
FD7E05	LD	A,(IY+d)
3A8405	LD	A,(nn)
7F	LD	A,A
78	LD	A,B
79	LD	A,C
7A	LD	A,D
7B	LD	A,E
7C	LD	A,H
ED57	LD	A,I
7D	LD	A,L
3E20	LD	A,n
ED5F	LD	A,R
46	LD	B,(HL)
DD4605	LD	B,(IX+d)
FD4605	LD	B,(IY+d)
47	LD	B,A
40	LD	B,B
41	LD	B,C
42	LD	B,D
43	LD	B,E
44	LD	B,H
45	LD	B,L
0620	LD	B,n
ED488405	LD	BC,(nn)
018405	LD	BC,nn
4E	LD	C,(HL)
DD4E05	LD	C,(IX+d)
FD4E05	LD	C,(IY+d)
4F	LD	C,A
48	LD	C,B
49	LD	C,C
4A	LD	C,D
4B	LD	C,E
4C	LD	C,H
4D	LD	C,L
0E20	LD	C,n
56	LD	D,(HL)
DD5605	LD	D,(IX+d)
FD5605	LD	D,(IY+d)
57	LD	D,A
50	LD	D,B
51	LD	D,C
52	LD	D,D
53	LD	D,E
54	LD	D,H
55	LD	D,L
1620	LD	D,n
ED588405	LD	DE,(nn)
118405	LD	DE,nn
5E	LD	E,(HL)
DD5E05	LD	E,(IX+d)
FD5E05	LD	E,(IY+d)
5F	LD	E,A
58	LD	E,B
59	LD	E,C
5A	LD	E,D

OBJ CODE	QUELLBEFEHL	
5B	LD	E,E
5C	LD	E,H
5D	LD	E,L
1E20	LD	E,n
66	LD	H,(HL)
DD6605	LD	H,(IX+d)
FD6605	LD	H,(IY+d)
67	LD	H,A
60	LD	H,B
61	LD	H,C
62	LD	H,D
63	LD	H,E
64	LD	H,H
65	LD	H,L
2620	LD	H,n
2A8405	LD	HL,(nn)
218405	LD	HL,nn
ED47	LD	I,A
DD2A8405	LD	IX,(nn)
DD218405	LD	IX,nn
FD2A8405	LD	IY,(nn)
FD218405	LD	IY,nn
6E	LD	L,(HL)
DD6E05	LD	L,(IX+d)
FD6E05	LD	L,(IY+d)
6F	LD	L,A
68	LD	L,B
69	LD	L,C
6A	LD	L,D
6B	LD	L,E
6C	LD	L,H
6D	LD	L,L
2E20	LD	L,n
ED4F	LD	R,A
ED7B8405	LD	SP,(nn)
F9	LD	SP,HL
DDF9	LD	SP,IX
FDf9	LD	SP,IY
318405	LD	SP,nn
EDA8	LDD	
ED88	LDDR	
EDA0	LDI	
ED80	LDIR	
ED44	NEG	
00	NOP	
B6	OR	(HL)
DD8605	OR	(IX+d)
FD8605	OR	(IY+d)
B7	OR	A
B0	OR	B
B1	OR	C
B2	OR	D
B3	OR	E
B4	OR	H
B5	OR	L
F620	OR	n
ED88	OTDR	

OBJ CODE	QUELLBEFEHL	
ED83	OTIR	
ED79	OUT	(C),A
ED41	OUT	(C),B
ED49	OUT	(C),C
ED51	OUT	(C),D
ED59	OUT	(C),E
ED61	OUT	(C),H
ED69	OUT	(C),L
D320	OUT	(n),A
EDAB	OUTD	
EDA3	OUTI	
F1	POP	AF
C1	POP	BC
D1	POP	DE
E1	POP	HL
DDE1	POP	IX
FDE1	POP	IY
F5	PUSH	AF
C5	PUSH	BC
D5	PUSH	DE
E5	PUSH	HL
DDE5	PUSH	IX
FDE5	PUSH	IY
CB86	RES	0,(HL)
DDCB0586	RES	0,(IX+d)
FDCB0586	RES	0,(IY+d)
CB87	RES	0,A
CB80	RES	0,B
CB81	RES	0,C
CB82	RES	0,D
CB83	RES	0,E
CB84	RES	0,H
CB85	RES	0,L
CB8E	RES	1,(HL)
DDCB058E	RES	1,(IX+d)
FDCB058E	RES	1,(IY+d)
CB8F	RES	1,A
CB88	RES	1,B
CB89	RES	1,C
CB8A	RES	1,D
CB8B	RES	1,E
CB8C	RES	1,H
CB8D	RES	1,L
CB96	RES	2,(HL)
DDCB0596	RES	2,(IX+d)
FDCB0596	RES	2,(IY+d)
CB97	RES	2,A
CB90	RES	2,B
CB91	RES	2,C
CB92	RES	2,D
CB93	RES	2,E
CB94	RES	2,H
CB95	RES	2,L
CB9E	RES	3,(HL)
DDCB059E	RES	3,(IX+d)
FDCB059E	RES	3,(IY+d)

OBJ CODE	QUELLBEFEHL	
CB9F	RES	3,A
CB98	RES	3,B
CB99	RES	3,C
CB9A	RES	3,D
CB9B	RES	3,E
CB9C	RES	3,H
CB9D	RES	3,L
CBA6	RES	4,(HL)
DDCB05A6	RES	4,(IX+d)
FDCB05A6	RES	4,(IY+d)
CBA7	RES	4,A
CBA0	RES	4,B
CBA1	RES	4,C
CBA2	RES	4,D
DBA3	RES	4,E
CBA4	RES	4,H
CBA5	RES	4,L
CBAE	RES	5,(HL)
DDCB05AE	RES	5,(IX+d)
FDCB05AE	RES	5,(IY+d)
CBAF	RES	5,A
CBA8	RES	5,B
CBA9	RES	5,C
CBA A	RES	5,D
CBA8	RES	5,E
CBAC	RES	5,H
CBAD	RES	5,L
CB86	RES	6,(HL)
DDCB05B6	RES	6,(IX+d)
FDCB05B6	RES	6,(IY+d)
CB87	RES	6,A
CB80	RES	6,B
CB81	RES	6,C
CB82	RES	6,D
CB83	RES	6,E
CB84	RES	6,H
CB85	RES	6,L
CB8E	RES	7,(HL)
DDCB05BE	RES	7,(IX+d)
FDCB05BE	RES	7,(IY+d)
CB8F	RES	7,A
CB88	RES	7,B
CB89	RES	7,C
CB8A	RES	7,D
CB8B	RES	7,E
CB8C	RES	7,H
CB8D	RES	7,L
C9	RET	
D8	RET	C
F8	RET	M
D0	RET	NC
C0	RET	NZ
F0	RET	P
E8	RET	PE
E0	RET	PO
C8	RET	Z

OBJ CODE	QUELLBEFEHL	
ED4D	RETI	
ED45	RETN	
CB16	RL	(HL)
DDCB0516	RL	(IX+d)
FDCB0516	RL	(IY+d)
CB17	RL	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
17	RLA	
CB06	RLC	(HL)
DDCB0506	RLC	(IX+d)
FDCB0506	RLC	(IY+d)
CB07	RLC	A
CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
07	RLCA	
ED6F	RLD	
CB1E	RR	(HL)
DDCB051E	RR	(IX+d)
FDCB051E	RR	(IY+d)
CB1F	RR	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
1F	RRA	
CB0E	RRC	(HL)
DDCB050E	RRC	(IX+d)
FDCB050E	RRC	(IY+d)
CB0F	RRC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
0F	RRCA	
ED67	RRD	
C7	RST	00H
CF	RST	08H
D7	RST	10H
DF	RST	18H
E7	RST	20H
EF	RST	28H
F7	RST	30H
FF	RST	38H
DE20	SBC	A,n

OBJ CODE	QUELLBEFEHL	
9E	SBC	A,(HL)
DD9E05	SBC	A,(IX+d)
FD9E05	SBC	A,(IY+d)
9F	SBC	A,A
98	SBC	A,B
99	SBC	A,C
9A	SBC	A,D
9B	SBC	A,E
9C	SBC	A,H
9D	SBC	A,L
ED42	SBC	HL,BC
ED52	SBC	HL,DE
ED62	SBC	HL,HL
ED72	SBC	HL,SP
37	SCF	
CBC6	SET	0,(HL)
DDCB05C6	SET	0,(IX+d)
FDCB05C6	SET	0,(IY+d)
CBC7	SET	0,A
CBC0	SET	0,B
CBC1	SET	0,C
CBC2	SET	0,D
CBC3	SET	0,E
CBC4	SET	0,H
CBC5	SET	0,L
CBCF	SET	1,(HL)
DDCB05CF	SET	1,(IX+d)
FDCB05CF	SET	1,(IY+d)
CBCF	SET	1,A
CBC8	SET	1,B
CBC9	SET	1,C
CBCA	SET	1,D
CBCB	SET	1,E
CBCC	SET	1,H
CBCD	SET	1,L
CBDE	SET	2,(HL)
DDCB05D6	SET	2,(IX+d)
FDCB05D6	SET	2,(IY+d)
CBDF	SET	2,A
CBDO	SET	2,B
CBDI	SET	2,C
CBDE	SET	2,D
CBDE	SET	2,E
CBDE	SET	2,H
CBDE	SET	2,L
CBDE	SET	3,B
CBDE	SET	3,(HL)
DDCB05DE	SET	3,(IX+d)
FDCB05DE	SET	3,(IY+d)
CBDF	SET	3,A
CBDF	SET	3,C
CBDA	SET	3,D
CBDB	SET	3,E
CBDC	SET	3,H
CBDD	SET	3,L
CBE6	SET	4,(HL)

OBJ CODE	QUELLBEFEHL	
DDCB05E6	SET	4,(IX+d)
FDCB05E6	SET	4,(IY+d)
CBE7	SET	4,A
CBE0	SET	4,B
CBE1	SET	4,C
CBE2	SET	4,D
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBE6	SET	5,(HL)
DDCB05EE	SET	5,(IX+d)
FDCB05EE	SET	5,(IY+d)
CBEF	SET	5,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBF6	SET	6,(HL)
DDCB05F6	SET	6,(IX+d)
FDCB05F6	SET	6,(IY+d)
CBF7	SET	6,A
CBF0	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBFE	SET	7,(HL)
DDCB05FE	SET	7,(IX+d)
FDCB05FE	SET	7,(IY+d)
CBFF	SET	7,A
CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CB26	SLA	(HL)
DDCB0526	SLA	(IX+d)
FDCB0526	SLA	(IY+d)
CB27	SLA	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB2E	SRA	(HL)
DDCB052E	SRA	(IX+d)
FDCB052E	SRA	(IY+d)
CB2F	SRA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D

OBJ CODE		QUELLBEFEHL
CB2B	SRA	E
CB2C	SRA	H
CB2D	SRA	L
CB3E	SRL	(HL)
DDCB053E	SRL	(IX+d)
FDCB053E	SRL	(IY+d)
CB3F	SRL	A
CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
96	SUB	(HL)
DD9605	SUB	(IX+d)
FD9605	SUB	(IY+d)
97	SUB	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
D620	SUB	n
AE	XOR	(HL)
DDAE05	XOR	(IX+d)
FDAE05	XOR	(IY+d)
AF	XOR	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
EE20	XOR	n

(Courtesy of Zilog Inc.)

Äquivalente Befehle: Z80-8080

Z80	8080	Z80	8080	Z80	8080
ADC A, (HL)	ADC M	EX (SP), HL	XTHL	OR n	ORI [B2]
ADC A, n	ACI [B2]	HALT	HLT	OR r	ORA r
ADC A, r	ADC r	IN A, (n)	IN [B2]	OR (HL)	ORA M
ADD A, (HL)	ADD M	INC BC	INX B	OUT (n), A	OUT [B2]
ADD A, n	ADI [B2]	INC DE	INX D	POP AF	POP PSW
ADD A, r	ADD r	INC HL	INX H	POP BC	POP B
ADD HL, BC	DAD B	INC r	INR r	POP DE	POP D
ADD HL, DE	DAD D	INC SP	INX SP	POP HL	POP H
ADD HL, HL	DAD H	INC (HL)	INR M	PUSH AF	PUSH PSW
ADD HL, SP	DAD SP	JP C, nn	JC [B2] [B3]	PUSH BC	PUSH B
AND n	ANI [B2]	JP M, nn	JM [B2] [B3]	PUSH DE	PUSH D
AND r	ANA r	JP NC, nn	JNC [B2] [B3]	PUSH HL	PUSH H
AND (HL)	ANA M	JP nn	JMP [B2] [B3]	RET	RET
CALL C, nn	CC [B2] [B3]	JP NZ, nn	JNZ [B2] [B3]	RET C	RC
CALL M, nn	CM [B2] [B3]	JP P, nn	JP [B2] [B3]	RET M	RM
CALL NC, nn	CNC [B2] [B3]	JP PE, nn	JPE [B2] [B3]	RET NC	RNC
CALL nn	CALL	JP PO, nn	JPO [B2] [B3]	RET NZ	RNZ
CALL NZ, nn	CNZ [B2] [B3]	JP Z, nn	JZ [B2] [B3]	RET P	RP
CALL P, nn	CP [B2] [B3]	JP (HL)	PCHL	RET PE	RPE
CALL PE, nn	CPE [B2] [B3]	LD A, (DE)	LDAX	RET PO	RPO
CALL PO, nn	CPO [B2] [B3]	LDA, (nn),	LDA [B2] [B3]	RET Z	RZ
CALL Z, nn	CZ [B2] [B3]	LD DE, nn	LXID, [B2] [B3]	RLA	RAL
CCF	CMC	LD SP, nn	LXI SP, [B2] [B3]	RLCA	RLC
CP r	CMP r	LD (BC), A	STAX B	RRA	RAR
CP (HL)	CMP M	LD (DE), A	STAX D	RRCA	RRC
CPL	CMA	LD (HL), r	MOV M, r	RST P	RST P
CP n	CPI [B2]	LD (nn), A	STA [B2] [B3]	SBC A, (HL)	SBB M
DAA	DAA	LD (nn), HL	SHLD [B2] [B3]	SBC A, n	SBI [B2]
DEC BC	DCX B	LD A, (BC)	LDAX B	SBC A, r	SBB r
DEC DE	DCX D	LD BC, nn	LXIB, [B2] [B3]	SCF	STC
DEC HL	DCX H	LD HL, (nn)	LHLD [B2] [B3]	SUB n	SUI [B2]
DEC r	DCR r	LD HL, nn	LXI H [B2] [B3]	SUB r	SUB r
DEC SP	DCX SP	LD r, (HC)	MOV I, M	SUB (HL)	SUB M
DEC (HL)	DCR M	LD r, n	MVI r, [B2]	XOR n	XRI [B2]
DI	DI	LD r, r'	MOV r1, r2	XOR r	XRA r
EI	EI	LD SP, HL	SPHL	XOR (HL)	XRAM
EX DE, HL	XCHG	NOP	NOP		

Äquivalente Befehle: 8080–Z80

8080	Z80	8080	Z80	8080	Z80
ACI [B2]	ADC A, n	IN [B2]	IN A, (n)	POP H	POP HL
ADC M	ADC A, (HL)	INR M	INC (HL)	POP PSW	POP AF
ADC r	ADC A, r	INR r	INC r	PUSH B	PUSH BC
ADD M	ADD A, (HL)	INX B	INC BC	PUSH D	PUSH DE
ADD r	ADD A, r	INX D	INC DE	PUSH H	PUSH HL
ADI [B2]	ADD A, n	INX H	INC HL	PUSH PSW	PUSH AF
ANA M	AND (HL)	INX SP	INC SP	RAL	RLA
ANA r	AND r	JC [B2] [B3]	JP C, nn	RAR	RRA
ANI [B2]	AND n	JM [B2] [B3]	JP M, nn	RC	RET C
CALL	CALL nn	JMP [B2] [B3]	JP nn	RET	RET
CC [B2] [B3]	CALL C, nn	JNC [B2] [B3]	JP NC, nn	RLC	RLCA
CM [B2] [B3]	CALL M, nn	JNZ [B2] [B3]	JP NZ, nn	RM	RET M
CMA	CPL	JP [B2] [B3]	JP P, nn	RNC	RET NC
CMC	CCF	JPE [B2] [B3]	JP PE, nn	RNZ	RET NZ
CMP M	CP (HL)	JPO [B2] [B3]	JP PO, nn	RP	RET P
CMP r	CP r	JZ [B2] [B3]	JP Z, nn	RPE	RET PE
CNC [B2] [B3]	CALL NC, nn	LDA [B2] [B3]	LD A, (nn)	RPO	RET PO
CNZ [B2] [B3]	CALL NZ, nn	LDAX B	LD A, (BC)	RRC	RRCA
CP [B2] [B3]	CALL P, nn	LDAX D	LD A, (DE)	RST	RST P
CPE [B2] [B3]	CALL PE, nn	LHLD [B2] [B3]	LD HL, (nn)	RZ	RET Z
CPI [B2]	CP n	LXI B [B2] [B3]	LD BC, nn	SBB M	SBC A, (HL)
CPO [B2] [B3]	CALL PO, nn	LDID [B2] [B3]	LD DE, nn	SBB r	SBC A, r
CZ [B2] [B3]	CALL Z, nn	LXI H [B2] [B3]	LD HL, nn	SBI [B2]	SBC A, n
DAA	DAA	LXI SP [B2] [B3]	LD SP, nn	SHLD [B2] [B3]	LD (nn), HL
DAD B	ADD HL, BC	MOV M, r	LD (HL), r	SPHL	LD SP, HL
DAD D	ADD HL, DE	MOV r, M	LD r, (HL)	STA [B2] [B3]	LD (nn), A
DAD H	ADD HL, HL	MOV r1 r2	LD r, r1	STAX B	LD (BC), A
DAD SP	ADD HL, SP	MVI M	LD (HL), n	STAX D	LD (DE), A
DCR M	DEC (HL)	MVI r [B2]	LD r, n	STC	SCF
DCR r	DEC r	NOP	NOP	SUB M	SUB (HL)
DCX B	DEC BC	ORA M	OR (HL)	SUB r	SUB r
DCX D	DEC DE	ORA r	OR r	SUI [B2]	SUB n
DCX H	DEC HL	ORI [B2]	OR n	XCHG	EX DE, HL
DCX SP	DEC SP	OUT [B2]	OUT (n), A	XRA M	XOR (HL)
DI	DI	PCHL	JP (HL)	XRA r	XOR r
EI	EI	POP B	POP BC	XRI [B2]	XOR n
HALT	HLT	POP D	POP DE	XTHL	EX (SP), HL

Z80 – CPU INTERRUPT STRUCTURE

MASKABLE (INT)

Mode 0

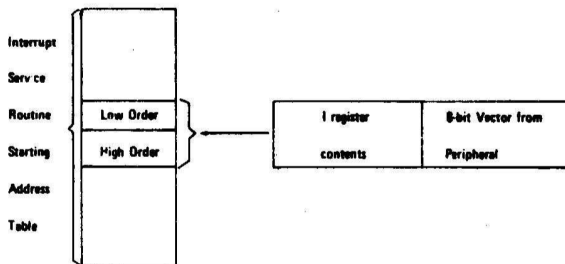
Place instruction onto Data Bus during $\overline{INTA} = \overline{MI} = \overline{IORQ}$ like 8080A

Mode 1

Restart to $38H$ or $56H$ ('RST 56')

Mode 2

Used by Z80 Peripherals



NON MASKABLE (NMI)

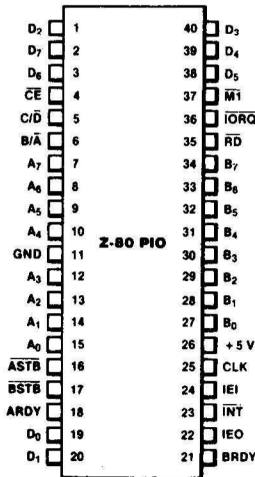
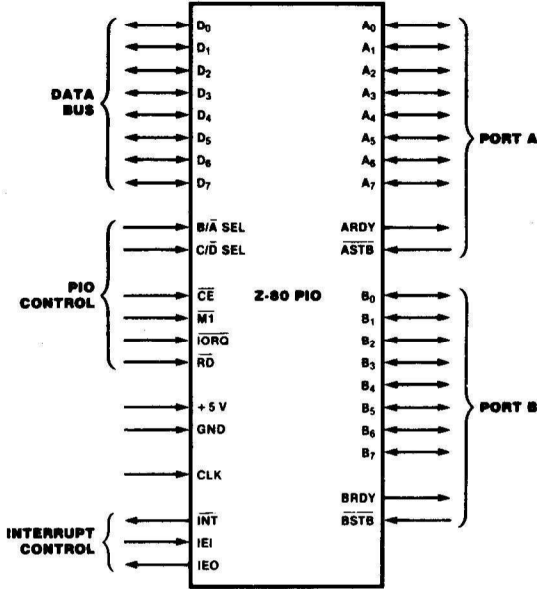
Restart to $66H$ or $102H$

INTERRUPT ENABLE/DISABLE FLIP-FLOPS

Action	IFF ₁	IFF ₂	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	IFF ₂ → Parity flag
LD A, R	•	•	IFF ₂ → Parity flag
Accept $\overline{NMI}$	0	IFF ₁	$\overline{NMI} \rightarrow IFF_2$
RETN	IFF ₂	•	IFF ₂ → IFF ₁
Accept $\overline{INT}$	0	0	
RETI	•	•	

• • indicates no change

Z80-PIO PIN ASSIGNMENT



CTC PROGRAMMING SUMMARY

REGISTER SELECTION

SELECT LINES		
C/D	B/A	
0	0	A Data
0	1	B Data
1	0	A Control
	1	B Control

LOAD INTERRUPT VECTOR

D7				D0				
V7	V6	V5	V4	V3	V2	V1	0	
								Control Register

SET OPERATING MODE

D7				D0				
M1	M0	X	X	1	1	1	1	
								Control Register

Mode Number	M1	M0	Mode
0	0	0	Output
1	0	1	Input
2	1	0	Bidirectional
3	1	1	Bit Control

If Mode 3 selected, the next control word is

D7				D0				
I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0	
								Control Register

I/O = 1 Sets bit to Input

I/O = 0 Sets bit to Output

SET INTERRUPT CONTROL

D7				D0				
Int Enable	AND/OR	High/Low	Mask Follows	0	1	1	1	
								Control Register

In Mode 3 if Mask follows = 1, the next control word is

D7				D0				
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0	
								Control Register

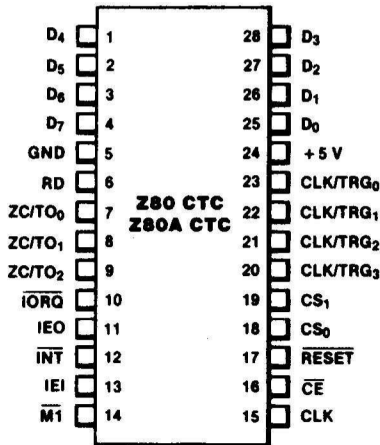
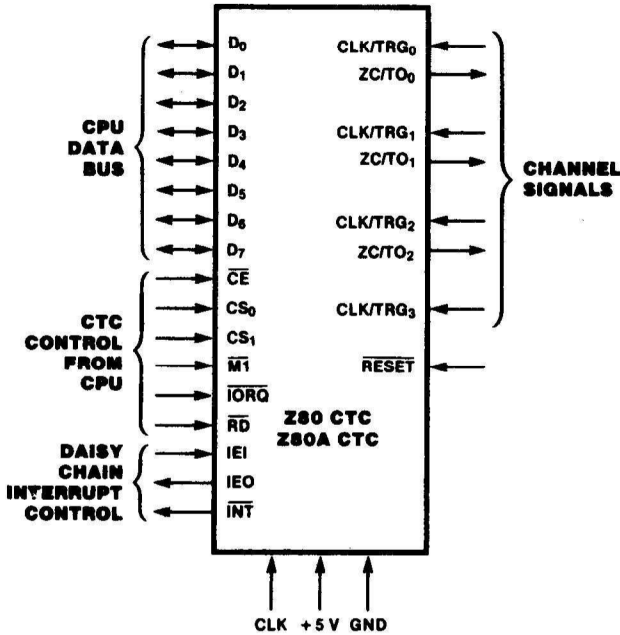
MB = 0 Monitor the bit

MB = 1 Mask the bit

ENABLE / DISABLE INTERRUPTS

D7				D0				
Int Enable	X	X	X	0	0	1	1	
								Control Register

3.0 CTC PIN DESCRIPTION



SELECT LINES		CHANNEL SELECTED	PRIORITY
CS ₁	CS ₀		
0	0	0	Highest
0	1	1	
1	0	2	
1	1	3	Lowest

READ = DOWN COUNTER

WAITE = CONTROL REGISTER

LOAD INTERRUPT VECTOR

$$CS_0 = CS_1 = 0$$

D7						D0	
V7	V6	V5	V4	V3	X	X	0

Control Register

XX is the binary equivalent of interrupting channel number

SET OPERATING MODE

D7		Timer Mode only				D0	
Interrupt Enable	Mode	Range	Slope	Trigger	Load Time Constant	Resct	1
	Counter/Timer	256/16	+/-	On/Off			

Control Register

If Load Time Constant = 1 the next control word is the Time Constant:

D7							D0
TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0

CTC Channel interrupts when 01H is decremented to 00H

Time Content

Decimal counts to interrupt

01H

1

•

•

FFH

255

00H

256

ANHANG D

Reference Books:

1. Z80 Assembly Language programming Manual.
2. Intel Component Data Catalog.
3. The TTL handbook.
4. Programmierung des Z80
Deutsch von Sybex, Zaks

MICROPROFESSOR

Handbuch für den
MPF - I

2. Teil: Source Listing Monitor Program

(Wird mit dem Gerät im Original mitgeliefert)

MICROPROFESSOR

Handbuch für den
MPF - I

3. Teil: Einführung
in die Programmierung
des Z 80

Microcomputerprogrammierung

Ein Microcomputerprogramm besteht aus einer Anzahl von Befehlen, die von der CPU abgearbeitet werden, um ein bestimmtes logisches oder arithmetisches Resultat zu erhalten.

Ein Programm, das der Computer ausführen soll, muß im Speicher abgelegt sein, damit die CPU auf die Befehle und Daten zugreifen kann. Im Speicher ist das Programm in binärer Form, d. h. als eine Folge von Einsen und Nullen eingeschrieben, jeweils ein 8-bit-Wort an einer Speicherstelle. Jeder Speicherstelle ist eine Adresse zugeordnet.

Der Inhalt einer Speicherstelle läßt sich beim MPF-1 dadurch auslesen, daß man die Taste -ADDR- betätigt und anschließend die gewünschte Adresse eingibt. Im Datenfeld erscheint nun der Inhalt des angesprochenen Speicherplatzes in hexadezimaler Form als 2 Hexziffern. Diese sind leichter zu lesen und zu überblicken als die entsprechende Folge von 8 Einzelbits 00111110B=3EH. Trotzdem ist es sehr schwierig, ein Programm, das nur aus 2stelligen Hexzahlen besteht, also in Maschinensprache geschrieben ist, zu identifizieren und zu verstehen. Daher faßt man die einzelnen Befehle eines Microprozessors in Gruppen zusammen und bezeichnet sie mit Symbolen, die die Funktion des Befehls ausdrücken.

01111101 B = 7D H = LD A,L

LD A,L bezeichnet man als mnemonische, d. h. merkfähige, symbolische Schreibweise. Der angegebene Befehl bedeutet: Lade (engl. load) das Register A mit dem Inhalt des Registers L. Ein Programm, das in dieser mnemonischen Schreibweise abgefaßt ist, bezeichnet man als Programm in Assembler-Sprache. Spezielle Software ist in der Lage, solche Assemblerprogramme in die Maschinensprache zu übertragen. Ein solches Software-Programm heißt "Assembler". Diese sind jedoch relativ teuer und benötigen viel Speicherplatz. Beim MPF-1 muß daher das Assemblieren "von Hand" mit Hilfe einer Befehlsliste des Z80 erfolgen.

In einer Programm-Übersicht sind dann die einzelnen Eintragungen wie folgt angeordnet:

1800 7D LD A,L ;Text

Links ist die Speicheradresse eingetragen, dann folgt der Befehl in Maschinensprache und schließlich der Assembler-Code in der mnemonischen Schreibweise. Durch ein Semicolon abgetrennt, kann dann noch eine Erläuterung angefügt werden.

Beschreibung der Befehlsgruppen des Z80

(1) 8-BIT-LADEBEFEHLE

Alle Ladebefehle sind an der Buchstabenkombination LD zu erkennen. Danach folgt, abgetrennt durch einen Zwischenraum, das Ziel des Ladebefehls und dann, durch ein Komma abgetrennt, die Quellenangabe.

Beispiele:

78H LD A,B

Lade das Register A, den Accumulator, mit dem Inhalt des Registers B. Bei dieser Art von Befehlen, bei denen es um Registerinhalte geht, spricht man von Registeradressierung.

3E XX LD A,n

Lade den Accumulator mit einem bestimmten Wert n, wobei n eine zweiziffrige Hexadezimalzahl ist; dies ist durch XX im Maschinenbefehl angedeutet. Hier spricht man von unmittelbarer Adressierung (engl. immediate). Der gewünschte Wert wird unmittelbar eingegeben.

5E LD E,(HL)

Lade das Register E mit dem Inhalt der Speicherstelle, deren Adresse im Registerpaar HL abgelegt ist. Dabei handelt es sich um eine Kombination von 2 Registern zur Aufnahme eines 16-bit-Wortes, das hier als Adresse verstanden wird. Das 8-bit-Wort, das an dem Speicherplatz mit dieser Adresse steht, soll in das Register E geladen werden.

Beispiel: Im H-Register steht 18H, im L-Register 50H; damit zeigt das HL-Registerpaar auf den Speicherplatz mit der Adresse 1850H. Dort sei der Wert 3AH eingetragen. Nach Ausführung des Befehls 5E steht der Wert 3A auch im Register E. Bei solchen Befehlen spricht man von indirekter Adressierung. Die Registerpaare BC und DE können ebenfalls zur indirekten Adressierung verwendet werden; jedoch läßt sich von diesen aus nur der Accumulator laden, nicht die anderen Register.

3A XX XX LD A,(nn)

Lade den Accumulator -und nur dies ist möglich- mit dem Inhalt der Speicherstelle mit der Adresse nn, ausgedrückt durch eine 4-ziffrige Hexadezimalzahl XXXX. Dabei muß zuerst das niederwertige Byte und danach das höherwertige Byte der Adresse angegeben werden.

Beispiel: 3A 33 19 LD A,1933

33 ist das niederwertige Byte, 19 das höherwertige. Durch diesen 3-Byte-Befehl wird der Accumulator mit dem Inhalt des Speicherplatzes 1933 geladen.

Man nennt dieses Verfahren absolute oder direkte Adressierung.

02 LD (BC),A

Lade den Inhalt des Accumulators in den Speicherplatz, dessen Adresse im Registerpaar BC abgelegt ist.

32 XX XX LD (nn),A

Lade den Inhalt des Accumulators in den Speicherplatz mit der Adresse nn, ausgedrückt durch die 4-ziffrige Hexadezimalzahl XXXX.

Beispiel: Im Accumulator steht FF; der 3-Byte-Befehl

32 52 1A LD (1A 52),A

bewirkt, daß der Wert FF an der Speicherstelle mit der Adresse 1A52 abgelegt wird. - Zu beachten ist, daß auch hier wieder in der Maschinensprache zuerst das niederwertige - lower - Byte und dann das höherwertige - higher - Byte geschrieben werden muß.

Indizierte Adressierung

Für die indizierte Adressierung stehen die Indexregister IX und IY zur Verfügung. Mit Hilfe dieser 16-bit-Register lassen sich manche komplizierte Aufgaben elegant lösen; z. B. das Lesen von Tabellen. Das Indexregister enthält dann die Anfangsadresse der Tabelle. Mit Hilfe des Befehls

DD 7E 03 LD A,(IX+3)

kann man den 4. Tabellenwert (IX+3) in den Accumulator laden. Beispiel: Die Tabelle beginnt bei 1A50; dieser Wert ist Inhalt des Registers IX. An der Speicherstelle 1A53 steht C5. Der Maschinensprache-Befehl DD 7E 03 lädt den Accumulator mit C5. Auf diese Weise können alle Register indiziert über IX oder IY plus Offset geladen werden.

FD 36 XX XX LD (IY+d),n

Lade den Wert n, ein 8-bit-Wort, an die Speicherstelle mit der Adresse IY+d. Beispiel: IY zeigt auf die Anfangsadresse 1900;

FD 36 10 77 LD (IY+10),77

lädt den Wert 77H unmittelbar in die Speicherstelle mit der Adresse 1910.

(2) 16-BIT-LADEBEFEHLE

Die Z80-CPU enthält mehrere 16-bit-Register bzw. Registerpaare: IX, IY, SP und BC, DE, HL. Auf die Stack-Operationen LD SP, PUSH und POP sei an dieser Stelle noch nicht eingegangen.

Beispiele:

21 XX XX LD HL,nn

Lade das HL-Register mit dem 16-bit-Wort nn, ausgedrückt durch die 4ziffrige Hexzahl XXXX.

ED 5B XX XX LD DE,(nn)

Lade das Registerpaar DE mit dem Inhalt der Speicherstelle mit den Adressen nn und nn+1, wobei der Inhalt des Speicherplatzes nn in das Register E, und der Inhalt des Speicherplatzes nn+1 in das Register D geladen wird.

Beispiel: Unter der Adresse 1900 sei 11H, unter 1901 sei 22H gespeichert.
Die Ausführung des Befehls

ED 5B 00 19 LD DE,(1900)

lädt das Register E mit 11H und Register D mit 22H. Das bedeutet, der Inhalt der niedrigeren Adresse wird ins E-Register, der Inhalt der höheren Adresse ins D-Register geladen.

Exchange-Befehle

Exchange bedeutet Austausch: Es werden die Inhalte der Register untereinander ausgetauscht.

EB EX DE,HL

Schreibt die Daten des HL-Registerpaares ins Registerpaar DE und umgekehrt.

D9 EXX

Die Inhalte der allgemeinen Register BC, DE, HL werden gegen die Inhalte des Zweitregister-Blocks BC', DE', HL' ausgetauscht.

08 EX AF,AF'

Mit diesem Befehl werden auch die Inhalte von AF und AF' gegeneinander ausgetauscht.

Programmierübung 1 – Ladebefehle

Lineare Programmierung

8-bit-Ladebefehle

Aufgabe: Es soll ein Programm erstellt werden, das die Register wie folgt lädt:
A=0; B=1; C=2; D=3; E=4; H=5; L=6.
Benutzen Sie die 8-bit-Ladebefehle!

- (1) Schreiben Sie zunächst das Assembler-Programm in mnemonischer Schreibweise. Das Programm muß abgeschlossen werden mit einem RST 38H-Befehl, damit der MPF-1 nach der Abarbeitung des Benutzerprogramms zu seinem Monitor zurückkehrt.
- (2) Benutzen Sie die Übersicht über den Befehlsatz des Z80, um das Programm in die Maschinensprache zu übersetzen. Wählen Sie als Anfangsadresse 1800H und ordnen Sie jeder Befehlszeile die korrekte Speicheradresse zu.
- (3) Geben Sie das Programm in den MPF-1 ein: Betätigen Sie zunächst die Tasten —RS— und —PC—; es erscheint die Adresse 1800H, d. h. die niedrigste RAM-Adresse. Geben Sie jetzt das erste Byte Ihres Programms ein. Danach eröffnen Sie mit der Taste —+— die nächste Speicheradresse 1801H, um das nächste Byte einzugeben, usw.
- (4) Überprüfen Sie zum Schluß die korrekte Eingabe Ihres Programms, indem Sie mit den Tasten —RS— und —PC— zum Anfang zurückkehren und mit —+— alle Bytes durchtesten.
- (5) Setzen Sie den Programmzähler mit der Taste —PC— wieder auf die Anfangsadresse und starten Sie Ihr Programm mit der Taste —GO—.
- (6) Überprüfen SIE dann mit Hilfe der Taste —REG— die Inhalte der Register A bis L, außer F.

Programmierübung 1 A – Ladebefehle

Lineare Programmierung

16-bit-Ladebefehle

Aufgabe: Schreiben Sie ein Programm, das die Register mit folgenden Hexzahlen lädt: B=12H; C=34H; D=56H; E=78H; H=9H; L=AH
Benutzen Sie die 16-bit-Ladebefehle, um jeweils ein Registerpaar zu laden.

Die Startadresse dieses neuen Programms sei die Speicherstelle 1820H. Betätigen Sie die Taste —ADDR— und geben Sie dann 1820 ein. Um auf dieser Adresse das erste Byte Ihres Programms abzuspeichern, müssen Sie jedoch unbedingt erst die Taste —DATA— betätigen, da sonst durch eine Eingabe die Adresse verändert würde.

Weiterhin ist zu beachten, daß bei den 16-bit-Ladebefehlen in der Maschinensprache hinter dem Befehlswort zunächst das niederwertige und dann das höherwertige Byte folgen muß.

Beispiel: 013412 LD BC,1234H

01 ist das Befehlswort

34 ist das niederwertige Byte

12 ist das höherwertige Byte des Operanden

Verfahren Sie bei dieser Aufgabe ebenso wie bei dem ersten Programmierbeispiel: Schritte 1 bis 6.

Speicher- platz	Befehlswort + Operand	Assembler- Code
1800	3E 00	LD A,00
1802	06 01	LD B,1
1804	0E 02	LD C,2
1806	16 03	LD D,3
1808	1E 04	LD E,4
180A	26 05	LD H,5
180C	2E 06	LD L,6
180E	FF	RST 38H

Speicher- platz	Befehlswort + Operand	Assembler- Code
1820	01 34 12	LD BC,1234H
1823	11 78 56	LD DE,5678H
1826	21 0A 09	LD HL,090AH
1829	FF	RST 38H

(3) Arithmetische und logische Operationen

Der Microprozessor Z80 verfügt über eine Reihe von Befehlen zur Addition und Subtraktion; ebenfalls können logische Operationen wie AND-, OR-, EXOR-Verknüpfungen und Vergleiche durchgeführt werden. Teilweise lassen sich diese Operationen auch durch 16-bit-Befehle ausführen.

Beispiele:

87 ADD A
Der Inhalt des Accumulators wird mit sich selbst addiert. Das Ergebnis steht wiederum im Accumulator. Dies gilt im übrigen für nahezu alle arithmetischen und logischen Operationen: Das Ergebnis steht im Accumulator.

88 ADC B
Der Inhalt des Registers B wird zum Inhalt des Accumulators hinzuaddiert; außerdem wird noch der Wert des Carry-flags, eine 1 oder eine 0, hinzurechnet. Das Ergebnis steht im Accumulator. (Das Carry-flag wird dann 1, wenn z. B. bei einer Addition ein Übertrag entstanden ist. Durch die Ausführung des Befehls ADC wird es wieder gelöscht, wenn kein neuer Übertrag errechnet wurde.)

91 SUB C
Der Inhalt des Registers C wird vom Inhalt des Accumulators abgezogen; das Ergebnis steht im Accumulator.

9E SBC(HL)
Auch bei den arithmetischen Befehlen gibt es die indirekte Adressierung. Hier werden die Daten, die unter der Adresse abgespeichert sind, auf die das HL-Registerpaar zeigt, vom Inhalt des Accumulators abgezogen. Außerdem wird noch der Wert des Carry-flags subtrahiert. Das Ergebnis steht wieder im Accumulator.

A2 AND D
Es wird eine logische AND-Verknüpfung der Inhalte der Register D und A durchgeführt; Ergebnis in A.

Beispiel: Inhalt der Register D=33H, A=55H

1800	3E 55	LD A,55H
1802	16 33	LD D,33H
1804	A2	AND D
1805	FF	RST 38H

Ergebnis im Accumulator: 11H.

01010101B 55H

00110011B 33H

00010001B 11H

F6 XX OR n

OR-Verknüpfung einer 1-Byte-Hexzahl mit dem Inhalt von A. Ein Beispiel für die unmittelbare Adressierung bei logischen Operationen.

A7 XOR A

Exclusive-OR-Verknüpfung des Accumulators mit sich selbst. Diese Operation wird oft zum Löschen des Accumulators verwendet: das Ergebnis ist stets 0.

24 INC H

Erhöht den Inhalt des Registers H um 1 (engl. increment)

DD 34 XX INC (IX+d)

Indizierte Adressierung: Der Inhalt der Speicherstelle mit der Adresse, die man erhält, wenn man zum Inhalt des IX-Registers d hinzuaddiert, wird um 1 erhöht.

Beispiel: IX=1B00; d=00; adressierte Speicherstelle also 1B00

1B00 DD 21 00 1B LD IX,1B00 ;1B00=DD

1B04 DD 34 00 INC (IX+0)

1B07 FF RST 38H

Ergebnis: Inhalt von 1B00 ist jetzt DE.

OD DEC C

Erniedrigt den Inhalt des Registers C um 1. (engl. decrement)

FD 35 XX DEC (IY+d)

Addiert zur Adresse, die in IY gespeichert ist, den Wert d hinzu und erniedrigt den Inhalt der Speicherstelle mit dieser Adresse um 1.

Beispiel: IY=1B00; IY+3=1B03;
1B00 FD 21 00 1B LD IY,1B00 ;Inhalt 1B03=1B
1B04 FD 35 03 DEC (IY+3)
1B07 FF RST 38H
Ergebnis: Inhalt von 1B03 ist 1A.

iel für eine 16-bit-Operation:

A ADC HL,BC

Durch diesen Befehl werden die 16-bit des Registerpaares BC plus Carry zum Inhalt des Registerpaares HL hinzu addiert. Die Summe steht im HL-Register.

DEC DE

Der Inhalt des Registerpaares DE wird um 1 erniedrigt.

Programmierübung 2 – Ladebefehle

Verzweigte Programme – Programmschleifen

Aufgabe: Es soll ein Programm entwickelt werden, das alle Speicherstellen von 1850H bis 186F löscht, d. h. mit dem Wert 00H lädt.

Es gibt eine ganze Reihe von Möglichkeiten, diese Aufgabe zu erfüllen. Hier soll eine bedingte Verzweigung benutzt werden, um zum Ziel zu gelangen. Die einzelnen Programmbefehle werden im folgenden erläutert.

LD B,20H

Das Register B wird hier als Zähler benutzt; es wird mit dem Wert 20H=32D geladen, weil ja auch 32 Speicherstellen gelöscht werden sollen.

LD HL,1850H

Das HL-Registerpaar wird mit der Adresse der ersten Speicherstelle geladen.

XOR A

Dieser Befehl setzt den Accumulator - Register A - auf 0; außerdem wird das Carry-flag auf 0 gesetzt.

LD (HL),A

Der Inhalt des Accu, also 0, wird an die Speicherstelle geladen, deren Adresse im HL-Register enthalten ist; zunächst 1850H.

INC HL

Der Inhalt des HL-Registerpaares wird um 1 erhöht, also auf 1851H.

DEC B

Der Inhalt des Registers B wird um 1 erniedrigt; auf 1FH.

JR NZ

Springe unter der Bedingung, daß durch den Decrement-Befehl keine 0 entstanden ist, das Zero-flag noch nicht gesetzt ist, von der momentanen Stelle aus zurück zum Befehl LD (HL),A. (Siehe unter Berechnung relativer Adressen.)

Es wird also im nächsten Durchgang – Schleife – an die Speicherstelle 1851h, dem neuen Inhalt des Registerpaares HL, ebenfalls eine 0 geschrieben. Danach wird der Inhalt des Registerpaares HL wieder um 1 erhöht, der Inhalt des Zählerregisters B um 1 erniedrigt usw. bis schließlich Register B=0 ist. Dadurch wird das Zero-flag gesetzt und der Rücksprungbefehl nicht mehr ausgeführt. Der nächste gültige Befehl

tet RST 38H
ckkehr zum Monitor am Programmende.

fgaben und Übungen:

Übertragen Sie das Programm in die Maschinensprache. Geben Sie es in den MPF-1 ein und starten Sie es.

Kontrollieren Sie, ob das Programm die gestellte Aufgabe korrekt erfüllt hat.

Setzen Sie nun wieder den PC auf den Programmanfang und betätigen Sie die Taste —STEP— Einzelschrittabarbeitung:

Die nächste gültige, aber noch nicht abgearbeitete Befehlsadresse erscheint im Adressfeld. Sie ist der Inhalt des Programnzählers PC. Drücken Sie jetzt die Taste —REG— und überprüfen Sie den Inhalt von Register B. Um den nächsten Programmschritt abzuarbeiten, holen Sie mit —PC— die gültige Programmadresse zurück. Drücken Sie wieder —STEP—. Überprüfen Sie nun den Inhalt von Registerpaar HL usw.

Auf diese Weise lassen sich Programme auf ihre korrekte Funktion hin in Einzelschritten überprüfen und nachvollziehen.

Beachten Sie den Inhalt des PC beim Sprungbefehl JR NZ. Der Verzweigungsbefehl beeinflusst den Inhalt des Programnzählers.

Die beiden Befehle

DEC B und JR NZ

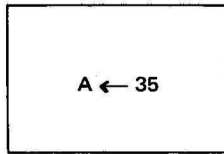
lassen sich zu einem einzigen zusammenfassen:

DJNZ

Der darin enthaltene Decrement-Befehl bezieht sich nur auf das Register B. Aus diesem Grunde benutzt man das Register B häufig als Zählerregister.

Flußdiagramm

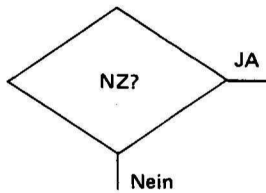
Zum Entwurf von Programmen verwendet man oft sogenannte Flußdiagramme; vor allem, um die Verzweigungen korrekt auszuarbeiten. Man bedient sich dabei verschiedener Symbole in der Form geometrischer Figuren. Die wichtigsten seien hier vorgestellt.



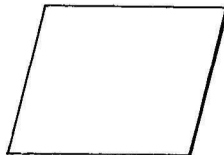
Befehlsausführung,
Operation



Programmbegrenzung



Entscheidung

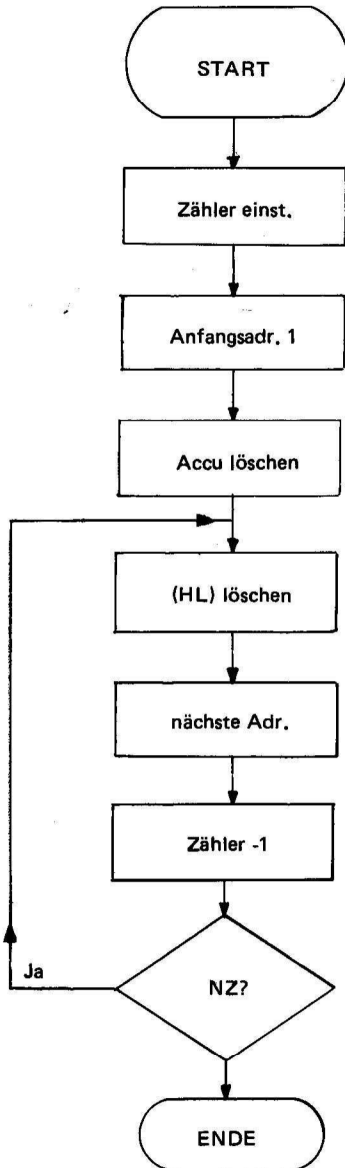


Ein- oder Ausgabe



Unterprogramm

FLUSSDIAGRAMM



Auflistung des Programms

1800	06 20	LD B,20H
1802	21 50 18	LD HL,1850
1805	AF	XOR A
1806	77	M1 LD (HL),A
1807	23	INC HL
1808	05	DEC B
1809	20 FB	JR NZ M1
180B	FF	RST 38H

Anmerkung: M1 ist eine sogenannte Sprungmarke.

Aufgabe: Ändern Sie das Programm so um, daß die Speicherstellen wie folgt belegt werden:

1840 = 0
1841 = 1
1842 = 2
1843 = 3
usw.
184F = F

Das F-Register

Das F-Register ist kein allgemeines Register, sondern zeigt verschiedene Zustände an, die infolge der ausgeführten Operationen entstanden sind. Diese Zustände können dann die Bedingungen für die weitere Fortführung des Programms sein. Die einzelnen bit des F-Registers werden als flags = "Fähnchen" bezeichnet, die den jeweiligen Zustand signalisieren.

Übersicht:

Bit-Nummer	7	6	5	4	3	2	1	0
Bezeichn. des flag	S	Z		H		P/V	N	C
Bezeichn. bei 1	M	Z				PE		C
Bezeichn. bei 0	P	NZ				PO		NC

Bit 0 ist das Carry-flag: gesetzt = Carry, nicht gesetzt = NC

Bit 2 ist das P/V-flag: gesetzt = parity even für gerade Zahl
nicht gesetzt = p. odd f. ungerade Z.

Bit 6 ist das Zero-flag: gesetzt = Z (zero = null)
nicht gesetzt : NZ

Bit 7 ist das Sign-flag: gesetzt = M für Minus
nicht gesetzt = P für Plus

Die anderen Bits werden von der CPU intern benutzt und spielen für die Programmierung keine Rolle.

Erläuterungen:

Zero-flag

Dieses Flag wird immer dann gesetzt, d. h. gleich 1, wenn das Ergebnis einer Operation eine 0 ist. Anderenfalls ist es nicht gesetzt. Durch Ergebnisse ungleich 0 wird es wieder zurückgesetzt, also 0.

Carry-flag

Das Carry-flag, Bit 0 des F-Registers, wird auf 1 gesetzt, wenn bei einer arithmetischen Operation ein Übertrag entsteht. Das ist dann der Fall, wenn bei einer Addition das Ergebnis größer als 8 Bit ist. Dann wird das 9. Bit sozusagen ins Carry-flag geschoben. Bei Subtraktionen wird ein negativer Übertrag ebenfalls im Carry-flag angezeigt. borrow = leihen.

P/V-flag

Durch dieses Flag wird angezeigt, ob das Ergebnis einer logischen Operation eine gerade Zahl oder eine ungerade ist. Bei gerader Zahl ist es gesetzt, bei ungerader Parität gelöscht.

(even = gerade, odd = ungerade)

Außerdem dient dasselbe Bit auch als Overflow-flag (Überlauf). Dieser Überlauf findet dann statt, wenn das Zweierkomplement von Zahlen mit Vorzeichen die Grenzwerte überschreitet. (größer als +127 oder kleiner als -128) – Wenn die Z80-CPU eine arithmetische Operation durchführt, kann die Zahl im Accumulator sowohl als eine vorzeichenlose Zahl im Bereich von 0 bis 255 als auch als Zahl mit Vorzeichen im Bereich von -128 bis +127 aufgefaßt werden. Bei der arithmetischen Operation kann das Carry- oder das Overflow-flag beeinflusst werden. Der Programmierer entscheidet, welcher Wert der Operation gültig ist.

Beispiel 1:

10101100B = ACH = 172D oder -84D

11101000B = E8H = 232D oder -24D

Die Addition dieser Zahlen ergibt:

10010100B + Carry = 194H = 404D oder 94H (ohne Carry) = -108D

Da bei dieser Addition das Carry-flag, nicht aber das Overflow-flag gesetzt wurde, ergeben sich die Möglichkeiten: Entweder eine vorzeichenlose Zahl mit Carry oder eine negative Zahl ohne Overflow. Das Zweierkomplement von 94H ist 6CH = 108D.

Beispiel 2:

01001010B = 4AH = 74D

01000010B = 42H = 66D

Die Addition ergibt ein Overflow, aber kein Carry.

10001100B = 8CH = 140D oder die negative Zahl -116D

Das Zweierkomplement von 8CH ist 74H = 116D.

Vorzeichen-flag (Sign-flag)

Negative Zahlen sind dadurch gekennzeichnet, daß das am weitesten links stehende Bit (bit 7) eine 1 ist. Wenn bit 7 im Accumulator nach einer arithmetischen oder logischen Operation gleich 1 ist, wird das Vorzeichen-flag auf 1 gesetzt, um das Vorliegen einer negativen Zahl anzuzeigen. Diese Flag wird vom Programmierer bei vorzeichenlosen Zahlen nicht beachtet.

Den Inhalt des F-Registers kann man mit Hilfe der Tasten —REG— und AF abfragen. Er erscheint in der Form einer 2ziffrigen Hexzahl. Daraus lassen sich dann die einzelnen Bits errechnen.

Der MPF-1 ist jedoch in der Lage, die einzelnen flags bitweise als Einsen und Nullen darzustellen. Dazu dienen die Tasten —SZ H— und —PNC—. Entsprechendes gilt für das Zweitregister F'.

Programmbeispiel 3:

Arithmetische Befehle

Aufgabe: Übertragen Sie das folgende symbolische Programm in die Assembler- und dann in die Maschinensprache.

Startadresse: 1800H

A ← E

A ← A+D

L ← A

A ← 0 LD A,0 nicht XOR A !

A ← A+0+C

H ← A

Ende

Geben Sie das Programm ein und laden Sie die Register D und E mit den unten angegebenen Werten. Führen Sie das Programm aus und überprüfen Sie die Ergebnisse.

Vorgaben		Ergebnisse				
D	E	HL	S	Z	P/V	C
5AH	A6H					
46H	77H					

Aufgabe: Verfahren Sie ebenso bei diesem Programm für eine 16-bit-Addition. Die Daten in den Speicherstellen 1A00 und 1A01 sollen zum Inhalt den Registerpaars DE hinzuaddiert werden. Die Summe soll im HL-Registerpaar stehen.

Startadresse: 1800

A ← (1A00)

A ← A+E

L ← A

A ← (1A01)

A ← A+D+C

H ← A

Ende

Vorgaben (1A00) = (1A01) =
 E = D =
 Ergebnisse HL = C= Z= P/V= S=

Aufgabe: Formen Sie das Programm in eine Subtraktion für 16 Bit um.

Programmbeispiel: Das folgende Programm kann dazu benutzt werden, die 32-bit-Daten an den Speicherstellen 1A00 bis 1A03 und 1A04 bis 1A07 zu addieren. Das Ergebnis wird unter den Adressen 1A08 bis 1A0B abgespeichert. Dabei wird die indizierte Adressierung angewandt.

Übersicht:

1A03	1A02	1A01	1A00
+ 1A07	1A06	1A05	1A04
1A0B	1A0A	1A09	1A08

Auflistung des Programms:

1800	06 04		LD B,4
1802	DD 21 00 1A		LD IX,1A00
1806	A7		AND A
1807	DD 7E 00	M1	LD A,(IX+0)
180A	DD 8E 04		ADC (IX+4)
180D	DD 77 08		LD (IX+8),A
1810	DD 23		INC IX
1812	10		DJNZ M1
1814	FF		RST 38H

Aufgaben: Berechnen Sie den relativen Sprung in 1812.

1. Laden Sie das Programm in den MPF-1 und testen Sie es.
2. Ersetzen Sie den Befehl ADC (IX+4) durch die Anweisung SBC (IX+4). Welchen Effekt hat diese Änderung?
3. Fügen Sie unmittelbar hinter dem Additions- bzw. Subtraktionsbefehl die Anweisung DAA ein. Benutzen Sie dazu die Taste —INS— des MPF-1 (insert=einfügen). Nach dieser Änderung rechnet das Gerät im Dezimalsystem.
4. Was bewirkt die Programm-Zeile AND A?

(5) Die Sprungbefehle

Die Z80-CPU kann beim Ablauf des Programms bedingte und unbedingte Sprungbefehle ausführen. Das bedeutet, daß ein Programm nicht nur linear von Anfang bis Ende gestaltet werden kann, sondern daß es möglich ist, mit Hilfe der Sprungbefehle die Abarbeitung des Programms an bestimmten anderen Programmteilen fortzuführen.

Die Bedingungen für die bedingten Sprünge werden von den Flags des F-Registers geliefert; unbedingte Sprünge werden unabhängig von Bedingungen in jedem Fall ausgeführt.

Beispiele zu den bedingten Sprüngen:

X XX JP Z,nn

Durch diesen Befehl wird bei Vorliegen der Bedingung Z —Zero-flag gesetzt— ein Sprung an die Adresse XXXX durchgeführt. Das bedeutet, der Programmzähler PC wird mit der Adresse XXXX geladen und so die Abarbeitung des dort abgelegten Programms erzwungen. Liegt die Bedingung Z nicht vor, wird mit dem Befehl fortgefahren, der hinter dem Sprungbefehl folgt. Der Programmzähler wird nicht mit XXXX geladen, sondern mit der Adresse des Befehls nach dem Sprungbefehl.

X JR NZ,n

Ein Sprung wird ausgeführt bei Vorliegen der Bedingung NZ —Zero-flag nicht gesetzt—. Hier handelt es sich jedoch um einen relativen Sprungbefehl. Das bedeutet, es soll von der momentanen Programmzeile aus um n Stellen vorwärts oder rückwärts gesprungen werden, um dort mit der Abarbeitung des Programms fortzufahren. JR deutet auf relative Adressierung hin. Die Sprungweite n —maximal 127 Stellen vorwärts und 128 Stellen rückwärts— kann beim MPF-1 mit Hilfe der —RELA— Taste berechnet werden. Nach Drücken dieser Taste wird mit "S" die Startadresse des Sprungbefehls abgefragt. Es muß mit der Adresse geantwortet werden, in der der Sprungbefehl —in diesem Falle 20— steht. Nach Betätigen der Taste +/- wird mit "d" = destination die Zieladresse erfragt, zu der gesprungen werden soll. Nach deren Eingabe und Betätigen der Taste —GO— wird automatisch der relative Wert in die richtige Speicherstelle eingetragen.

X XX CALL C,nn

Dieser Sprungbefehl ruft unter der Bedingung C —Carry-flag gesetzt— das Unterprogramm auf, das bei der Adresse XXXX beginnt. Ist die Bedingung C nicht gegeben, wird mit dem folgenden Befehl fortgefahren, der hinter dem Call-Befehl steht.

X DJNZ n

Durch diesen Befehl wird zunächst der Inhalt des Registers B um 1 erniedrigt. Sooft durch diesen Decrement-Befehl die Bedingung NZ erfüllt ist, wird ein relativer Sprung um n Stellen vor- oder rückwärts ausgeführt. Erst wenn der Inhalt des Registers B gleich 0 geworden ist und damit die Bedingung NZ

nicht mehr gegeben ist, wird mit der Abarbeitung des folgenden Befehls fortgefahren.

Beispiele für unbedingte Sprungbefehle:

C3 XX XX JP nn

Durch diesen Befehl wird in jedem Falle ein Sprung zum Programmteil mit der Adresse XXXX erzwungen.

E9 JP (HL)

Springe zu der Adresse, auf die das HL-Registerpaar zeigt und führe das Programm dort weiter durch.

18 JR n

Spring relativ zur momentanen Adresse vor- oder rückwärts unabhängig vom Zustand des F-Registers.

RETURN - Befehle

Unterprogramme müssen immer mit einem RET-Befehl abgeschlossen werden, um zu dem Programmteil zurückzuführen, von dem sie aufgerufen wurden. Es gibt auch hier wieder bedingte und unbedingte RET-Befehle.

E8 RET PE

Kehre zum aufrufenden Programm zurück, wenn die Bedingung PE –parity even = gerade Zahl– gegeben ist.

C9 RET

Kehre in jedem Falle zum aufrufenden Programm zurück. Dieser RET-Befehl wird am häufigsten verwendet.

Erläuterungen:

Beim Aufruf eines Unterprogramms wird die Rückkehradresse im Stack abgelegt. Durch einen RET-Befehl wird der Programmzähler PC wieder mit dieser Adresse aus dem Stack geladen.

rogrammbeispiel 4:

Programmverzweigungen (Loops)

rogrammbeispiel: Das folgende Programm bildet die Summe der Inhalte der Speicherstellen 1900 bis 190F und speichert das Ergebnis im DE-Registerpaar. Die einzelnen Programmschritte sind im folgenden ausführlich erläutert.

LD C,10H

Das Register C wird hier als Zähler benutzt. Es wird mit 10H=16D geladen, da die Inhalte von 16 Speicherplätzen aufaddiert werden sollen.

XOR A

Der Accumulator wird auf 0 gesetzt und das Carry gelöscht.

LD HL,1900

Die Anfangsadresse der Speicherstellen, die aufaddiert werden sollen, wird ins HL-Registerpaar geladen.

LD D,A

Der Inhalt des Accumulators wird ins Register D kopiert; zunächst ist also auch D=0. Im Verlaufe des Programms soll D den Übertrag speichern, der zwischendurch auftreten kann.

1 ADD A,(HL)

Addiert den Inhalt der Speicherstelle, deren Adresse im HL-Registerpaar steht, zum Accumulatorinhalt hinzu. Dieser Befehl wird 16mal ausgeführt; er ist Sprungziel-Adresse für einen später folgenden Sprungbefehl. M1 ist die Sprungmarke.

INC HL

Erhöht den Inhalt des HL-Registers um 1 auf 1901, 1902 usw.

JR NC,M2

Sprung bei Vorliegen der Bedingung NC. Zieladresse ist die Sprungmarke M2. Ist das Carry-flag gesetzt, so wird der nächste Befehl ausgeführt.

INC D

Der Inhalt des Registers D wird um 1 erhöht, wenn das Carry-flag gesetzt ist, sonst wird dieser Befehl übersprungen.

M2 DEC C

Der Inhalt des Registers C wird um 1 erniedrigt. M2 ist die Sprungmarke für den Fall, daß C=0 war.

JR NZ,M1

Sprung bei Vorliegen der Bedingung NZ. Zieladresse ist die Befehlszeile mit der Sprungmarke M1. Solange der Inhalt des Zählers noch nicht 0 ist, wird die Addition mit dem jeweils nächsten Speicherplatz durchgeführt. Erst wenn alle 16 Schleifendurchläufe beendet sind, folgt der nächste Befehl.

LD E,A

Der Inhalt des Accumulators, der ja das Ergebnis der Additionen enthält, wird in das Register E kopiert. Der Inhalt des Registers D ist jedesmal, wenn ein Überlauf entstanden war, um 1 erhöht worden. Daher enthält das Registerpaar DE die Summe der Speicherstellen 1900 bis 190F.

RST 38H

Es folgt noch der Befehl für das Programmende.

Bemerkungen: Wenn man sich vorstellt, daß die Speicherstellen 1900 bis 190 F die Ergebnisse von 16 Einzelmessungen enthalten, so läßt sich deren Durchschnittswert dadurch berechnen, daß noch ein Programmstück zum Teilen durch 16 angeführt wird.

- Aufgaben:
- Übertragen Sie das Programm in die Maschinensprache und lassen Sie es ausführen.
 - 1. Erstellen Sie das Flußdiagramm des Programms.
 - 2. Wählen Sie einen anderen Speicherbereich, dessen Summe berechnet werden soll.
 - 3. Was ist zu beachten, falls die Summe von mehr als 16 Speicherstellen berechnet werden soll?

Programmbeispiel 5:

Verschachtelte Verzweigungen, Zeitschleife

Der Maximalwert, mit dem sich das Zählerregister B laden lässt, beträgt FFH= 255D. Um größere Schleifen zu programmieren, bieten sich die Registerpaare an. Allerdings beeinflussen die entsprechenden INC- und DEC-Befehle nicht die Flags des F-Registers. Einen Ausweg bieten 2 ineinander geschachtelte Verzweigungen:

Programm-Auflistung:

1800	21 FF FF		LD HL,FFFF ; =65535D
1803	2D	M1,M2	DEC L ; L-1
1804	20		JR NZ M1 ; L=0?
1806	25		DEC H ; H-1
1807	20		JR NZ M2 ; H=0?
1809	FF		RST 38H

Dieses Programm führt 65278 Schleifendurchläufe aus. Es enthält eine kleine Schleife, bei der L jeweils um 1 erniedrigt wird, solange bis L=0 ist. Dann wird zur großen Schleife übergegangen; es wird H um 1 erniedrigt. Solange durch diese Operation H noch nicht 0 ist, wird zum Befehl DEC L zurückgesprungen. Da L zuletzt 0 war, wird durch den DEC-Befehl der Inhalt von L wieder auf FF gesetzt. Die kleine Schleife wird also erneut bis zum Wert L=0 durchlaufen usw.

- Aufgaben:
1. Geben Sie das Programm in den MPF-1 und lassen Sie es ausführen. Beobachten Sie, wieviel Zeit die Ausführung in Anspruch nimmt.
 2. Schreiben Sie ein Programm, dessen Ausführungszeit länger ist, indem Sie das Programm um eine weitere Schleife erweitern.
 3. Programmieren Sie eine Zeitschleife für ca. 3 Sekunden.

Bemerkungen: Die Ausführungszeiten lassen sich exakt berechnen. Die Abarbeitung eines jeden Befehls nimmt eine ganz bestimmte Anzahl von Taktzyklen in Anspruch. Die Länge des Taktzyklus ist festgelegt durch die Quarzfrequenz und den nachfolgenden Teiler; d. h. die Frequenz O, die an Pin 6 der CPU anliegt. Der MPF-1 enthält einen Quarz mit der Frequenz 3,58 MHz; diese wird durch 2 geteilt und als Clockfrequenz an die Z80-CPU gelegt. Die Zeit für einen Taktzyklus beträgt also ca. 560ns. In der Beschreibung der Befehle des Z80 findet man in der Spalte "No of T States" für jeden Befehl die Angabe, wieviele Taktzyklen benötigt werden.

Beispiel: Der Befehl DEC L beansprucht 4 Taktzyklen. Das bedeutet, die Ausführung des Befehls dauert 2,24us. Für den Befehl JR NZ werden 12 oder 7 Taktzyklen angegeben, je nachdem ob die Bedingung erfüllt ist oder nicht.

-
1. Der Inhalt der Speicherstelle, deren Adresse im HL-Registerpaar steht, wird an die Speicherstelle geladen, deren Adresse im DE-Registerpaar steht:
(DE)–(HL) oder (1900)–(0000)
 2. Dann wird der Inhalt des HL-Registerpaares um 1 erhöht und ebenso der Inhalt des DE-Registerpaares:
HL–0000+1 und DE–1900+1
 3. Der Inhalt des BC-Registerpaares wird um 1 erniedrigt, d. h. es brauchen jetzt nur noch 1FF Daten übertragen werden.
BC–0200-1
 4. Solange der Inhalt des Registerpaares noch nicht 0 ist, werden die Operationen 1 bis 4 wiederholt.

Auf ähnliche Weise läßt sich durch einen einzigen Befehl ein ganzer Speicherblock nach einem bestimmten Datum oder Zeichen durchsuchen.

Beispiel: Der Speicherbereich 1B00 bis 1CFF – 200H Stellen – wird nach dem Datum 3EH durchsucht.

LD A,3E	;gesuchtes Zeichen
LD HL,1B00	;Startadresse
LD BC,200	;Größe des Blocks
CPIR	;Suchen bis Zeichen gefunden oder
RST 38H	;alle Stellen überprüft sind

- Aufgaben:
1. Übertragen Sie das Programm in die Maschinensprache und lassen Sie es abarbeiten.
 2. Was geschieht, wenn das Datum 3E gefunden wird? Wie ist zu verfahren, um festzustellen, ob weitere Speicherstellen innerhalb der festgelegten Grenzen das Datum 3E enthalten?
 3. Schreiben Sie ein Programm, das sowohl einen Blocktransfer als auch eine Durchsuchung nach einem bestimmten Datum durchführt, indem Sie diese speziellen Befehle verwenden.
 4. Stellen Sie den Unterschied zwischen den Befehlen LDIR und LDDR fest.

Stack-Operationen und Unterprogramm-Technik

(1) Der Stack

Der Stack ist ein spezieller Speicherbereich im RAM, der für normale Programme und Daten nicht zur Verfügung steht. Er besitzt eine eigene Speicherorganisation: Die Daten, die zuletzt eingegeben wurden, werden als erste wieder ausgelesen. Man nennt das einen "lifo"-Speicher – last in first out –. Beim MPF-1 ist der Stackbereich festgelegt auf die Anfangsadresse 1FAE/1FAF. Die Bezeichnung Anfangsadresse bedeutet hier natürlich, von dort aus abwärts.

Ein Stackbereich läßt sich auch vom Benutzer definieren, indem man die oberste gewünschte Adresse-1 in den sogenannten Stackpointer –SP– lädt. Dabei handelt es sich um ein Register innerhalb der CPU, das die gerade gültige Stackadresse enthält: Der Stackpointer zeigt auf den Stack (pointer=Zeiger). Wird der Stack durch bestimmte Operationen geladen, so wird der Inhalt des Stackpointers erniedrigt; wird ein Datum aus dem Stack entnommen, so wird der Inhalt des Stackpointers erhöht.

Zur Veranschaulichung der Stack-Operationen soll das folgende Programm dienen.

1. LD SP,1FAF

Der Stackpointer –SP– wird mit der Adresse 1FAF geladen, d. h. der RAM-Bereich mit den Adressen niedriger als 1FAF steht jetzt als Stack zur Verfügung.

2. DEC SP

Der Inhalt des Stackpointers wird um 1 erniedrigt; er zeigt jetzt auf den "Boden" des Stack, die niedrigste mögliche Adresse für Stack-Operationen: 1FAE

3. LD (SP),H

Der Inhalt des Registers H wird an den Speicherplatz geladen, auf den der Stackpointer zeigt: 1FAE--H

4. DEC SP

SP wird um 1 erniedrigt: 1FAD

5. LD (SP),L

Der Inhalt von L wird in 1FAD geladen.

6. DEC SP

SP zeigt auf 1FAC

7. LD (SP),A

Accumulatorinhalt an die "oberste" Stackadresse.

8. DEC SP

9. LD (SP),F

Dann wird der Inhalt von F im Stack abgelegt.

-
10. LD C,(SP)
Mit diesem Befehl wird das zuletzt im Stack abgelegte Byte, auf dessen Adresse der Stackpointer zeigt, in das Register C geladen.
 11. INC SP
Der Inhalt des SP wird um 1 erhöht und zeigt die nächste gültige Adresse an.
 12. LD B,(SP)
Holt das nächste Byte vom Stack und übergibt es an B.
 13. INC SP
 14. LD E,(SP)
 15. INC SP
 16. LD D,(SP)
 17. INC SP

Durch die Befehle 1 bis 17 wurde mit Hilfe der Zeigeroperationen Daten im Stack abgespeichert und in umgekehrter Reihenfolge wieder hervorgeholt, um sie in den Registerpaaren BC und DE abzulegen.

Diese Art der Speicherorganisation wird von der Z80-CPU jedoch automatisch durchgeführt und zwar für jeweils 16 Bit, um vollständige Adressen, die in einem Registerpaar stehen, zu laden. Dazu dienen die Befehle PUSH und POP.

PUSH bedeutet: Schreibe in den Stack ein 16-bit-Wort und erniedrige den Inhalt des Stackpointer um 2, da ja 2 Byte im Stack abgelegt wurden.

POP bedeutet: Es werden die "obersten", d. h. die zuletzt abgelegten beiden Bytes wieder ausgelesen und der Inhalt des Stackpointers um 2 erhöht.

Mit Hilfe dieser Befehle läßt sich das obige Programm aus 17 Befehlsschritten wesentlich vereinfachen:

PUSH HL ein 1-byte-Befehl (implizierte Adressierung) ersetzt die Instruktionen 2, 3, 4 und 5 im oben aufgelisteten Demonstrationsprogramm: Der 16-bit-Inhalt des Registerpaares HL wird im Stack abgelegt; der Inhalt des Stackpointers wird um 2 erhöht.

PUSH AF bewirkt dasselbe wie die Befehlsfolge 6, 7, 8 und 9.

POP BC holt die zuletzt abgelegten 16 Bit aus dem Stack und schreibt sie in Registerpaar BC. 10, 11, 12 und 13.

POP DE ist der 1-byte-Befehl anstelle der Befehlsfolge 14, 15, 16 und 17.

Die Befehle PUSH und POP können also zur zeitweisen Zwischenspeicherung von Daten im Stack, aber auch, wie hier zum Austausch von Registerinhalten verwendet werden.

Zu beachten ist, daß die Anzahl der POP-Befehle gleich der Anzahl der PUSH-Befehle sein muß.

(2) Unterprogramm (subroutine)

Es gibt innerhalb von längeren Programmen oft den Fall, daß das gleiche Teilproblem mehrmals bearbeitet werden muß. So kann es zum Beispiel sein, daß im Verlaufe eines Programms mehrere Additionen vorkommen. Es ist dann nicht notwendig, das Additionsprogramm mehrmals zu schreiben. Man wendet dann die Unterprogramm-Technik an: Man schreibt das Programmstück "Addition" nur einmal und ruft es, wenn erforderlich, mit einem CALL-Befehl auf. (call=rufen) Dabei wird die nächste gültige Adresse des aufrufenden Programms in den Stack geladen. Der Programmzähler wird dagegen mit der Adresse aus dem CALL-Befehl geladen: Es erfolgt ein Sprung zum Unterprogramm.

CALL 1A38

ruft das Unterprogramm auf, das bei der Adresse 1A38 beginnt. Er bewirkt dasselbe wie die Befehlsfolge:

PUSH PC	Lege die gültige PC-Adresse im Stack ab.
JP 1A38	Springe zur Adresse 1A38.

Am Ende eines Unterprogramms muß die Rückkehr zum aufrufenden Programm erfolgen:

RET

schließt das Unterprogramm ab und bewirkt dasselbe wie der Befehl POP PC. Es wird die zuletzt eingetragene Adresse vom Stack geholt und an den Programmzähler übergeben.

Unterprogramme können auch ineinander verschachtelt sein, d. h. ihrerseits wieder Unterprogramme aufrufen, da durch die Organisation des Stack als lifo-Speicher eine korrekte Rückkehr aus den einzelnen Unterprogrammebenen gesichert ist.

Oft werden Unterprogramme verwendet, die ganz speziellen Zwecken dienen, z. B. komplizierteren mathematischen Operationen. Bevor ein Hauptprogramm ein solches Unterprogramm aufruft, ist eine Abstimmung der Übergabeparameter erforderlich:

-
1. Es muß festgestellt werden, aus welchen Registern oder Speicherplätzen das Unterprogramm die zu verarbeitenden Daten übernimmt. (Übernahme)
 2. Die Inhalte der Register, die durch die Ausführung des Unterprogramms nicht verändert werden dürfen, müssen in den Zweitregisterblock oder den Stack gerettet werden.
 3. Es muß festgestellt werden, in welcher Weise das Unterprogramm das Resultat an das aufrufende Programm übergibt. (Rückgabe)

Falls Unterprogramme selbst erstellt werden sollen, ist es daher zweckmäßig, folgende Punkte zu beachten:

1. Wählen Sie für das Unterprogramm einen (kurzen) Namen, der auf die Aufgabenstellung des Unterprogramms hinweist.
2. Legen Sie fest, wie die Daten bereitgestellt werden, die das Unterprogramm zur Bearbeitung benötigt. (Übernahme)
3. In welcher Weise kann das Unterprogramm das Ergebnis am Ende wieder zurückgeben? (Rückgabe)
4. Welche Registerinhalte werden durch das Unterprogramm zerstört?
5. Wieviel Speicherplatz und wieviel Zeit für die Abarbeitung benötigt das Unterprogramm?

Diese Parameter sollte man der Auflistung immer hinzufügen. Dadurch wird die Benutzung wesentlich erleichtert.

Programmbeispiel 7:

Unterprogramm für eine mehr-byte-Addition im BCD-Format

Programmbeispiel:

Name: MADD (multiple byte addition decimal)
Übernahme: HL zeigt auf das niedrigste Byte des ersten Summanden
DE zeigt auf das niedrigste Byte des zweiten Summanden,
B enthält die Anzahl der Bytes, die addiert werden
(1 Byte = 2 BCD-Ziffern)
Rückgabe: IX zeigt auf das niedrigste Byte des Resultats
veränderte Register: AF, B, HL, DE, IX
Speicherbedarf: 15 Bytes

AF		XOR A	;Carry löschen
1A	M1	LD A,(DE)	
86		ADD A,(HL)	
27		DAA	
DD 77 00		LD (IX),A	
13		INC DE	
23		INC HL	
DD 23		INC IX	
10 F4		DJNZ M1	
C9		RET	

Aufgaben und Übungen:

1. Zwei 4-byte BCD-Zahlen sind im Speicherbereich mit den Anfangsadressen 1A00 bzw. 1A40 abgelegt. Sie sollen mit Hilfe des obigen Unterprogramms addiert werden; das Ergebnis soll im Speicherbereich ab 1A08 stehen. Zum Aufruf von MADD dient das folgende Hauptprogramm:

```
LD B,4      ;Anzahl d. Bytes=4
LD HL,1A00
LD DE,1A40
LD IX,1A08
CALL MADD
RST 38H
```

Übertragen Sie das Hauptprogramm in die Maschinensprache; ordnen Sie beiden Programmen Speicherplätze zu und testen Sie sie auf dem MPF-1.

2. Schreiben Sie ein Programm, mit dem zwei 8-byte-Zahlen addiert werden, die im Speicherbereich 1A00 bzw. 1A08 stehen. Das Ergebnis soll wieder in 1A00 abgelegt werden.
3. Ändern Sie das Unterprogramm in eine BCD-Subtraktion oder eine binäre Addition/Subtraktion um.
4. Schreiben Sie ein Programm, das die 16-bit-Daten im HL-Register in ihr 2er-Komplement umwandelt. (Accu-Befehl NEG)
5. Benutzen Sie die Methode der Zweierkomplementbildung, um die Daten im DE-Registerpaar von den Daten im IY-Register abzuziehen und das Ergebnis in IY zu speichern.

Rotations- und Schiebefehle

Zusammen mit dem Carry-flag ergibt sich aus den 8-bit-Daten ein 9-bit-Wort. Der Mikroprozessor Z80 verfügt über insgesamt 13 verschiedene Befehle, um dieses 9-bit-Wort durch Rotations- und Schiebefehle zu manipulieren. Der Aufbau des mnemonischen Codes sei im Folgenden erläutert. Die graphische Darstellung macht die Wirkung der Befehle deutlich.

1. Wenn der erste Buchstabe des Befehls ein R ist, handelt es sich um einen Rotationsbefehl. Er wird dazu benutzt, ein 9-bit-Wort (8 bit+C) um eine Stelle rechts oder links herum rotieren zu lassen.

Beispiele: RLCA, RL, RRA usw.

Ist der erste Buchstabe ein S, so handelt es sich um einen Schiebefehl. Alle 9 bit werden um eine Stelle nach links oder rechts verschoben. Im Unterschied zu den Rotationsbefehlen wird das rechts bzw. links herausgeschobene bit nicht an der anderen Seite wieder hereingeschoben.

Beispiele: SLA, SRL

2. Wenn der 2. Buchstabe – von links – ein R ist, handelt es sich um einen Rechtsschiebe oder Rechtsrotationsbefehl. Entsprechend deutet ein L an 2. Stelle auf einen Links-, Schiebe- oder Rotationsbefehl hin.

3. Die Bedeutung des 3. Buchstabens ist komplizierter:

- a) bei Rotationsbefehlen

C bedeutet eine Rotation von 8 bit, carry nicht eingeschlossen.

A an 3. oder 4. Stelle deutet an, daß es sich um einen Accumulator-Rotationsbefehl handelt.

D weist auf 4-bit-Rotationsbefehle zwischen HL und Accumulator hin.

- b) bei Schiebefehlen

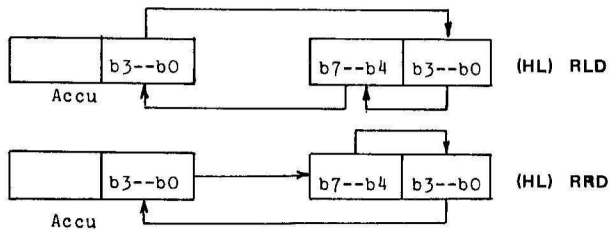
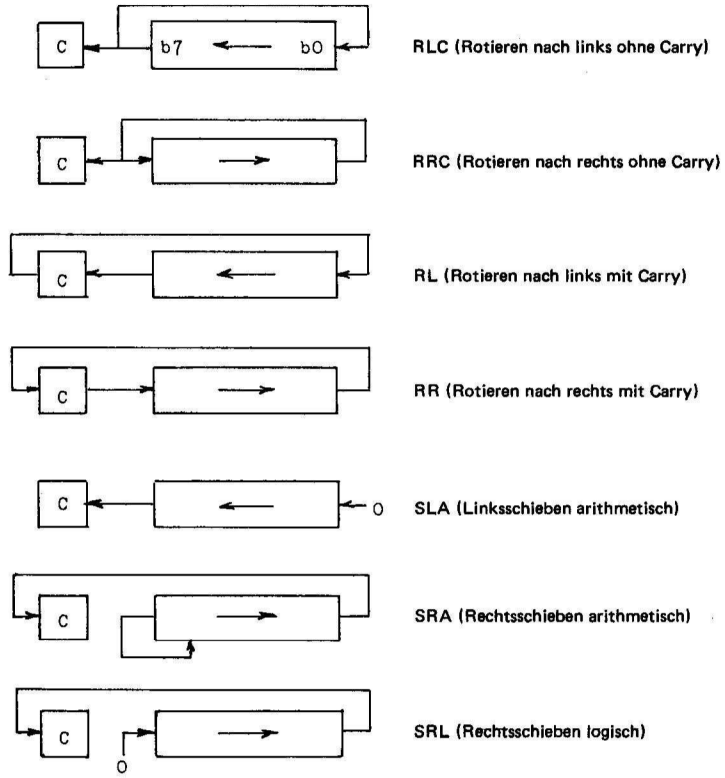
A bedeutet arithmetischer Schiebefehl: 1 bit nach links schieben entspricht einer Multiplikation mit 2; 1 bit nach rechts schieben einer Division durch 2. Da bit 7 als Vorzeichen interpretiert wird und das Vorzeichen durch diese Operationen nicht beeinflußt wird, muß bit 7 unverändert bleiben.

L an 3. Stelle deutet auf einen logischen Schiebefehl hin.

Beispiel: SRL; dabei wird stets eine 0 von links nach bit 7 hineingeschoben.

Graphische Darstellung der Rotations- und Schiebefehle:

ÜBERSICHT ÜBER DIE ROTATIONS- UND SCHIEBEBEFEHLE



Programmbeispiel 8:

Binäre Multiplikation

Es gibt zumindest zwei einfache Möglichkeiten, eine Multiplikation durchzuführen. Entweder man benutzt den Algorithmus der mehrfachen Addition oder den Verschiebealgorithmus.

Die mehrfache Addition ergibt sich daraus, daß man für z. B. 3x4 auch 4+4+4 schreiben und ausrechnen kann. Diese Möglichkeit erweist sich bei größeren Zahlen aber als umständlich und langwierig. Einen Ausweg bietet der Verschiebe-Algorithmus. Die Multiplikation von zwei Binärzahlen gestaltet sich bei der Durchführung "von Hand" so:

11x5

1011x0101

1011

0000

1011

0000

110111

oder

1011x0101

0000

1011

0000

1011

110111

55D

Aus dem angeführten Beispiel ist zu ersehen, daß die Multiplikation lediglich aus einer mehrfachen Verschiebung und einer Addition besteht. Dabei ist das Teilprodukt gleich dem 1. Faktor, wenn der 2. eine 1 im entsprechenden bit aufweist.

Diese Methode der Verschiebung und Addition wird im Programmierbeispiel angewendet. Allerdings wird das Verfahren zur günstigeren Programmierung etwas verändert.

Der 2. Faktor wird so lange nach links geschoben, bis im Carry-flag eine 1 erscheint. Dann wird der 1. Faktor in ein Ergebnisregister geladen und nach links geschoben. Wieder wird der 2. Faktor nach links geschoben und das Carry-flag abgefragt; ist es 1, so wird der 1. Faktor zum bisherigen Inhalt des Ergebnisregisters hinzuaddiert. Ein Zähler überwacht die Anzahl der Verschiebungen.

Die Multiplikation des obigen Beispiels sieht dann so aus:

1.	links schieben des 2. Faktors:	1010 0101	
2.	Kein Carry: Links schieben:	C 0100	
3.	Carry: 1. Faktor ins Ergebn.:		1011
4.	Ergebnis links schieben:		10110
5.	2. Faktor links schieben:	1000	
6.	Kein Carry: Erg. 1. sch:		101100
7.	2. Fakror links schieben:	C 0000	
8.	Carry: 1. Faktor addieren:		110111

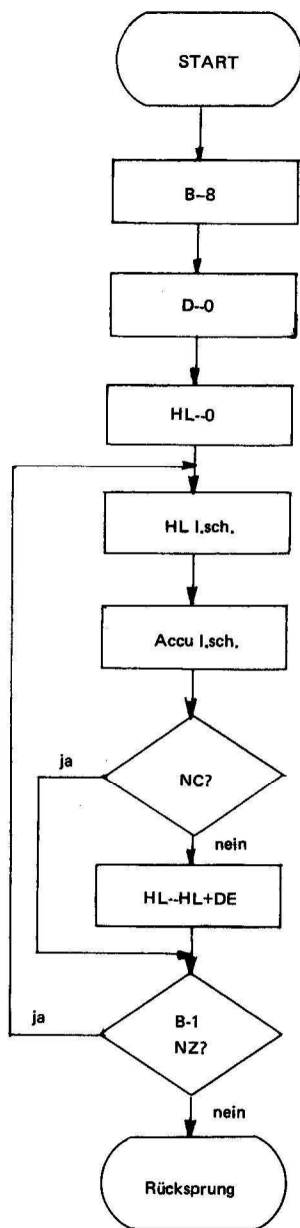
Im folgenden Programmierbeispiel soll eine 8-bit-Zahl, die im Register E steht mit einer 8-bit-Zahl im Accumulator multipliziert werden. Das Produkt soll im HL-Registerpaar stehen.

Das Programm ist als Unterprogramm ausgeführt.

Name: MP8
Übernahme: Multiplikator = 2. Faktor in A
Multiplikand = 1. Faktor in E
Rückgabe: Produkt in HL
Speicherbedarf: 14 Bytes
Zeit: 395 Taktzyklen ca 0,2ms

Programmauflistung:

06 08		LD B,8	;Zähler für 8 bit
16 00		LD D,0	
62		LD H,D	
6A		LD L,D	;D u. HL löschen
29	M1	ADD HL,HL	;HL links schieben
07		RLCA	;bit 7 ins Carry-flag
30 01		JR NC,M2	;Carry überprüfen
19		ADD HL,DE	;HL←HL+DE
10 F9	M2	DJNZ M1	;Ende?
C9		RET	



Aufgaben und Übungen:

1. Das folgende Programm kann dazu benutzt werden, die 32-bit-Daten im HL- und DE-Registerpaar, die als eine Zahl zusammenhängend betrachtet werden, nach rechts zu schieben bzw. durch 2 zu dividieren.

```
SRA H
RR L
RR D
RR E
RST 38H
```

Verändern Sie das Programm so, daß die Daten nach links geschoben bzw. mit 2 multipliziert werden.

2. Schreiben Sie ein Programm, mit dem die 32-bit-Daten im Speicherbereich 1A00 bis 1A03 um 5 bit nach links geschoben werden. (Multiplikation mit 20H=32D) Als Anfangsadresse des Programms wählen Sie 1830.
3. Das folgende Programm multipliziert die 16-bit-Daten im DE-Registerpaar mit dem Inhalt des Accumulator. Vergleichen Sie dieses Programm mit dem Unterprogramm MP8.

```
LD BC,800
LD H,C
LD L,C
M1 ADD HL,HL
RLA
JR NC,M2
ADD HL,DE
ADC A,C
M2 DJNZ M1
RST 38H
```

4. Schreiben Sie ein Programm zur Multiplikation der 32-bit-Daten im Speicherbereich 1A00 bis 1A03 mit den 32-bit-Daten im Bereich 1A04 bis 1A07. Das Ergebnis soll im Speicherbereich 1A08 bis 1A0F abgelegt werden.

Programmbeispiel 9:

Binäre Division

Für die binäre Division wird ebenfalls ein Verschiebealgorithmus angewandt, in Verbindung mit einer Subtraktion. Das Verfahren sei am Beispiel der Division 237:20 erläutert. Quotient, Dividend und Divisor werden zunächst wie im Ablaufschema ganz oben angeordnet. Danach werden Quotient und Dividend um ein bit nach links geschoben. Um den Vergleich von Dividend und Divisor zu erleichtern, werden rechts vom Divisor 7 Nullen angefügt (2). Da der Divisor jetzt größer ist als der Dividend, wird in den Quotienten eine 0 eingeschrieben (2). Es erfolgt die nächste Linksverschiebung von Quotient und Divisor und wieder der Vergleich (3). Solange der Divisor größer ist als der Dividend, werden in den Quotienten 0 eingeschrieben. (4,5) Ist jedoch der Divisor aufgrund der Verschiebungen größer als der Dividend, so erscheint im Quotient eine 1 (6). Es wird eine Subtraktion durchgeführt: Dividend minus Divisor (6). Die Differenz ist der neue Wert des Dividenden. Diese Prozedur aus Verschiebungen, Vergleichen und Subtraktionen wird so lange fortgeführt, bis Divisor und Dividend genau untereinander stehen (9). Ist der "Rest" der Division nicht gleich 0, so kann man durch weitere Divisionen auch die Werte der binären Brüche berechnen: Halbe, Viertel, Achtel usw.

Aus der Beschreibung ergibt sich, daß 3 Speicherbereiche benötigt werden, um Quotient, Divisor und Dividend abzulegen und zu bearbeiten.
Beispiel: Es soll ein Programm erstellt werden, das die 16-bit-Daten im DE-Registerpaar durch die 16-bit-Daten im BC-Registerpaar teilt. Das Ergebnis, der Quotient soll im HL-Registerpaar, der Rest als binärer Bruch im DE-Registerpaar abgelegt sein.

Ablaufplan einer binären Division
237 : 20

	00000000	Quotient
(1)	11101101	Dividend
	00010100	Divisor
	0	links schieben
(2)	11101101	
	0001010000000000	mit Nullen auffüllen
	00	links schieben
(3)	11101101	
	0001010000000000	
	000	links schieben
(4)	11101101	
	0001010000000000	
	0000	links schieben
(5)	11101101	
	00010100000000	
	00001	links schieben
(6)	11101101	Subtraktion, 1 in den Quotienten
	000101000000	
	000010	links schieben
(7)	11101101	Differenz der Subtraktion
	0001010000	
	0000101	links schieben
(8)	01001101	Subtraktion, 1 in den Quotienten
	000101000	
	00001011 = Quotient = 11D	
(9)	00100101	links schieben
	00010100	Subtraktion, 1 in den Quotienten
	00010001 = Rest = 17D	

Programmauflistung

Unterprogramm zur Division zweier 16-bit-Zahlen

Name: DIV16
Übernahme: Dividend in DE
Divisor in BC
Rückgabe: Quotient in HL
Rest in DE
veränderte Register: AF, DE, HL

AF		XOR A	;Carry löschen
67		LD H,A	
6F		LD L,A	;HL löschen
3E 10		LD A,10H	;Zähler A=16D
CB 13	M1	RL E	
CB 12		RL D	;DE links schieben
ED 6A		ADC HL,HL	;HL links schieben
ED 42		SBC HL,BC	;Subtraktion
30 01		JR NC,M2	;Vergleich
09		ADD HL,BC	;Subt. rückgäng. machen
3F	M2	CCF	;Zwischenerg. in Carry
3D		DEC A	
20 F1		JR NZ,M1	
EB		EX DE,HL	
ED 6A		ADC HL,HL	;letztes Bit
C9		RET	

Bemerkungen zum Programm:

1. Statt LD H,A und LD L,A könnte man auch den Befehl LD HL,0 benutzen. Aber dieser scheinbar einfachere Befehl belegt 3 Byte Speicherplatz und benötigt 10 Taktzyklen Ausführungszeit. Stattdessen beanspruchen die beiden im Programm benutzten Befehle nur 2 Byte und 8 Taktzyklen.
2. Additions- und Subtraktionsbefehle können als Linksschiebe- oder Rotationsbefehle betrachtet werden. Im obigen Beispiel hat der Befehl ADC HL,HL die Wirkung einer 16-bit-Rotation. Einige weitere Beispiele dieser Art sind im folgenden beschrieben:
ADD A,A Linksschieben des Inhalts von Register A um 1 bit nach links oder Multiplikation von A mit 2.
ADC A,A Rotieren von A nach links um 1 bit.
ADD HL,HL Linksschieben des HL-Registerpaares oder Multiplikation mit 2.

Aufgaben und Übungen:

1. Laden Sie das Programm in den MPF-1 und speichern Sie es auf Cassette ab.
2. Ersetzen Sie den Befehl RET durch RST 38H und führen Sie folgende Divisionen durch:

Dividend	Divisor	Quotient	Rest
8686H	0020H		
FFFFH	0003H		
5A48H	124H		
0H	124H		
1234H	0H		

3. Ändern Sie das Programm so ab, daß der binäre Bruch auf 16 Stellen berechnet wird.
4. Benutzen Sie das obige Programm als Unterprogramm und schreiben Sie ein Hauptprogramm, um zu erreichen, daß die Daten im Speicherbereich 1A00 - 1A01 durch die Daten in 1A04 - 1A05 dividiert werden. Der Quotient soll an den Speicherstellen 1A00 und 1A01 abgelegt werden.
5. Schreiben Sie ein Programm zur Division der 4-byte-Daten in 1A00 bis 1A03 durch die 4-byte-Zahl, die unter einer Adresse abgelegt sind, auf die das HL-Registerpaar zeigt. Der Quotient soll an den Speicherplätzen 1A00 - 1A03, der Rest an den Speicherplätzen 1A04 - 1A07 stehen.

Programmbeispiel 10:

Umwandlung von Binärzahlen in BCD-Zahlen

Es gibt verschiedene Methoden, eine Zahl im Zweiersystem in eine BCD-Zahl umzurechnen. (BCD = binär codiertes Dezimalsystem) Dabei wird jede einzelne Ziffer zwar im Zweiersystem dargestellt, jedoch entspricht die Position jeder Ziffer ihrem Stellenwert im Dezimalsystem).
Der Befehlssatz des Z80 enthält bereits den Befehl DAA = BCD-Korrektur im Accumulator. Damit kann eine Binärzahl im Accumulator unmittelbar in eine BCD-Zahl umgewandelt werden. Allerdings genügt dieser Befehl allein nicht für die unbeschränkte Umwandlung, da die größte Hexzahl im Accumulator FFH ist, alle 8 bit = 1. Dies entspricht aber einer 3ziffrigen BCD-Zahl: 255 oder 10 0101 0101; das sind 10 bit, die im Accumulator keinen Platz finden: Es würde nur der Wert 55 ausgegeben.

Für die Durchführung einer brauchbaren BCD-Wandlung werden zwei Speicherbereiche festgelegt, die die binären, bzw. die BCD-Daten enthalten. Der Bereich für die BCD-Daten wird zu Beginn auf 0 gesetzt. Im folgenden werden so lange Schiebe- und Testbefehle ausgeführt, bis alle Binärdaten vollständig nach links herausgeschoben sind. Durch diese Verschiebungen wird jeweils das am weitesten links stehende bit automatisch ins Carry-flag geschoben. Die BCD-Daten werden dann verdoppelt (linksschieben) und das Carry-flag aus den Binärdaten von rechts in bit 0 hineingeschoben. Danach erfolgt mit DAA die BCD-Korrektur. Aus dem Ablaufplan für die Umwandlung von 3DH sind die einzelnen Schritte zu ersehen.

	Binärdaten 3DH	BCD-Daten	
	00111101B =3DH	00000000	
(1)	01111010 linksschieben	00000000 linksschieben	
(2)	11110100 "	00000000 "	
(3)	C11101000 "	00000001 "	C einschieben
(4)	C11010000 "	00000011 "	"
(5)	C10100000 "	00000111 "	"
(6)	C01000000 "	00001111 "	"
		00010101	DAA
(7)	10000000 "	00101010 "	
		00110000	"
(8)	C00000000 "	01100001 " =61D	

Vorüberlegungen zur Programmgestaltung:

1. Die binären Daten werden im RAM-Bereich abgespeichert, beginnend mit der Adresse 1A00. Register D sei der Zähler für die Anzahl der binären Bytes. Register E zählt die BCD-Stellen.
2. Der Speicherbereich für die BCD-Daten muß gelöscht werden; seine Anfangsadresse ist 1A08.
3. Linksschieben der Binärdaten um 1 bit. Bit 7 wird ins Carry-flag geschoben.
4. Die BCD-Daten werden mit sich selbst und mit C addiert: Linksschieben und Carry auf bit 0 einschieben.
5. Überprüfen, ob alle bits der Binärdaten aus dem Speicherbereich herausgeschoben sind; wenn nicht, ab Schritt 3 wiederholen. Wenn ja, Programm beenden.

Aufistung des Programms:

Binärdaten in 1A00 ff

Anzahl der Binär-byte ins D-Reg.

BCD-Daten in 1A08 ff

Anzahl der BCD-byte ins E-Register

Verwendete Register: AF, BC, DE

1800	AF		XOR A	
1801	43		LD B,E	
1802	21 08 1A		LD HL,1A08	
1805	77	M1	LD (HL),A	;BCD-Speicher löschen
1806	23		INC HL	
1807	10 FC		DJNZ M1	
1809	7A		LD A,D	;A= Anzahl d. Bytes
180A	87		ADD A,A	
180B	87		ADD A,A	
180C	87		ADD A,A	;Ax8
180D	4F		LD C,A	;C= Anzahl d. Bit
180E	2E 00	M2	LD L,0	;HL= 1A00
1810	42		LD B,D	
1811	CB 16	M3	RL (HL)	
1813	23		INC HL	
1814	10 FB		DJNZ M3	
1816	2E 08		LD L,8	;HL= 1A08
1818	43		LD B,E	
1819	7E	M4	LD A, (HL)	
181A	8F		ADC A,A	
181B	27		DAA	
181C	77		LD (HL),A	
181D	23		INC HL	
181E	10 F9		DJNZ M4	
1820	0D		DEC C	
1821	20 EB		JR NZ,M2	
1823	FF		RST 38H	

Bemerkungen zur Programmiertechnik

- Multiplikationen von Binärzahlen mit einer Konstanten: Im obigen Programm wird die Anzahl der bits einer bekannten Anzahl von Bytes durch eine Multiplikation mit 8 errechnet. Diese Multiplikation wird hier durch 3malige Addi-

tion des Accumulatorinhaltes mit sich selbst ausgeführt; das entspricht jeweils einer Multiplikation mit 2 oder einem Linksschieben um 1 bit. Diese Methode der Multiplikation läßt sich auch bei Zahlen anwenden, die nicht Potenzen von 2 sind.

Beispiel: Die Daten im Register D sollen mit 6 multipliziert werden und das Ergebnis im Accumulator stehen.

```
LD A,D      ;A=D
ADD A,A     ;A=2xD
ADD A,D     ;A=3xD
ADD A,A     ;A=6xD
```

2. Adressierungsmethode für Speicherstellen der gleichen Seite.
Das höherwertige Byte einer Adresse bezeichnet man auch als deren "Seite". So gehören die Adressen 1A08 und 1ADE zur Seite 1A. Diese Seite eines Speicherbereichs steht bei der indirekten Adressierung mit Hilfe des HL-Registers im Register H. Für Arbeiten auf der gleichen Seite eines Speicherbereiches genügt es dann, den Inhalt des L-Registers zu ändern, um den Bereich zu ändern. Von dieser "Seitenadressierung" (page-addressing) wird im obigen Programm Gebrauch gemacht.

Aufgaben und Übungen:

1. Laden Sie das Programm in den MPF-1 und speichern Sie es auf Cassette.
2. Führen Sie die folgenden Umrechnungen durch und überprüfen Sie die Ergebnisse.

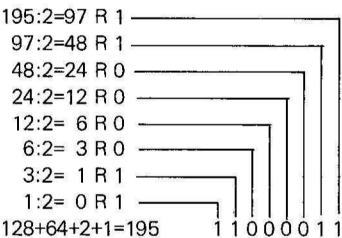
Binärzahl	HEX-Zahl	BCD-Zahl	Register D u. E	
1000000000	0200H		2	2
	FFFFH		2	3
	18000H		3	4
	5A48347FH		4	6
	2 hoch 32		8	0A
	2 hoch 63		8	0A
	2 hoch 64-1		8	0A

3. Machen Sie aus dem Programm ein Unterprogramm, und erstellen Sie ein Hauptprogramm, von dem aus der Inhalt der Registerpaare DE in eine BCD-Zahl umgerechnet wird. Diese soll dann im HL-Registerpaar stehen. Der ursprüngliche Inhalt von DE soll erhalten bleiben.
4. Die Binärzahl im E-Register (kleiner als 20H) soll mit 7 multipliziert werden; das Ergebnis soll im Accumulator stehen.

Programmbeispiel 11:

Umrechnung von BCD-Zahlen in Binärzahlen

Auch für diese Umwandlung werden die Schiebebefehle verwendet. Zur Veranschaulichung dient das folgende Beispiel:
Die Zahl 195D soll in eine duale Zahl umgerechnet werden. Dazu wird die Zahl bis zum "Ende" immer wieder durch 2 geteilt. Die entstehenden "Reste" dieser Teilungen ergeben dann die gesuchte Dualzahl.



Beim BCD-Code wird jede Dezimalstelle durch 4 bit = 1 digit repräsentiert. Verschiebt man ein solches digit um 1 bit nach rechts, so entspricht das zunächst einer Division durch 2. Wird jedoch eine 1 aus einem höherwertigen digit in ein niederwertiges geschoben, so wird sein Wert auf 5 reduziert. Da aber eine 1 an der am weitesten links liegenden Stelle eines digit eine 8 repräsentiert, muß bei jedem Übergang von einem digit ins niederwertige, eine 3 subtrahiert werden, um den korrekten Wert 5 zu erhalten.

Das folgende Programm baut auf diesen Überlegungen auf und ist in der Lage, eine 5-byte = 10-digit BCD-Zahl in eine 4-byte-Binärzahl umzurechnen. Die größte mit 4 byte darstellbare Zahl ist 4 294 967 295; diesen Wert darf die BCD-Zahl, die umgerechnet werden soll, nicht überschreiten.

Der Speicherbereich mit den Adressen 1A00 - 1A03 nimmt die errechneten Binärdaten auf, der Speicherbereich 1A04 - 1A08 enthält die umzurechnende BCD-Zahl.

Auflistung des Unterprogramms
zur Umrechnung einer 10stelligen BCD-Zahl

Übernahme: BCD-Daten in 1A04 bis 1A08
max. Wert: 4 294 967 295
Rückgabe: Binärzahl in 1A00 bis 1A03
veränderte Register: AF, BC, HL

```

0E 20          LD C,20H          ;ZÄHLER C=32
06 05          LD B,5            ;5 BCD-byte
AF             XOR A
21 08 1A       LD HL,1A08
7E             LD A,(HL)
1F             RRA
F5             PUSH AF          ;Carry retten
CB 7F          BIT 7,A           ;bit 7 testen
28 02          JR Z,M3          ;keine Korr,w.bit 7=0
D6             SUB 30H          ;Subtr.
CB 5F          BIT 3,A           ;bit 3 testen
28 02          JR Z,M4
D6 03          SUB 3            ;Subtr.
77             LD (HL),A        ;abspeichern
2B             DEC HL           ;nächstes byte
F1             POP AF          ;Carry zurückholen
10 EC          DJNZ M2
06 04          LD B,4           ;4 Binärbyte
CB 1e          RR (HL)
2B             DEC HL
10FB          DJNZ M5
0D             DEC C
20 DC          JR NZ,M1
C9             RET

```

Zur unmittelbaren Rückumrechnung folgt die Auflistung eines Programms zur Umwandlung einer 4-byte-Binärzahl in eine BCD-Zahl. Die Speicherbereiche für Übernahme und Rückgabe stimmen überein.

Auflistung des Unterprogramms
zur Umrechnung einer 4-byte-Binärzahl

Übernahme: Binärzahl in 1A00 bis 1A03
Rückgabe: BCD-Zahl in 1A04 bis 1A08
veränderte Register: AF, BC, HL

21 04 1A		LD HL,1A04
0605		LD B,5
36 00	M6	LD (HL),0
23		INC HL
10 FB		DJNZ M6
0E 20		LD C,20
68	M7	LD L,B
06 04		LD B,4
AF		XOR A
CB 16	M8	RL (HL)
23		INC HL
10 FB		DJNZ M8
06 05		LD B,5
7E	M9	LD A,(HL)
8F		ADC A,A
27		DAA
77		LD (HL),A
23		INC HL
10 F9		DJNZ M9
0D		DEC C
20 EB		JR NZ,M7
C9		RET

Aufgaben und Übungen:

1. Laden Sie die beiden Unterprogramme in den MPF-1 und speichern Sie sie auf Cassette.
2. Ersetzen Sie den RET-Befehl durch RST 38H. Laden Sie eine beliebige BCD-Zahl in den Speicherbereich 1A04 - 1A08 und lassen Sie die Umrechnung durchführen.
Tragen Sie die errechneten Werte in die folgende Tabelle ein.

	Dezimalzahl	umgerechnete Hexzahl	rückberechn.Dezim.Z.
1			
2			
3			
4			
5			

3. Schreiben Sie ein Programm für eine dezimale Multiplikation, indem Sie die entsprechenden Unterprogramme benutzen. Die dezimalen Faktoren sollen in den Registerpaaren DE und HL stehen, das Produkt im Speicherbereich 1A04 bis 1A08. Die Daten in DE und HL müssen unverändert bleiben.

Programmbeispiel 12:

Berechnung der Quadratwurzel

Es gibt verschiedene Methoden zur Errechnung der Wurzel einer Zahl. Die hier vorgestellte Methode des Wurzelziehens bei einer Binärzahl lässt sich verhältnismäßig leicht in ein Microcomputerprogramm umsetzen. Es soll die Quadratwurzel der Zahl 01010001B = 81D berechnet werden.

1. Ausgangsanordnung: X, Y, R und P seien Speicherbereiche. Die Zahl, deren Wurzel berechnet werden soll, steht in Block Y. Block P enthält die Zahl 01; die Blöcke X und R sind zu Beginn gelöscht, um später Daten aufzunehmen.
2. Die Zahl, die durch die Blöcke R und P gebildet wird, wird vom Inhalt der Blöcke X und Y abgezogen. Wenn das Ergebnis nicht negativ ist, wird eine 1 von rechts in Block R eingeschoben und dabei sein Inhalt um eine Stelle nach links geschoben. Ist das Ergebnis der Subtraktion dagegen negativ, dann bleiben die Daten in X und Y unverändert und die Daten in R werden um 1 bit nach links geschoben. Danach wird der Inhalt von Block Y um 2 bit nach links geschoben und der Vorgang beginnt von vorne. Diese Prozedur wird solange wiederholt, bis alle bit, die ursprünglich in Y standen, vollständig herausgeschoben sind.
Im Block R steht jetzt die Quadratwurzel.
Werden die Verschiebungen weiter durchgeführt, so erhält man gegebenenfalls die binären Brüche einer Wurzel.

Auf der Grundlage der beschriebenen Methode wurde das folgende Programm erstellt. Es berechnet die Wurzel einer 16-bit-Zahl, die im BC-Registerpaar steht. Die Berechnung wird bis zur achten Binärbruchstelle durchgeführt. Der ganzzahlige Anteil der Lösung steht im Register D, der gebrochene Anteil im Register E.

Ablaufplan einer Quadratwurzelberechnung

X	Y
	01010001 = 81D
	01
R	P

Ausgangsordnung

Ergebnis der Subtraktion nicht neg.

1 in R einschieben

Y enthält die Differenz d. Subtr.

00010001

101

0001000100 _

101

X u. Y 2 bit nach links

Erg. d. Subtr. negativ: 0 in R

0001000100

1001

000100010000 _

1001

X u. Y 2 bit nach links

Erg. d. Subtr. negativ: 0 in R

000100010000

10001

00010001000000 _

10001

X u. Y 2 bit nach links

Ergebnis nicht negativ: 1 in R

00000000000000

100101

Y enthält die Differenz

0000000000000000

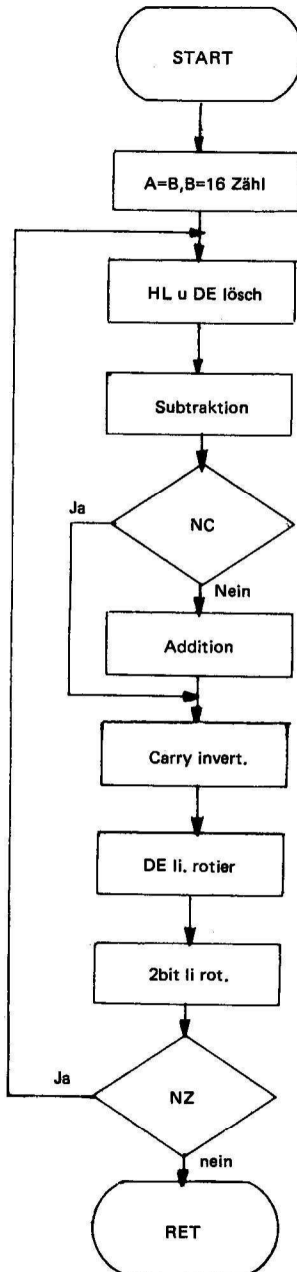
100101

X u. Y 2 bit nach links

Die ursprünglichen Daten in Y sind jetzt um 8 bit nach links verschoben.

Block R enthält die Wurzel 1001B = 9D.

Da die Blöcke X und Y gleich 0 sind handelte es sich bei der zu berechnenden Zahl um eine Quadratzahl.



Auflistung des Unterprogramms
zur Berechnung der Quadratwurzel
einer 16-bit-Binärzahl

Name: SQRT16
Übernahme: Binäre Daten in BC
Rückgabe: ganzzahliger Anteil in D
gebrochener Anteil in E
veränderte Register: AF, BC, DE, HL

```

78          LD A,B
06 10      LD B,10H          ;Zähler B=16
21 00 00   LD HL,0          ;HL löschen
54          LD D,H
5C          LD E,H          ;Ergebnisreg. löschen
d6 40      M2 SUB 40H        ;Konstante subtrah.
ED 52      SBC HL,DE        ;HL - DE
30 04      JR NC,M1         ;HL größer DE?
C6 40      ADD 40H
ED 5A      ADC HL,DE        ;A u. HL wiederherst.
3F          M1 CCF          ;Teilerg. in Carry
CB 13      RL E             ;Teilerg. in E schieb.
CB 12      RL D             ;u. Ergebn. links sch.
CB 21      SLA C
17          RLA
ED 6A      ADC HL,HL        ;AC u. HL 2 bit links
CB 21      SLA C
17          RLA
ED 6A      ADC HL,HL
10 E5     DJNZ M2
C9          RET

```

Bemerkung: Aus der Übersicht geht hervor, wie die einzelnen Register des Unterprogramms den Blöcken des Ablaufplanes entsprechen.

X		Y	
H	L	A	C
D	E	40H	
R		P	

Aufgaben und Übungen:

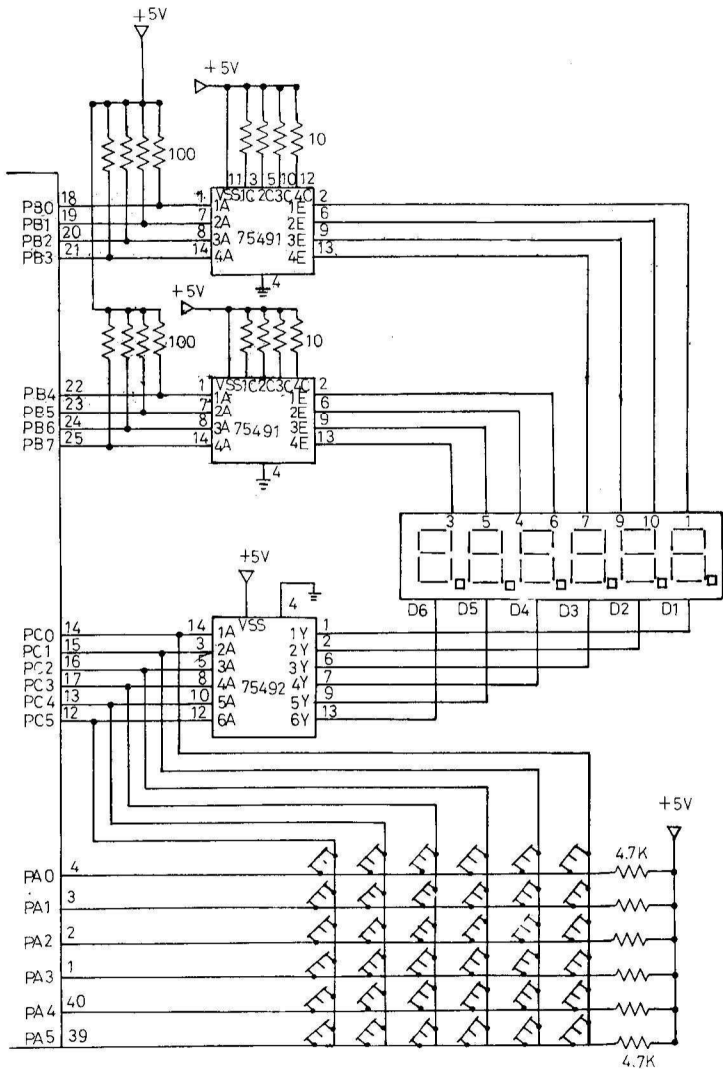
1. Laden und Abspeichern des Programms auf Cassette.
2. Ändern Sie den letzten Befehl in RST 38H um. Geben Sie ein 16-bit-Wort in das BC-Registerpaar und lassen Sie die Quadratwurzel berechnen.

Werte in BC	Ergebnis d. Programms	Überprüfung
0051H		
0000H		
FFFFH		
4000H		

3. Ändern Sie das Programm so um, daß die Quadratwurzel eines 32-bit-Wortes berechnet werden kann. Diese Zahl stehe in den Registern BC und IX. Die Wurzel – nur der ganzzahlige Anteil – soll in DE abgespeichert sein.
4. Erstellen Sie ein Programm zur Berechnung des Vectors D, bestehend aus den Komponenten H und L, d. h. die Längen sind in H und L gespeichert; das Ergebnis soll in D stehen.

$$D = \sqrt{H^2 + L^2}$$

Benutzen Sie die Unterprogramme für die Multiplikation und die Berechnung der Quadratwurzel.



Überblick über die Schaltung der Anzeige und der Tastatur

Die Anzeigeneinheit des MPF-1

Der MPF-1 benutzt zur Anzeige ein sechsstelliges 7-Segment-Display. Jede Anzeigestelle enthält 7 Leuchtdioden, die in Form von kleinen Rechtecken ausgebildet sind. Damit können alle Ziffern des Dezimalsystems, aber auch andere Zeichen mit gewissen Einschränkungen, dargestellt werden. Die 6 Stellen der Anzeige werden in einer Art Multiplexbetrieb angesteuert: Obwohl es so scheint, als wären alle 6 Stellen dauernd eingeschaltet, sind jeweils nur die Segmente einer einzigen Stelle der Anzeige eingeschaltet. Da jedoch die einzelnen Stellen sehr schnell hintereinander angesteuert werden, erscheint es dem Betrachter so, als leuchteten die Segmente aller Stellen ständig.

Die Einzelsegmente werden mit den Buchstaben a bis g bezeichnet, der Dezimalpunkt mit p. Alle Segmente a der sechs Stellen sind miteinander verbunden; ebenso alle anderen Segmente miteinander. Die Auswahl der Ziffernstelle erfolgt über eine besondere Auswahlleitung.

Soll zum Beispiel eine 7 an der 3. Stelle der Anzeigeneinheit dargestellt werden, so werden die Leitungen a, b und c aller 6 Ziffernstellen angesteuert, aber nur die Auswahlleitung für die 3. Stelle. Im nächsten Augenblick kann dann z. B. eine 3 an der 4. Stelle angezeigt werden, indem die Leitungen zu den Segmenten a, b, c, d und g angesteuert werden, aber nun die Auswahlleitung der 4. Stelle.

Das scheinbar feststehende Bild der Anzeige wird also durch ein schnelles "Durchfahren" der einzelnen Stellen erreicht. Mindestens 40mal pro Sekunde wird jede einzelne Stelle angewählt. Auf diese Weise entsteht der Eindruck eines "Stehenden Bildes".

Die Tastatur

In ähnlicher Weise wie die Auswahl der Anzeige geschieht auch die Abfrage der Tastatur. Das Betriebssystem, der "Monitor", fragt sehr schnell hintereinander jede einzelne Taste ab und stellt auf diese Weise fest, ob eine Taste gedrückt ist, und wenn ja, welche. Tastaturprellen wird durch die Gestaltung der Abfrage kompensiert.

Sowohl die Abfrage der Tastatur als auch die Anzeige werden hardwaremäßig über den Baustein 8255 bewerkstelligt. Dieses IC besitzt 24 Eingabe- bzw. Ausgabeterminals: PA0 - PA7, PB0 - PB7, PC0 - PC7

Über PB0 - PB7 werden — verstärkt von den beiden Treibern 75491 — die Segmente a - g und die Dezimalpunkte angesteuert.

Die Tore PC0 - PC5 sorgen über das IC 75492 für die richtige Auswahl der Ziffernstellen. — Die Tore PC6 und PC7 erfüllen andere Aufgaben: PC6 ist mit BREAK verbunden, und PC7 bedient den Lautsprecher und die grüne Leuchtdiode des MPF-1. Die Tore PA0 - PA5 dienen zusammen mit den Toren PC0 - PC5 für die Decodierung

der Tastatur (Matrix 6x6). PA6 ist mit der Taste USER KEY verbunden und PA7 dient zum Laden von Programmen von der Cassette. Zunächst wird von PC0 die äußerste rechte Spalte der Tastaturmatrix und dann mit PA0 bis PA5 festgestellt, ob eine der Tasten dieser Spalte gedrückt ist. Ist das nicht der Fall, so wird die nächste Spalte durch PC1 aktiviert und abgefragt. Auf diese Weise werden nacheinander alle Spalten und Zeilen, die gesamte Matrix, untersucht. Dieser Vorgang wird so lange wiederholt, bis eine gedrückte Taste gefunden ist. Dann gibt der jeweilige Wert der Tore PAX und PCX die Matrixstelle der gedrückten Taste an. Dieser Wert wird als "Positions-Code" bezeichnet, da er der Position der Taste entspricht, aber noch nicht ihrer eigentlichen Bedeutung. Die Routine im Monitorprogramm des MPF-1, die für die ständige Abfrage der Tastatur sorgt, nimmt diesen Wert des Positions-Code auf. Er wird dann der Startadresse einer Tabelle mit Namen KEYTAB, die im ROM-Bereich gespeichert ist, hinzuaddiert. Startadresse der Tabelle ist 077B. Auf diese Weise ergibt sich dann aus dem Positions-Code der "Interne Code". Aus diesem Code ergibt sich die eigentliche Bedeutung der jeweiligen Taste bzw. die Information, die zur Anzeige auf dem Siebensegmentdisplay führt.

Das Sieben-Segment-Format zur Darstellung von Ziffern, Buchstaben und Zeichen



bit	7	6	5	4	3	2	1	0
Segment	d	p	c	b	a	f	g	e

DISPLAY FORMAT:

CODE	8D	30	90	EA	36	AE	AF	3B	BF	BE	3F	A7	BD	B3
DATA	0	1	2	3	4	5	6	7	8	9	A	B	C	D
DISP	0	1	2	3	4	5	6	7	8	9	A	b	C	d

CODE	BF	0F	AD	37	B9	E1	97	85	2B	23	A3	1F	3E	03
DATA	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
DISP	E	F	G	H	I	J	K	L	M	N	O	P	Q	R

CODE	A6	B7	B5	B7	A9	07	B6	8A	B3	A2	32	02	C0	00
DATA	S	T	U	V	W	X	Y	Z	(	)	+	-	,	.
DISP	S	T	U	V	W	X	Y	Z	(	)	+	-	,	.

Zur Tabelle:

Diese Tabelle zeigt die Möglichkeiten der Darstellung von Ziffern, Buchstaben und Zeichen. Zusätzlich ist der Code eingetragen, der zur Darstellung der einzelnen Symbole führt. Jedem der 7 Segmente und dem Dezimalpunkt entspricht ein bestimmtes bit. Daraus ergibt sich der Code zur Darstellung der Symbole: Um eine 0 auf dem Display anzuzeigen, müssen alle Segmente außer g und dem Dezimalpunkt leuchten: Alle bit außer 1 und 6 müssen 1 sein.

10111101B = BDH

Auf diese Weise ist jedem Zeichen ein bestimmtes Bitmuster zugeordnet. Dieses Display-Format wird von einem Unterprogramm des Monitors erzeugt: Das Unterprogramm HEX7. Die Code-Worte für die 6stellige Anzeige werden im Display-Buffer 1FB6 - 1FBB zwischengespeichert. Kontrolliert und verwaltet wird dieser Zwischenspeicher vom Monitor-Unterprogramm SCAN.

Es folgen die Tabellen für den Positions-Code und den internen Code.

1. Positions-Code der einzelnen Tasten, berechnet vom Unterprogramm SCAN1:
Inhalt des Registers A.

SBR 1E	CBR 18	'0' 12	'1' 0C	'2' 06	'3' 00
'_' 1F	PC 19	'4' 13	'5' 0D	'6' 07	'7' 01
DATA 20	REG 1A	'8' 14	'9' 0E	'A' 08	'B' 02
'+' 21	ADDR 1B	'C' 15	'D' 0F	'E' 09	'F' 03
INS 22	DEL 1C	GO 16	STEP 10	0A	04
MOVE 23	RELA 1D	TPWR 17	TPRD 11	0B	05

2. Interner Code der einzelnen Tasten, berechnet vom Unterprogramm SCAN, abgelegt in Register A.
Positions-Code + 077B ist der Speicherplatz des internen Code. Beispiel: der Inhalt des Speicherplatzes 077B+3 ist 0F, der interne Code von 'F'.

SBR 15	CBR 1A	'0' 00	'1' 01	'2' 02	'3' 03
'-' 11	PC 18	'4' 04	'5' 05	'6' 06	'7' 07
DATA 14	REG 1B	'8' 08	'9' 09	'A' 0A	'B' 0B
'+' 10	ADDR 19	'C' 0C	'D' 0D	'E' 0E	'F' 0F
INS 16	DEL 17	GO 12	STEP 13	22	20
MOVE 1C	RELA 1D	TPWR 1E	TPRD 1F	23	21

Monitor-Unterprogramme des MPF-1

für Tastatur und Anzeige

Übersicht:

0624	SCAN1	1 Durchlauf der Tastaturabfrage und der Anzeige
05FE	SCAN	Tastaturabfrage und Displayanzeige bis neuer Tastendruck erfolgt
0689	HEX7	Umwandlung eines HEX-Zeichens in Siebensegmentformat
0678	HEX7SG	Umwandlung von 2 HEX-Zeichen in 7-Segmentformat
0665	ADDRDP	Umwandlung einer Adresse ins 7-Segmentformat: DE nach 1FB8-1FBB
0671	DATADP	Umwandlung der Daten-Anzeige: A nach 1FB6, 1FB7

1.	Unterprogramm	SCAN1
	Startadresse:	0624
	Funktion:	Durchläuft die Tastaturabfrage und die Anzeige einmal.
	Ausführungszeit:	10 ms (genau: 9,97 ms)
	Übernahme:	IX-Register zeigt auf die Adresse für den Anzeigen-Buffer der äußersten rechten Anzeigenstelle, IX+5 auf die linke Stelle
	Rückgabe:	a) Wenn keine Taste gedrückt ist, wird das Carry-flag auf 1 gesetzt. b) Ist eine Taste gedrückt, wird das Carry-flag gelöscht und der Positions-Code der Taste ins Register A geladen.

-
- | | | |
|----|---------------|---|
| 2. | Unterprogramm | SCAN |
| | Startadresse: | 05FE |
| | Funktion: | Ähnlich wie SCAN1 außer:
a) SCAN durchläuft die Tastaturabfrage und die Anzeige, bis ein neuer Tastendruck erfolgt.
b) SCAN liefert den internen Code der betreffenden Taste. |
| | Übernahme: | IX zeigt auf die Anfangsadresse des Anzeigen-Buffers. |
| | Rückgabe: | Register A enthält den internen Code der gedrückten Taste. |
| | | |
| 3. | Unterprogramm | HEX7 |
| | Startadresse: | 0689 |
| | Funktion: | Dekodiert eine Hexzahl ins 7-Segmentformat |
| | Übernahme: | Die niederwertigen 4 bit des Registers A enthalten die Hexzahl (0-F) |
| | Rückgabe: | Das Ergebnis steht ebenfalls im Register A. |
| | | |
| 4. | Unterprogramm | HEX7SG |
| | Startadresse: | 0678 |
| | Funktion: | Wandelt 2 Hexzahlen ins 7-Segmentformat |
| | Übernahme: | Die 1. Ziffer steht in den niederwertigen 4 bit, die 2. Ziffer in den höherwertigen 4 bit des Registers A. |
| | Rückgabe: | Die Adresse der Speicherstelle für den 7-Segment-Code der 1. Ziffer steht im HL-Registerpaar, für die 2. Ziffer in HL+1. Der Inhalt des HL-Registerpaares wird um 2 erhöht. |

Programmbeispiel 13:

Programmieren der Anzeige "HELP US" oder "HILFE!"

Beispiel 1: Auf der Anzeige des MPF-1 erscheint die Anzeige "HELP US" oder "HILFE!"; bei Betätigung der Taste —STEP— erfolgt ein Software-Halt, die rote LED leuchtet auf.

Auflistung des Programms:

1800	DD 21 20 18	LD IX,1820	;1820=Tabelle d. Zeichen
1804	CD FE 05 M1	CALL 05FE	;SCAN
1807	FE 13	CP 13H	;Taste -Step-
1809	20 F9	JR NZ,M1	
180B	76	HALT	

Tabelle der Zeichen:

	englisch / deutsch	
1820	AE S	/ 70 !
1821	B5 U	/ 8F E
1822	1F P	/ 0F F
1823	85 L	/ 85 L
1824	8F E	/ 30 I
1825	37 H	/ 37 H

Die äußerste rechte Stelle wird vom Unterprogramm SCAN zuerst angezeigt: IX+0, dann IX+1 usw. für die nächsten Stellen.

Aufgaben und Übungen:

1. Laden Sie das Programm und speichern Sie es auf Cassette.
2. Testen Sie beide Versionen der Anzeige.
3. Ändern Sie die Speicherstelle 1808 in 1A; jetzt erfolgt der Software-Halt durch Betätigen der Taste —CBR—. Warum?
4. Ändern Sie die Speicherstellen der Tabelle: 1821=B5, 1822=85, 1824=3F. Wie verändert sich dadurch die Anzeige "HILFE!"?

Beispiel 2: Die Anzeige "HALLO!" blinkt. Es wird das Unterprogramm SCAN1 benutzt. Jede Anzeige dauert 500ms, da SCAN1 50mal durchlaufen wird.

```

1800 21 26 18      LD HL,1826      ;keine Anzeige
1803 E5           PUSH HL
1804 DD 21 20 18   LD IX,1820      ;"HALLO!"
1808 DD E3         M1 EX (SP),IX
180A 06 32        LD B,32H        ;=50D
180C CD 24 06     M2 CALL 0624     ;SCAN1
180F 10 FB        DJNZ M2
1811 18 F5        JR M1

```

```

1820 70           `I`
1821 BD           `O`
1822 85           `L`
1823 85           `L`
1824 3F           `A`
1825 37           `H`
1826 00
1827 00
1828 00
1829 00
182A 00
182B 00

```

Bemerkung: Der Inhalt der Speicherstelle 180B legt die Blinkfrequenz fest. Sie kann durch Einschreiben eines anderen Wertes verändert werden.

Aufgaben und Übungen:

1. Laden Sie auch dieses Programm in den MPF-1 und speichern Sie es auf Cassette.
2. Ändern Sie die Speicherstelle 180B in 01. Was geschieht?
3. Ändern Sie 180B in 05.
4. Ändern Sie die Blinkfrequenz so, daß die Anzeige ca. 2 Sekunden lang aufleuchtet.

Beispiel 3: Es wird der Tasten- oder Positions-Code der gedrückten Taste angezeigt.

```
1800 DD 21 00 19 LD IX,1900 ;Ausgabe-Puffer
1804 CD 24 06 M1 CALL 0624 ;SCAN1
1807 38 FB JR C,M1
1809 21 00 19 LD HL,1900
180C CD 78 06 CALL 0678 ;HEX7SG
180F 18 F3 JR M1
```

Wird eine Taste gedrückt, so erscheint deren Positions-Code. Soll der interne Code der Taste angezeigt werden, so ist das Programm wie folgt zu ändern:

```
1800 DD 21 00 19 LD IX,1900
1804 CD FE 05 M1 CALL 05FE ;SCAN
1807 21 00 19 LD HL,1900
180D 18 F5 JR M1
```

Aufgaben:

1. Ändern Sie die Werte des Ausgabe-Buffers 1900 - 1905 in FF. Was erscheint jetzt in der Anzeige?
2. Welche Bedeutung haben der Positions-Code und der interne Code?

Beispiel 4: Wandelt 3 aufeinander folgende byte in 7-Segmentformat, speichert diese Werte in 1903 - 1908 und bringt sie zur Anzeige.

```
1800 11 00 19 LD DE,1900 ;3 byte in 1900 ff
1803 21 03 19 LD HL,1903 ;Anzeige-Puffer
1806 06 03 LD B,3
1808 1A M1 LD A,(DE)
1809 CD 78 06 CALL 0678 ;HEX7SG
180C 13 INC DE
180D 10 F9 DJNZ M1
180F DD 21 03 19 LD IX,1903
1813 CD FE 05 CALL 05FE ;SCAN
1816 76 HALT
```

Die 3 byte, die ins 7-Segmentformat umgerechnet werden sollen, werden an die Speicherstellen 1900 - 1902 geschrieben. Man kann in der Befehlszeile 180F einen

BREAKPOINT setzen, um zu überprüfen, ob die Umwandlung korrekt durchgeführt worden ist.

Aufgabe: Ändern Sie die Inhalte von 1900 - 1902 so um, daß die Anzeige '333446' erscheint.

Programmbeispiel 14:

Lauffeuer

Ein anderes Programm zur Demonstration der Programmierung der Anzeigeneinheit ist das Programm "Lauffeuer". Dabei werden die äußeren Segmente des Anzeigefeldes ringsum nacheinander in schneller Folge zum Aufleuchten gebracht.

1800	21 40 18	M1	LD HL,1840	;Tabelle
1803	DD 21 00 19		LD IX,1900	;Anzeigen-Puffer
1807	CD 30 18	M2	CALL 1830	;Puffer löschen
180A	5E		LD E,(HL)	;Tabellenwert holen 1.
180B	1C		INC E	;=FFH? für Wiederholung
180C	28 F2		JR Z,M1	;wenn ja, von vorne
180E	1D		DEC E	;wenn nicht, E=alter Wert
180F	16 00		LD D,0	;E=Displaystelle
1811	DD 19		ADD IX,DE	
1813	23		INC HL	
1814	7E		LD A,(HL)	;Tabellenwert holen 2.
1815	DD 77 00		LD (IX),A	;in Anzeigepuffer laden
1818	DD 21 00 19		LD IX,1900	
181C	06 03		LD B,3	;Geschwindigkeit
181E	CD 24 06	M3	CALL 0624	;SCAN1
1821	38 01		JR C,M4	
1823	4F		LD C,A	;Taste in C speichern
1824	10 F8	M4	DJNZ M3	
1826	79		LD A,C	
1827	FE 10		CP 10	;Taste -STEP-?
1829	28 02		JR Z,M5	;Stopp
182B	23		INC HL	
182C	23		INC HL	
182D	2B	M5	DEC HL	
182E	18 D7		JR M2	
1830	06 06		LD B,6	
1832	DD 36 00 00	M6	LD (IX),0	
1836	DD 23		INC IX	
1838	10 F8		DJNZ M6	
183A	11 FA FF		LD DE,-6	;alten Wert von IX
183D	DD 19		ADD IX,DE	;wiederherstellen
183F	C9		RET	

Tabelle: Das 1. byte gibt die angewählte Ziffernstelle an, das 2. das Segment, das an dieser Stelle aufleuchten soll.

1840	05	Anzeigenstelle Nr. 5 = äußerste linke Stelle
1841	08	Segment a
1842	04	2.Stelle von links
1843	08	Segment a
1844	03	3. Stelle
1845	08	Segment a
1846	02	
1847	08	
1848	01	
1849	08	
184A	00	Stelle Nr. 0 = rechts außen
184B	08	Segment a
184C	00	
184D	10	Segment b
184E	00	
184F	20	Segment c
1850	00	
1851	80	Segment d
1852	01	
1853	80	
1854	02	
1855	80	
1856	03	
1857	80	
1858	04	
1859	80	
185A	05	
185B	80	
185C	05	
185D	01	Segment e
185E	05	
185F	04	Segment f
1860	FF	Codewort für die Wiederholung

Aufgaben und Übungen:

1. Laden Sie das Programm und speichern Sie es auf Cassette.
2. Ändern Sie das Programm so um, daß das Lauffeuer im Gegenuhrzeigersinn umläuft.

-
3. Ändern Sie den Inhalt der Speicherstelle 1828; was bewirkt diese Änderung?
 4. Was wird durch eine Änderung des Speicherplatzes 181D bewirkt?
 5. Schreiben Sie ein Programm, das eine andere Bewegung der aufleuchtenden Segmente zur Folge hat.
 6. Schreiben Sie ein Programm, das für ca. 20s das Wort "HILFE!" anzeigt und dann in das Lauffeuer übergeht.

Programmbeispiel 15:

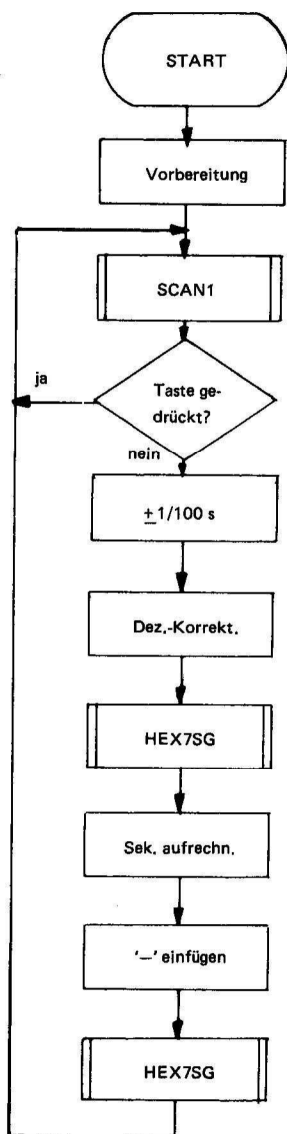
Stopp-Uhr

In diesem Programm soll per Software eine Stoppuhr für hundertstel Sekunden realisiert werden. Dazu wird die Taktfrequenz des Systems zugrunde gelegt. Diese bestimmt die Dauer der Unterprogramme und Schleifen, die so berechnet sind, daß sich eine annähernde Genauigkeit ergibt.

Durch Drücken einer Taste wird die Stoppuhr angehalten.

Auflistung des Programms:

1800	DD 21 00 19	LD IX,1900	;Anfangswert
1804	11 00 00	LD DE,0	;Anzeige mit 0 beginnen
1807	CD 24 06 M1	CALL 0624	;SCAN1
180A	30 FB	JR NC,M1	;Taste gedrückt?
180C	7B	LD A,E	;wenn nicht, Hundertst.
180D	C6 01	ADD A,1	;um 1 erhöhen
180F	27	DAA	
1810	5F	LD E,A	
1811	7A	LD A,D	;wenn ja wieder um 1
1812	CE 00	ADC A,0	;erniedrigen
1814	27	DAA	
1815	57	LD D,A	
1816	7B	LD A,E	
1817	21 00 19	LD HL,1900	;Anzeige-Puffer
181A	CD 78 06	CALL 0678	;HEX7SG
181D	36 02	LD (HL),2	;`-` einfügen
181F	23	INC HL	
1820	7A	LD A,D	;Sekunden umwandeln
1821	CD 78 06	CALL 0678	;HEX7SG
1824	36 00	LD (HL),0	;ausschalten d.li Stellen
1826	18 DF	JR M1	
1900			;Anzeige-Puffer



Aufgaben und Übungen:

1. Laden Sie das Programm und starten Sie es.
2. Betätigen Sie irgendeine Taste außer —RS— und —MONI—.
3. Beachten Sie, daß es sich bei diesem Programm um eine "Endlosschleife" handelt. Es kann nur durch die Tasten —RS— und —MONI— abgebrochen werden.
4. Ändern Sie das Programm so um, daß die Stoppuhr nur die Zehntelsekunden und die Sekunden anzeigt. Das Programm soll zu Beginn alle Nullen anzeigen.
5. Sorgen Sie dafür, daß die Stoppuhr von einer bestimmten Taste gestoppt wird.
6. Überprüfen Sie die Genauigkeit der Stoppuhr. Geben Sie den Grund für evt. Abweichungen an.

Programmbeispiel 16:

Uhr 1

Das Programm für die vollständige Anzeige der Uhrzeit in Stunden, Minuten und Sekunden basiert ebenfalls auf dem Systemtakt von 560 ns entspr. 1,79 MHz.
Die Summe der Taktzyklen für einen Durchlauf ist genau berechnet:

SCAN: 17812

Schleife 1: $(17+17812+13) \times 100 - 5 = 1784195$

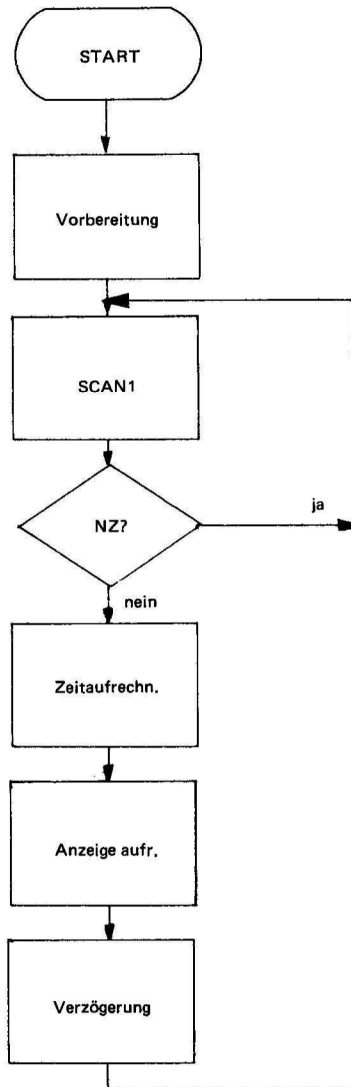
TMUPDT: (Aufrechnen der Zeit): 258

BFUPDT: (Aufrechnen des Buffers): 914

Schleife 2: $(4+13) \times 256 - 5 = 4347$

Summe: 1789755 Taktzyklen

$560 \text{ ns} \times 1789755 = 1,002 \text{ Sekunden}$



Auflistung des Programms:

1800	F3	DI	;Interrupt nicht zugelass.
1801	DD 21 03 1A	LD IX, 1A03	;Anzeige-Puffer

Es folgt das Programmteil ONESEC, dessen Ausführungszeit 1 Sekunde beträgt.
Es besteht aus 3 Unterprogrammen und einem zusätzlichen Verzögerungsteil.

1805	06 64	M3	LD B, 64H	;100D
1807	CD 24 06	M1	CALL 0624	;SCAN1
180A	10 FB		DJNZ M1	;1784195 Taktzyklen
180C	CD 17 18		CALL 1817	;TMUPDT = 275 T
180F	CD 2F 18		CALL 182F	;BFUPDT = 931 T
1812	00	M2	NOP	
1813	10 FD		DJNZ M2	;4347 T
1815	18 EE		JR M3	;ONESEC

Unterprogramm TMUPDT (Aufrechnen des Zeit-Buffers). Die Ausführungszeit beträgt unter allen Bedingungen 275 Taktzyklen.

1817	21 47 18		LD HL, 1847	;Tabelle der Max-Werte
181A	11 00 1A		LD DE, 1A00	;Sekunden
181D	06 03		LD B, 3	
181F	37		SCF	;Carry setzen
1820	1A	M4	LD A, (DE)	
1821	CE 00		ADC A, 0	;+1
1823	27		DAA	
1824	12		LD (DE), A	
1825	96		SUB (HL)	;Vergl. m. Max-Wert
1826	38 01		JR C, M5	
1828	12		LD (DE), A	
1829	3F	M5	CCF	;Carry invertieren
182A	23		INC HL	
182B	13		INC DE	
182C	10 F2		DJNZ M4	
182E	C9		RET	

Unterprogramm BFUPDT (Aufrechnen des Anzeigen-Buffers)
Ausführungszeit: 914 Taktzyklen.

182F	21 03 1A		LD HL, 1A03	;Anzeigen-Puffer
1832	11 00 1A		LD DE, 1A00	;Sekunden
1835	06 03		LD B, 3	
1837	1A	M6	LD A, (DE)	

1838	CD 6D 06	CALL 066D	;HEX7SG
183B	13	INC DE	
183C	10 F9	DJNZ M6	
183E	2B	DEC HL	
183F	2B	DEC HL	
1840	CB F6	SET 6,(HL)	;Dezimalp. setzen
1842	2B	DEC HL	
1843	2B	DEC HL	
1844	CB F6	SET 6,(HL)	
1846	C9	RET	

Tabelle der Maximalwerte für Sekunden, Minuten und Stunden:

1847	60
1848	60
1849	12 / 24

Zeit-Buffer für Sekunden, Minuten und Stunden. Hier kann die aktuelle Zeit eingegeben werden.

1A00	Sekunden
1A01	Minuten
1A02	Stunden

Die folgenden Speicherstellen bilden den Anzeigen-Buffer.

1A03 ff

Aufgaben und Übungen:

1. Laden Sie das Programm. Tragen Sie die aktuelle Zeit in den Zeit-Buffer 1A00 – 1A02 ein und starten Sie das Programm.
2. Stellen Sie die Uhr auf 24-Stunden-Modus um.

Programmbeispiel 17:

Uhrenprogramm mit dem CTC-Baustein

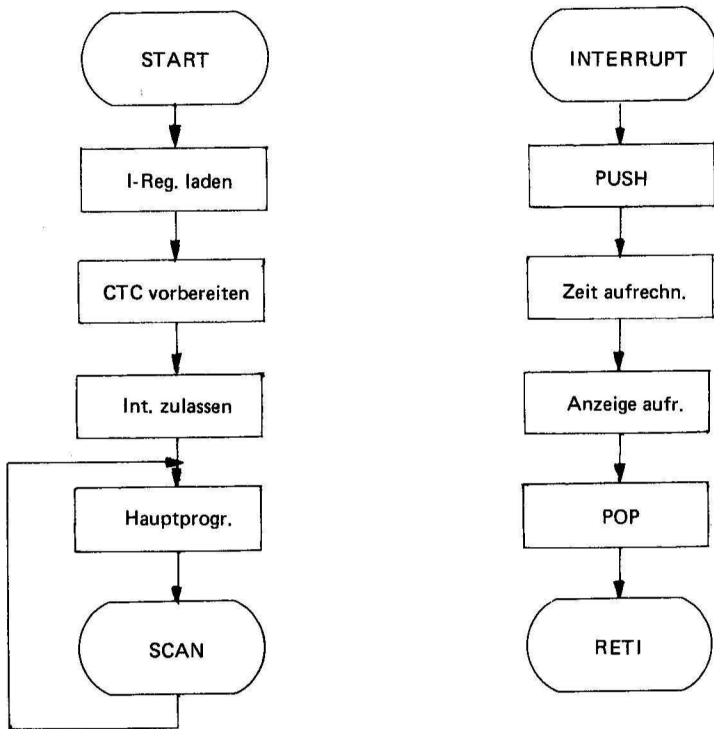
Dieses Programm benutzt den Zähler- und Zeitgeberbaustein Z80-CTC. Dieser erzeugt einen Interrupt zur Zeitsteuerung. Auf den Aufbau des CTC-Bausteins sei hier nicht näher eingegangen. Detaillierte Beschreibungen enthalten die entsprechenden Handbücher. Die Adressen des CTC beim MPF-1 sind 40H — 43H. Diese entsprechen den 4 Kanälen.

Beim vorliegenden Uhrenprogramm wird das Interrupt-Register I der CPU mit 18H geladen. Das ist das höherwertige byte des Interruptvectors. Dann wird der CTC-Baustein wie folgt programmiert: In Adresse 40 H für den Kanal 0 wird der Wert 10110101B = B5H eingeschrieben; damit wird das Channel-Control-Register programmiert:

- bit 7 = 1 bedeutet, daß mit einem Interrupt gearbeitet werden soll.
- bit 6 = 0 der Baustein soll als Zeitgeber arbeiten, nicht als Zähler.
- bit 5 = 1 der Faktor des Vorteilers ist 256, nicht 16
- bit 4 = 1 die positive Flanke des Taktsignals startet den Zeitgeber
- bit 3 = 0 der Zeitgeber beginnt bei der positiven Flanke desjenigen Taktimpulses zu arbeiten, der der Eingabe der Zeitkonstanten folgt.
- bit 2 = 1 die Zeitkonstante wird mit dem nächsten Kontrollwort übertragen.
- bit 1 = 0 Reset-bit
- bit 0 = 1 dieses bit muß immer 1 sein.

Anschließend wird wie angekündigt das Kontrollwort für die Zeitkonstante in Adresse 40H eingetragen. Es ist hier der Wert 20H = 32D. Damit erzeugt der CTC-Baustein nach jeweils $256 \times 32 = 8192$ Taktzyklen einen Interrupt. Das 3. Kontrollwort beeinflusst das Interrupt-Vector-Register:

Der Wert A8H bezeichnet das niederwertige byte des Interrupt-Vectors, während das I-Register der CPU das höherwertige byte enthält. Dadurch ergibt sich die Adresse 18A8. Dieser sogenannte "Vector" ist nicht selbst die Startadresse, sondern zeigt auf die Startadresse des Interrupt-Programms. Die Speicherstelle 18A8 enthält das niederwertige, die nächste Speicherstelle 18A9 das höherwertige byte der Startadresse des Interruptprogramms: 18AA.



Auflistung des Programms

1800	3E 18	LD A,18H	;I-Register laden
1802	ED 47	LD I,A	
1804	3E B5	LD A,B5	;Kontroll-Reg. laden
1806	D3 40	OUT 40,A	
1808	3E 20	LD A,20	;Konstante laden
180A	D3 40	OUT 40,A	
180C	3E A8	LD A,A8	;Interrupt-Vector lad.
180E	D3 40	OUT 40,A	
1810	ED 5E	IM 2	; Interrupt-Mode 2
1812	FB	EI	

Hauptprogramm:

1813	DD 21 04 1A M1	LD IX,1A04	;Anzeigen-Puffer
1817	CD FE 05	CALL 05FE	;SCAN
181A	18 F7	JR M1	

Unterprogramm Zeit-Buffer aufrechnen (ATB):

181C	11 00 1A	LD DE,1A00	;Zeit-Puffer
181F	1A	LD A,(DE)	
1820	3C	INC A	
1821	12	LD (DE),A	
1822	FE DA	CP DA	;Vergl. m.DAH=218d
1824	06 04	LD B,4	
1826	C0	RET NZ	
1827	AF	XOR A	
1828	05	DEC B	
1829	12	LD (DE),A	
182A	13	INC DE	
182B	21 53 18	LD HL, 1853	;Tabelle d. Max-Werte
182E	1A	M2 LD A,(DE)	
182F	C6 01	ADD A,1	
1831	27	DAA	
1832	12	LD (DE),A	
1833	96	SUB (HL)	;vergl.m.Max-Wert
1834	D8	RET C	
1835	12	LD (DE),A	
1836	23	INC HL	
1837	13	INC DE	
1838	10 F4	DJNZ M2	
183A	C9	RET	

Unterprogramm Anzeigen-Buffer aufrechnen (SETDB):

```

183B  21 04 1A      LD HL,1A04      ;Anzeigen-Puffer
183E  11 01 1A      LD DE,1A01      ;Sekunden
1841  06 03          LD B,3
1843  1A              LD A,(DE)      M3
1844  CD 78 06      CALL 0678        ;HEX7SG
1847  13              INC DE
1848  10 F9          DJNZ M3
184A  2B              DEC HL
184B  2B              DEC HL
184C  CB F6          SET 6,(HL)      ;Dezimalpunkt Std.
184E  2B              DEC HL
184F  2B              DEC HL
1850  CB F6          SET 6,(HL)      ;Dezimalpunkt Min.
1852  C9              RET

```

Es folgt die Tabelle der Höchstwerte:

```

1853  60
1854  60
1855  12 / 24

```

Interrupt-Vector:

```

18A8  AA 18

```

Das Interruptprogramm wird bei jedem Interrupt des CTC aufgerufen und abgearbeitet:

```

18AA  F5              PUSH AF
18AB  C5              PUSH BC
18AC  D5              PUSH DE
18AD  E5              PUSH HL
18AE  CD 1C 18      CALL 181C        ;Zeitpuffer aufrechnen
18B1  78              LD A,B
18B2  FE 04          CP 4
18B4  C4 3B 18      CALL NZ,183B      ;Anzeigepuffer aufrechnen
18B7  E1              POP HL
18B8  D1              POP DE
18B9  C1              POP BC
18BA  F1              POP AF
18BB  FB              EI
18BC  ED 4D          RETI

```

Die aktuelle Zeit wird in 1A01 – 1A03 eingetragen.

Bemerkungen zum Programm:

Es wird die Betriebsart Zeitgeber des CTC-Bausteins angewandt. Die Anzahl der Taktzyklen errechnet sich wie folgt:

Der Vorteiler wird auf 256 eingestellt, die Zeitgeber-Konstante auf $20H = 32D$. Die Anzahl der Interrupts pro Sekunde beträgt 218. Daraus ergeben sich $256 \times 32 \times 218 = 1785856$ Taktzyklen. Das entspricht einer Abweichung von 3916 Taktzyklen; der Fehler pro Sekunde beträgt 2,2ms oder +1 Sekunde nach 455 Sekunden.

Aufgaben und Übungen:

1. Laden Sie das Programm und speichern Sie es auf Cassette.
2. Ändern Sie den Inhalt der Speicherstelle 1823 in 6DH. Was geschieht?
3. Was ist zu tun, wenn Sie den Kanal 3 des CTC-Bausteins benutzen wollen?
4. Was geschieht, wenn die Höchstwerte in 1853 - 1855 geändert werden?
5. Die Ungenauigkeit beträgt 2,2ms. Wie könnte man die Genauigkeit noch erhöhen?

Programmbeispiel 18:

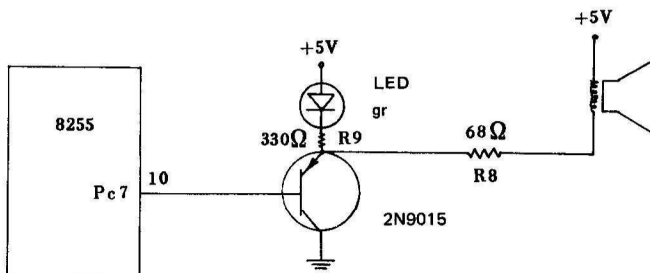
Telefon-Ton

Dieses Programm bildet das Klingeln eines Telefons nach. Es wird hier das Unterprogramm TONE aus dem Monitorprogramm des MPF-1 benutzt. Das Klingeln eines Telefons kann nachgebildet werden durch die Wiederholung von einer Sekunde Ton und nachfolgend 2 Sekunden Pause. Der Ton wird dadurch erzeugt, daß für jeweils 25ms (Halbperiode von 20 Hz) abwechselnd ein Ton von 320 Hz und ein Ton von 480 Hz gebildet wird.

Im vorgestellten Programm kontrolliert das Register C die Tonfrequenz und das Registerpaar HL die Länge des Tones.

- a) niedrige Frequenz: 320 Hz; C = 211; HL = 8
Dadurch ergeben sich $(44+13 \times 211) \times 2 \times 560 \text{ ns} = 3,121 \text{ ms}$
Die Frequenz beträgt $1/3,121 = 320 \text{ Hz}$.
Die Länge des Tones ist $3,121 \text{ ms} \times 8 = 25 \text{ ms}$.
- b) Hohe Frequenz: 480 Hz; C = 140; HL = 12.
So ergibt sich $(44+13 \times 140) \times 2 \times 560 \text{ ns} = 2,087 \text{ ms}$.
Die Frequenz beträgt also $1/2,087 = 480 \text{ Hz}$.
Die Länge des Tones ist $2,087 \text{ ms} \times 12 = 25 \text{ ms}$.

Der Ton wird vom Tor PC7 des Ein- Ausgabe-Bausteins 8255 über einen Transistor an den eingebauten Lautsprecher und die grüne Leuchtdiode des MPF-1 gegeben.



Auflistung des Programms

1800	3E 14	M1	LD A,14H	;20Hz
1802	08	M2	EX AF,AF	;Accu retten
1803	0E D3		LD C,D3H	;=211D
1805	21 08 00		LD HL,8	
1808	CD E4 05		CALL 05E4	;Unterprogr. TONE
180B	0E 8C		LD C,8CH	;=140D
180D	21 0C 00		LD HL,CH	;=12D
1810	CD E4 05		CALL 05E4	;TONE
1813	08		EX AF,AF	
1814	3D		DEC A	
1815	20 EB		JR NZ,M2	
1817	01 50 C3		LD BC,C350H	;50000D
181A	CD 1F 18		CALL 181F	;2 Sekunden Pause
181D	18 E1		JR M1	

Unterprogramm Verzögerung: Durch dieses Unterprogramm wird eine Verzögerung von 40 Mikrosekunden erreicht. Da der Zähler BC von 50000 auf 0 erniedrigt wird, ergibt sich eine Gesamtdauer von 2 Sekunden.

181F	E3	M3	EX (SP),HL	;19 T
1820	E3		EX (SP),HL	;19 T
1821	ED A1		CPI	;16 T
1823	E0		RET P0	;5 T
1824	18 F9		JR M3	;12 T

Aufgaben und Übungen:

1. Laden und speichern Sie das Programm.
2. Klingt es wie das Klingeln eines Telefons? Wenn nicht, versuchen Sie, die Frequenzen anzugleichen.
3. Versuchen Sie, das Besetztzeichen nachzubilden.
Hinweis: 0,5 Sekunden 400 Hz und 0,5 Sekunden Pause.

Programmbeispiel 19:

Martinshorn

Durch dieses einfache und kurze Programm wird der Klang des Martinshorns nachgebildet. Für jeweils 0,73 Sekunden werden abwechselnd die Frequenzen 265 Hz und 352 Hz erzeugt. Bei der niedrigen Frequenz wird das Register C mit 00H – hier entspr. 256 – geladen und das HL-Registerpaar mit COH = 192D: $(44+13 \times 256) \times 2 \times 560 = 3,777\text{ms}$; die Frequenz ist also $1/3777 = 265\text{ Hz}$. Die Dauer des Tones errechnet sich aus der Schwingungsdauer und dem Wert von HL: $3,777 \times 192 = 0,73\text{ Sekunden}$.

Um die höhere Frequenz von 352 Hz zu erzeugen, wird das C-Register mit COH = 192D geladen: $(44+13 \times 192) \times 2 \times 560\text{ns} = 2,845\text{ms}$. Die Dauer des Tones wird durch den Inhalt von HL bestimmt: HL = 256D=100H. $2,845\text{ms} \times 256 = 0,73\text{ Sekunden}$.

Auflistung des Programms:

1800	0E 00	M1	LD C,0	
1802	21 C0 00		LD HL,00C0	
1805	CD E4 05		CALL 05E4	;Unterprogr. TONE
1808	0E C0		LD C,C0	
180A	21 00 10		LD HL,0100	
180D	CD E4 05		CALL 05E4	
1810	18 EE		JR M1	

Programmbeispiel 20:

Eine Microcomputer-Orgel

In diesem Programm werden den Tasten O - F die Töne von zwei Oktaven zugeordnet. Eine Übersicht gibt das Schaubild. Jeder Ton wird so lange angehalten, wie die Taste gedrückt ist.

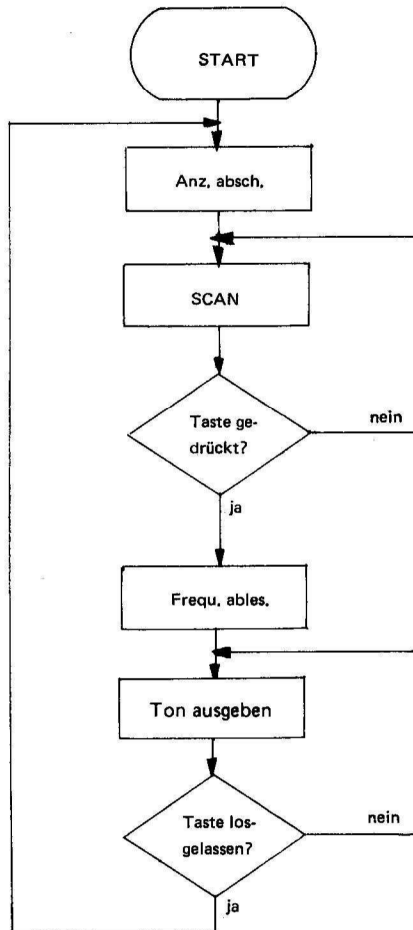
Die Werte für die Erzeugung der einzelnen Töne sind in einer Tabelle eingetragen: 1823 - 1832. Wird die Frequenz eines Tones verdoppelt, so wird er um eine Oktave höher. Die Frequenzen der Halbtonschritte ergeben sich aus der Frequenz des vorhergehenden Tones durch eine Multiplikation mit 1,059 (genauer: 12. Wurzel aus 2)

Beispiel: Die Frequenz von E ist 503 Hz; dann ist die Frequenz von F gleich $503 \times 1,059 = 532 \text{ Hz}$.

Taste Note	C 'G'	D 'A'	E 'H'	F 'C'
	8 'C'	9 'D'	A 'E'	B 'F'
	4 'F'	5 'G'	6 'A'	7 'H'
	0 'H'	1 'C'	2 'D'	3 'E'

Soll F erhöht werden, um ein Lied in G-Dur zu spielen, so läßt sich der Tabellenwert für das Fis folgendermaßen berechnen:

'F' hat den Tabellenwert 7EH = 126D; er stellt die Zeitkonstante des Tones dar. Daher muß dieser Wert durch 1,059 dividiert werden, um den Tabellenwert für das Fis zu erhalten:
 $126 : 1,059 = 118,9$
Der Wert 77H = 119D wird anstelle von 7E in die Tabelle eingetragen, um die G-Dur-Tonleiter zu erhalten.



Auflistung des Programms:

1800	DD 21 A5 07 M1	LD IX,07A5	;Anzeige abschalten
1804	CD FE 05	CALL 05FE	;SCAN
1807	21 23 18	LD HL,1823	;Tabelle der Frequ.
180A	85	ADD A,L	
180B	6F	LD L,A	;A+HL
180C	3E C0	LD A,C0	;11000000B
180E	DE 02	M2 OUT 02,A	;Tastatur aktivieren
1810	46	LD B,(HL)	;Wert v Tabelle holen
1811	00	M3 NOP	
1812	00	NOP	;Verzögerung
1813	00	NOP	
1814	10 FB	DJNZ M3	
1816	EE 80	XOR 80H	;bit 7 invert.
1818	4F	LD C,A	
1819	DB 00	IN A,00	;Taste gedrückt?
181B	F6 C0	OR C0	;bit 6 u 7 = 1
181D	3C	INC A	
181E	79	LD A,C	
181F	28 DF	JR Z,M1	
1821	18 EB	JR M2	

Tabelle der Frequenzwerte:

1823	B2	Ton H	;Taste '0'
1824	A8	C	'1'
1825	96	D	'2'
1826	85	E	'3'
1827	7E	F	'4'
1828	70	G	'5'
1829	64	A	'6'
182A	59	H	'7'
182B	54	C	'8'
182C	4A	D	'9'
182D	42	E	'A'
182E	3E	F	'B'
182F	37	G	'C'
1830	31	A	'D'
1831	2C	H	'E'
1832	29	C	'F'

Aufgaben und Übungen:

1. Laden Sie das Programm und speichern Sie es auf Cassette.
2. Überprüfen Sie, ob alle Töne richtig klingen.
3. Versuchen Sie ein Lied zu spielen.
4. Erweitern Sie das Programm so, daß weitere Tasten der Tastatur für zusätzliche Töne benutzt werden können.

Programmbeispiel 21:

Musik-Box

Das folgende Programm spielt ein Lied selbständig ab. Dazu werden zwei Tabellen benutzt, die nachstehend näher beschrieben sind.

1. Tabelle der Frequenzen:

Jedes Element der Tabelle besteht aus 2 byte; das erste enthält den Wert für die Tonhöhe, das zweite die Anzahl der Halbperioden für die Dauer des Tones.

2. Die Lied-Tabelle:

Auch hier hat jedes Element 2 byte; das erste enthält den Code für Ton oder Pause oder den Befehl für Wiederholung oder Stopp: Dabei haben die einzelnen bit folgende Bedeutung:

bit-Nr	7	6	5	4 - 0
Bedeut.	Stopp	Wiederh	Pause	Notenwert

Das zweite byte enthält die Anzahl der Zeiteinheiten, d. h. die Notenlänge.

Ein Beispiel: Die Frequenz für das 'C' ist in der Frequenzen-Tabelle als 'Code 05' festgelegt:

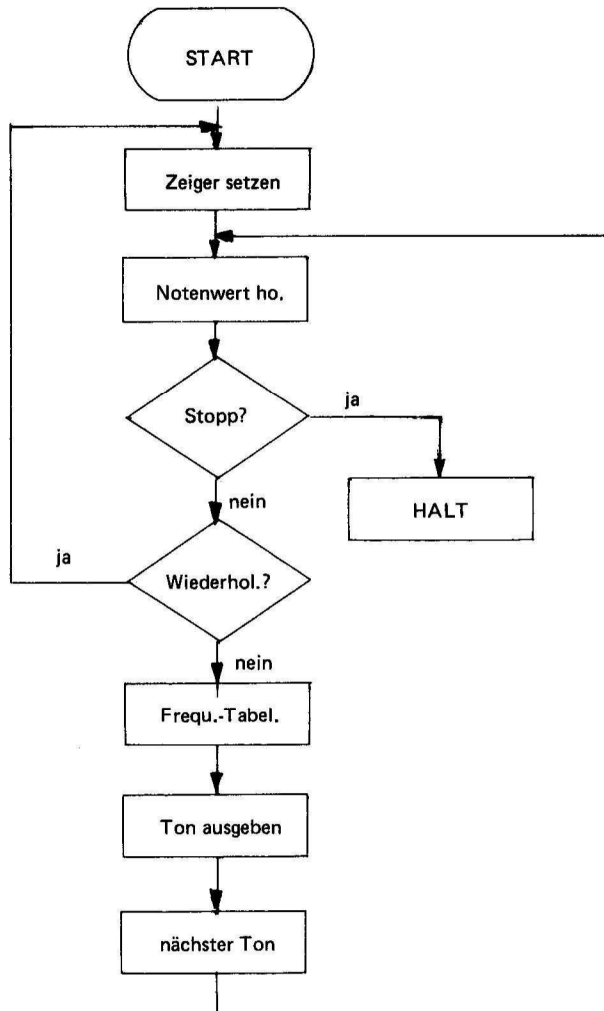
1845	AB	Tonhöhe
1846	20	Anzahl der Halbperioden

In der Lied-Tabelle steht der oben genannte Wert 05 für das C (erstes byte); im folgenden byte wird die aktuelle Notenlänge angegeben: $n \times 0,077$ Sekunden.

80H bedeutet das Ende des Liedes

40H bewirkt die Wiederholung des Liedes;

20H steht für eine Pause; das folgende byte gibt die Länge der Pause an wie bei den Noten: $n \times 0,077$ Sekunden.



Auflistung des Programms:

1800	DD 21 80 18 M1	LD IX,1880	;Lied-Tabelle
1804	DD 7E 00 M2	LD A,(IX)	
1807	87	ADD A,A	;Jede Note=2byte
1808	38 30	JR C,M3	;Stopp?
180A	FA 00 18	JP M,M1	;Wiederholung?
180D	0E 00	LD,C0	;bit 7 zurücksetzen
180F	CB 77	BIT 6,A	;Pause?
1811	20 02	JR NZ,M4	
1813	CB F9	SET 7,C	;bit 7 = 1
1815	E6 3F M4	AND 3F	;Notenwert
1817	21 3B 18	LD HL,183B	;Frequenztabelle
181A	85	ADD A,L	
181B	6F	LD L,A	;Zeiger auf Tabellenwert
181C	5E	LD E,(HL)	;Zähler d Halbperioden
181D	23	INC HL	
181E	56	LD D,(HL)	;Zähler pro Zeiteinh.
181F	DD 23	INC IX	
1821	DD 66 00	LD H,(IX)	
1824	3E FF	LD A,FF	

Die folgende Schleife erzeugt jeweils einen Ton oder eine Pause.

1826	6A	M5	LD L,D	
1827	D3 02	M6	OUT (02),A	
1829	43		LD B,E	
182A	00	M7	NOP	;Verzögerung Bx25-5 T
182B	00		NOP	
182C	00		NOP	
182D	10 FB		DJNZ M7	
182F	A9		XOR C	;C=80=Ton,C=00=Pause
1830	2D		DEC L	
1831	20 F4		JR NZ,M6	
1833	25		DEC H	
1834	20 F0		JR NZ,M5	

Die Note oder die Pause ist ausgeführt; der Zeiger wird auf den nächsten Tabellenwert eingestellt:

1836	DD 23	INC IX
1838	18 CA	JR M2
183A	76	HALT

Tabelle der Frequenzen:

183B	E1 18	Code 00	`G`
183D	D4 1A	01	`Gis`
183F	C8 1B	02	`A`
1841	BD 1D	03	`Ais`
1843	B2 1E	04	`H`
1845	AB 20	05	`C`
1847	9F 22	06	`Cis`
1849	96 24	07	`D`
184B	8D 26	08	`Dis`
184D	85 29	09	`E`
184F	7E 2B	0A	`F`
1851	77 2E	0B	`Fis`
1853	70 31	0C	`G`
1855	6A 33	0D	`Gis`
1857	64 37	0E	`A`
1859	5E 3A	0F	`Ais`
185B	59 3D	10	`H`
185D	54 41	11	`C`
185F	4F 45	12	`Cis`
1861	4A 49	13	`D`
1863	46 4D	14	`Dis`
1865	42 52	15	`E`
1867	3E 57	16	`F`
1869	3B 5C	17	`Fis`
186B	37 62	18	`G`
186D	34 67	19	`Gis`
186F	31 6E	1A	`A`
1871	2E 74	1B	`Ais`
1873	2C 7B	1C	`H`
1875	29 82	1D	`C`
1877	27 8A	1E	`Cis`
1879	25 92	1F	`D`

Lied-Tabelle "JINGLE BELL" (nur ein kurzes Stück)

1880	09	`E`
1881	04	
1882	09	`E`
1883	04	
1884	09	`E`
1885	06	
1886	20	Pause
1887	02	
1888	09	`E`
1889	04	
188A	09	`E`
188B	04	
188C	09	`E`
188D	06	
188E	20	Pause
188F	02	
1890	09	`E`
1891	04	
1892	0C	`G`
1893	04	
1894	05	`C`
1895	04	
1896	07	`D`
1897	04	
1898	09	`E`
1899	08	
189A	20	Pause
189B	08	
189C	80	Stopp

Lied-Tabelle eines deutschen Volksliedes als Speicherausdruck:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
188—	05	04	07	04	09	04	0A	04	0C	04	0E	04	0C	08	0A	04
189—	07	04	10	08	0C	04	09	04	11	08	05	04	07	04	09	04
18A—	0A	04	0C	04	0E	04	0C	08	0A	04	07	04	10	04	13	04
18B—	11	10	0E	08	0E	08	11	08	10	04	0E	04	0C	08	09	08
18C—	0C	10	0A	08	07	08	10	10	0C	08	09	08	11	10	0E	08
18D—	0E	08	11	08	10	04	0E	04	0C	08	09	08	0C	10	0A	08
18E—	07	08	10	08	13	08	11	10	20	10	40					

Das Lied wird so lange wiederholt, bis die Taste —RS— gedrückt wird.
(40H = Wiederholung)

Aufgaben und Übungen

1. Laden Sie das Programm in den MPF—1 und speichern Sie es auf Cassette.
2. Spielen Sie die vorgeschlagenen Lieder ab.
3. Versuchen Sie selbst ein Lied umzusetzen und vom MPF—1 abspielen zu lassen.

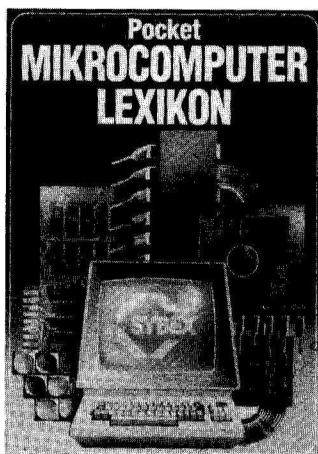
EPROM-Programmier-Service
Herrschinger Meßtechnik GmbH
Postfach 11 55

8036 Herrsching

Tel. (081 52) 25 21.

Mit Vorgabe von Adressen und zugehörigen Daten brennen wir auf Wunsch EPROM's
der Typen TMS 2508, -2516, -2532 / I-2758, -2716, -2732 ein.

Fragen Sie nach unserem Angebot.

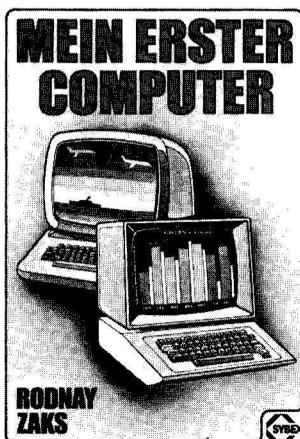


**Pocket
MIKROCOMPUTER LEXIKON**

ca. 150 Seiten
MSB 407E

DM 9,80

Jeder hat etwas zum Thema Computer zu sagen. Seien Sie sicher, daß Sie auch alles verstehen! Dieses Lexikon in Taschenformat enthält über 1300 Definitionen, Zahlen und Kurzformeln griffbereit. Ein Glossar in englischer Sprache, technische Daten, Standards und Lieferantenadressen machen dieses Buch zu Ihrer Informations-Börse.



Rodnay Zaks

MEIN ERSTER COMPUTER

305 Seiten
MSB 231D
(1981)

150 Abbildungen

DM 28,-

Die Einführung für jeden, der den Kauf oder den Gebrauch eines Kleincomputers erwägt. Das Buch setzt weder technisches Spezialwissen noch eine EDV-Erfahrung voraus. Alle Konzepte und Begriffe werden vor ihrer Anwendung erklärt. Das Wie und Warum des persönlichen und geschäftlichen Gebrauchs von Kleinstcomputern wird allgemeinverständlich dargestellt.



Rodnay Zaks
PROGRAMMIERUNG DES Z80
608 Seiten 200 Abbildungen
MSB 353D DM 48,-
(1982)

Dieses Buch beschreibt alle notwendigen Aspekte des Mikroprozessors Z80 samt Vor- und Nachteilen. Es ist angelegt als eine schrittweise Einführung, mit Übungen und Fragen, um das Erlernete zu vertiefen. Es beinhaltet eine vollkommene Aufzeichnung des Befehlssatzes und eine umfassende Beschreibung der internen Funktionen. Der Leser lernt das Programmieren auf einer praktischen Ebene.

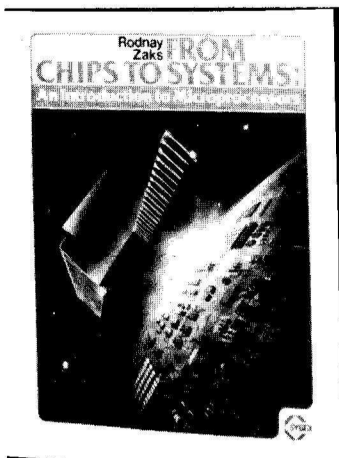


Rodnay Zaks / Austin Lesea
**MIKROPROZESSOR INTERFACE
TECHNIKEN**
440 Seiten 400 Abbildungen
MSB 200D DM 48,-
Übersetzung: Bernd Pol
(1980)

Dieses Buch zeigt systematisch alle nötigen Techniken, Bauteile und Schaltkreise, die für die Schnittstellenentwicklung in der Erstellung eines vollständigen Systems wichtig sind. Die beschriebenen Techniken sind anwendbar auf alle Mikroprozessoren. Alle Hardware- und Softwareaspekte werden dargestellt. Durchschnittliches technisches Wissen und Computererfahrung werden vorausgesetzt.

Weitere Fachbücher erhältlich vom

MSB-Verlag



Rodnay Zaks

FROM CHIPS TO SYSTEMS:

An Introduction to Microprocessing

552 Seiten

325 Abbildungen

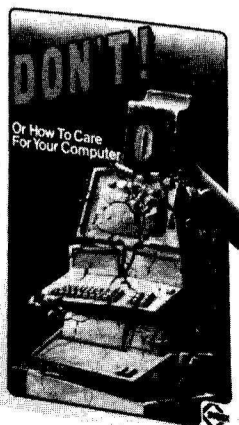
MSB 232E

DM 59,-

(1981)

(in englischer Sprache)

Eine klar gegliederte und sehr gut lesbare Einführung in die faszinierende Welt der Computer. Dieses Buch beschreibt, wie ein Computersystem funktioniert und wie das Herz eines jeden Systems, die Mikroprozessoren, entwickelt, hergestellt und eingesetzt werden. Alle Software- und Hardwareaspekte werden berücksichtigt, so auch die Interface-Techniken, die für die Verbindung der verschiedenen Teile eines Systems nötig sind. Das Buch setzt keine Vorkenntnisse voraus.



Rodnay Zaks

DON'T! (Or How to Care for Your Computer)

213 Seiten

96 Abbildungen

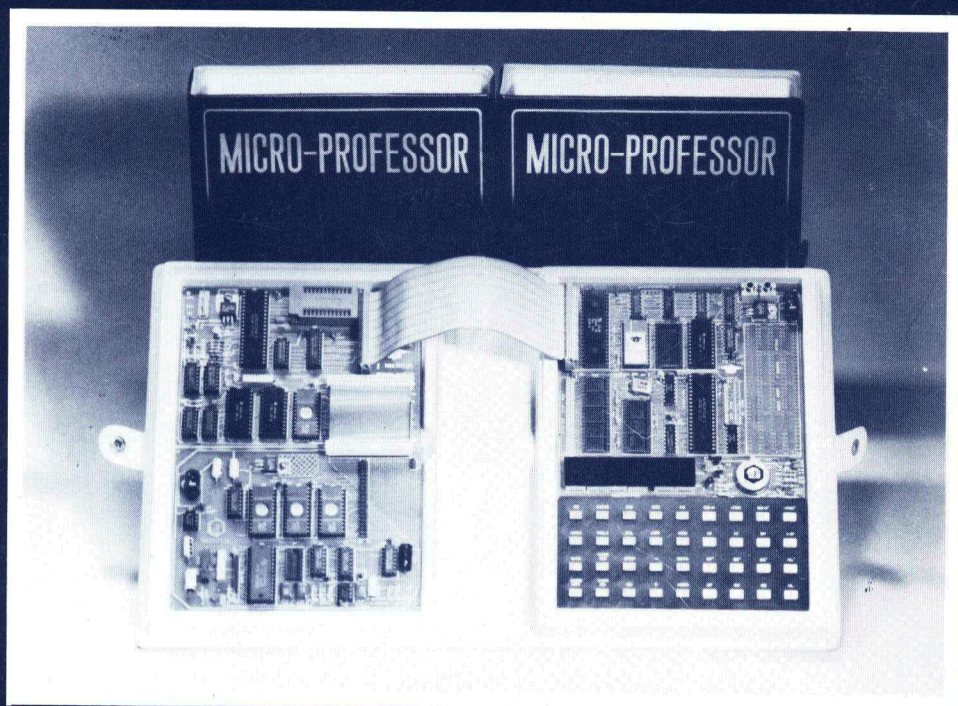
MSB 400E

DM 44,-

(1981)

(in englischer Sprache)

Das Buch, das Ihnen die Handhabung eines Computersystems erklärt – vor allem, was Sie damit nicht machen sollten. Die vorgestellten allgemeingültigen Regeln für die pflegliche Behandlung und vorsichtige Nutzung Ihres Computers, werden Ihnen mindestens einen Besuch eines Technikers ersparen.



MICRO PROFESSOR kann erweitert werden mit:
BASIC,
SPRACHAUSGABE
für englische Worte und Zahlen,
EPROM-Programmiereinheit