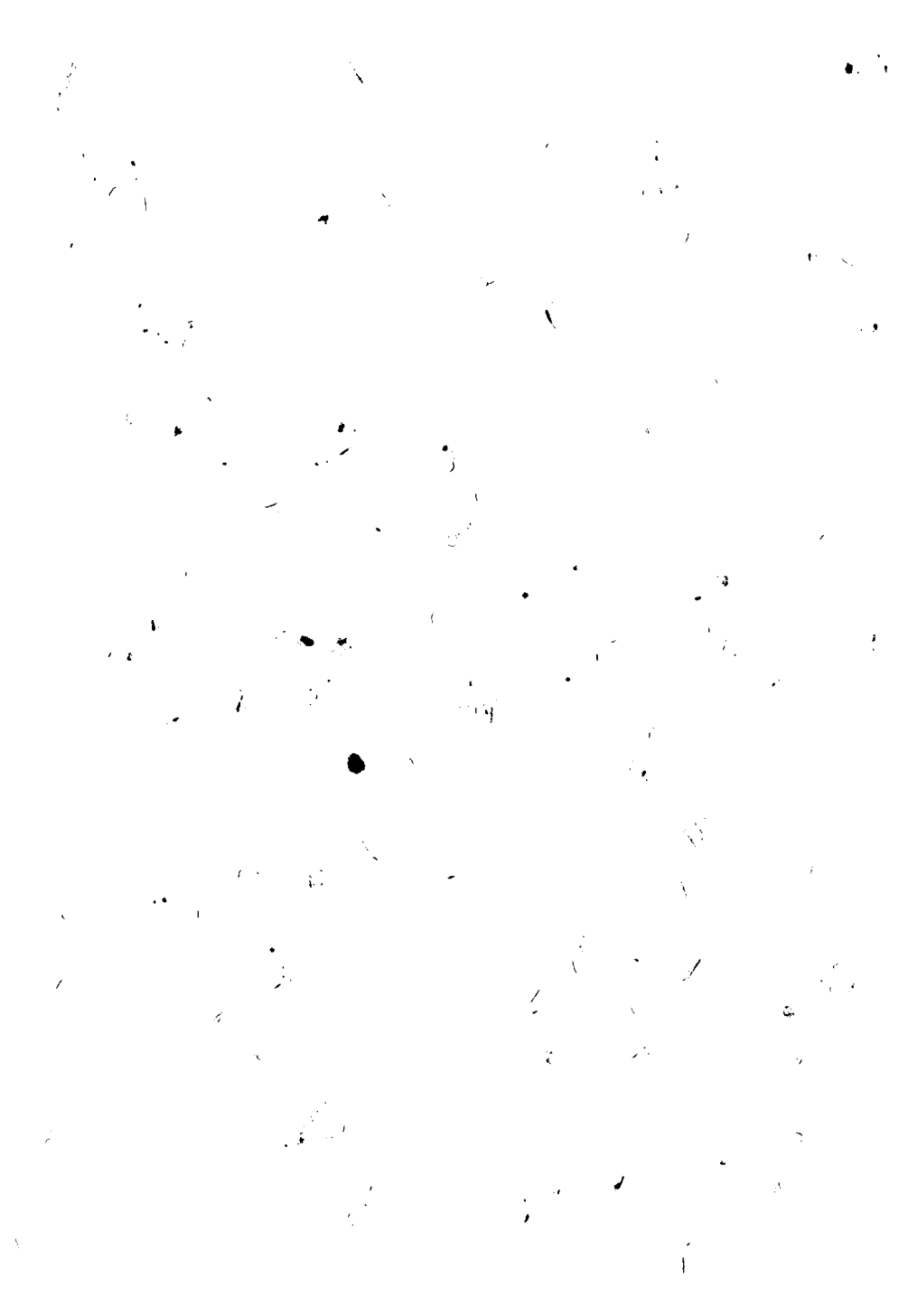


# Handbuch LC 80



# Handbuch Lerncomputer LC 80

1983

**vob mikroelektronik › karl marx › erfurt  
stammbetrieb**



Änderungen, insbesondere solche, die durch den technischen Fortschritt bedingt sind, vorbehalten.

## Inhaltsübersicht

Seite

0.	Einleitung	6
1.	Grundbegriffe der Informationsdarstellung in Mikrocomputern	8
1.1.	Binärelemente	8
1.2.	Bit, Byte und Wort	9
1.3.	Digitale Codes	10
1.4.	Zahlensysteme	11
1.4.1.	Das reine Binärsystem	11
1.4.2.	Das Hexadezimalsystem	15
1.5.	Darstellungsarten von Zahlen im Byte	17
1.5.1.	Darstellungsart positive Binärzahl	17
1.5.2.	Darstellungsart Zweierkomplementzahl	17
1.5.3.	Darstellungsart gepacktes BCD-Format	18
2.	Beschreibung des Mikroprozessorsystems U 880	20
2.1.	Übersicht über das Mikroprozessorsystem U 880	20
2.2.	Aufbau und Arbeitsweise des Mikroprozessors U 880	25
2.2.1.	Registerstruktur	29
2.2.2.	Interruptsystem	32
2.3.	Aufbau und Arbeitsweise des PIO U 855	39
2.3.1.	Schaltkreisbeschreibung	39
2.3.2.	Erläuterung der einzelnen Betriebsarten	46
2.3.2.1.	Betriebsart Byte-Ausgabe (Mode 0)	46
2.3.2.2.	Betriebsart Byte-Eingabe (Mode 1)	47
2.3.2.3.	Betriebsart Byte-Ein-/Ausgabe (bidirektional) (Mode 2)	48
2.3.2.4.	Bit-Ein-/Ausgabe (Mode 3)	49
2.3.3.	Interrupt-Bearbeitung	50
2.3.4.	Programmierung des PIO's	51
2.3.4.1.	Wahl der Betriebsart	51
2.3.4.2.	Ein- bzw. Ausgabedefinition bei Betriebsart Bit-Ein-/Ausgabe	51

2.3.4.3.	Laden des Interrupt-Vektors	52
2.3.4.4.	Interrupt-Steuerung	52
2.4.	Aufbau und Arbeitsweise des CTC U 857	54
2.4.1.	Schaltkreisbeschreibung	54
2.4.2.	Arbeitsweise des Schaltkreises	60
2.4.2.1.	Schreibzyklus	60
2.4.2.2.	Lesezyklus	61
2.4.2.3.	Interrupt-Quittungs-Zyklus	62
2.4.2.4.	Prioritätskaskadierung	64
2.4.2.5.	Zähl- und Zeitgeber-Vorgang	66
2.4.3.	Programmierung des CTC	66
2.4.3.1.	Laden des Interrupt-Vektors	67
2.4.3.2.	Format des Kanalsteuerwortes	68
2.4.3.3.	Format des Zeitkonstantensteuerwortes	69
3.	Befehlsbeschreibung des U 880	70
3.1.	Befehlsstruktur	70
3.2.	Syntax der Assemblersprache	72
4.3.	Adressierungsarten	73
3.4.	Flag-Bit-Technik	75
3.5.	Befehlssatz	78
3.5.1.	Ladebefehle	79
3.5.1.1.	8-Bit-Ladebefehle	79
3.5.1.2.	16-Bit-Ladebefehle	83
3.5.2.	Registertauschbefehle	87
3.5.3.	Blocktransfer- und Suchbefehle	90
3.5.4.	Arithmetikbefehle	95
3.5.4.1.	8-Bit-Arithmetikbefehle	95
3.5.4.2.	16-Bit-Arithmetikbefehle	97
3.5.5.	Sprungbefehle	98
3.5.6.	1-Byte-Logik-Befehle	100
3.5.7.	Rotations- und Schiebebefehle	102
3.5.8.	Bitmanipulationsbefehle	108
3.5.9.	Spezielle Akkumulator- und Flagbefehle	109
3.5.10.	Unterprogrammaufruf- und Rücksprungbefehle	110
3.5.11.	Allgemeine Steuerbefehle	113
3.5.12.	Ein- und Ausgabebefehle	114

4.	Programmierung der Peripherieschaltkreise des LC 80	117
4.1.	Programmäßige Organisation einer Inter- rupt-Serviceroutine (ISR)	117
4.2.	PIO-Programmierung	120
4.3.	CTC-Programmierung	124
5.	Befehlsliste U 880	127
Anhang:	Sachworterläuterungen	147
Anlage:	Bestückungsplan LC 80	

## O. Einleitung

Mit dem Einzug der Mikroelektronik in alle Zweige der Volkswirtschaft entwickelt sich bei vielen Menschen der Wunsch, näheres über dieses interessante technische Gebiet zu erfahren und sich auch entsprechend selbst zu betätigen.

Erzeugnisse, die früher ohne Elektronik auskamen, beinhalten heute die Mikroelektronik als deren miniaturisierte Form, z. B.

- Fotoapparate mit automatischer Einstellfunktion
- Werkzeugmaschinen mit mikroelektronischer Steuerung der Funktionen
- Landwirtschaftsmaschinen mit mikroelektronischer Steuerung der Mähhöhe, Pflugtiefe usw.
- Haushaltgeräte mit Programmablaufspeicherung.

Dazu kommen völlig neue Geräte und Einrichtungen, die nur mit Mikroelektronik möglich sind, z. B.

- Rechner verschiedenster Art
- Fahrkartenautomaten
- mikroelektronische Spiele.

Aus den genannten Beispielen wird deutlich, daß sich auch solche Berufsgruppen mit der Mikroelektronik beschäftigen, deren Haupttätigkeitsfeld der Maschinenbau, die Landwirtschaft, die Optik usw. ist.

Der Lerncomputer LC 80 ist dafür gedacht, der Bevölkerung im allgemeinen und Schülern, Studenten und Berufstätigen in der Aus- und Weiterbildung das modernste Gebiet der Mikroelektronik, die Mikroprozessortechnik, näher zu bringen.

Der Ursprung der Mikroprozessortechnik liegt im Bestreben der Menschen, so effektiv und preiswert wie möglich elektronische Bauelemente zu produzieren. Die Stückzahlen solcher Bauelemente (Schaltkreise) sind ein entscheidendes Kriterium.

Da die verschiedenen Anwender jeweils verschiedene Schaltkreise bei unökonomischen Stückzahlen, jedoch recht billig, verlangten, wurden die Techniker zu akzeptablen Lösungen angehalten.

Sie schufen ein System von Schaltkreisen, die elektrisch zu einer



Schaltung verbunden werden müssen. Dies ist ein Mikroprozessorsystem, dessen Kernstück ("Gehirn") ein Mikroprozessor (CPU) ist. Solche Zusammenschaltung eines Systems von Schaltkreisen nennt man "Hardware", was gleichbedeutend ist mit "körperlich vorhanden". Die Hardware allein löst keine der in den Beispielen genannten Anwendungsfälle. Man muß dem Mikroprozessorsystem erst "eingeben", welche Aufgabe es erledigen soll. Das macht man mit der "Software", einem Programm.

Jedes der genannten Beispiele besteht also aus einer konkreten Hardware und Software, Während die Hardware in ihrem Kern sehr ähnlich ist (immer ist ein Mikroprozessor, z. B. U 880, eingesetzt), hat natürlich ein Fahrkartenautomat eine andere Software als eine Werkzeugmaschine.

Um auf den Ausgangspunkt zurückzukommen: Bei großen Produktionsstückzahlen der Bauelemente können umfangreiche Anwendungsgebiete mit den gleichen Bauelementen befriedigt werden.

Bei Mikroprozessorsystemen erwirbt der Anwender also verschiedene Schaltkreise, schaltet sie zusammen und "sagt" diesem Gebilde mit einer Software, welche Funktion es zu erfüllen hat.

Deshalb wird gelegentlich vom Mikroprozessor als "Alleskönner" gesprochen. Was er aber kann übernimmt er jedoch in einer geeigneten Sprache vom Menschen. Hierfür wurde der Begriff Maschinensprache geprägt, um den qualitativen Unterschied zur menschlichen Sprache zu verdeutlichen.

Der Lerncomputer macht nicht nur mit dem Mikroprozessorsystem als Hardware bekannt. Die Sprache des Mikroprozessors und seine Arbeitsweise wird in diesem Handbuch beschrieben und man kann sogleich am Lerncomputer sein erworbenes Wissen überprüfen. So wird es möglich, verschiedene Programme auszuprobieren und im fortgeschrittenen Stadium dem "Alleskönner" Funktionen zu übertragen. Selbstverständlich handelt es sich beim Lerncomputer LC 80 um einen relativ kleinen Umfang der "Hardware".

Wer sich intensiv mit der Mikroprozessortechnik beschäftigen will, wird die im Handbuch genannten Erweiterungsmöglichkeiten ausnutzen oder sogar auf einer extra Leiterplatte eine spezielle Variante eines Mikroprozessorsystems aufbauen.

Mit der Maschinensprache wird der Mikroprozessor "direkt" angesprochen. Um diese zu erlernen, muß man sich Kenntnisse über die Eigenschaften der Bauelemente aneignen und kann sie dann maximal ausnutzen. Das Erlernen der Maschinensprache qualifiziert zum Programmierer, sowohl den Schüler als auch den Werk tätigen unterschiedlichen Alters und Berufes. Diese werden nur im Grad der Beherrschung der "Kniffe" und speziellen Fähigkeiten der Mikroprozessoren unterschieden.

Für solche Anwender, die mit Hilfe der Mikroelektronik "nur" bestimmte Probleme der Mathematik, Biologie, Statistik usw. lösen wollen, wurden "problemorientierte" Sprachen geschaffen, die in einem Befehl mehrere Maschinenbefehle realisieren, wie "Basic", "Pascal" usw.

Die Übersetzung in die Maschinensprache übernimmt in diesem Fall das Mikroprozessorsystem selbst. Auch diese Aufgabe wurde ihm vom Menschen mit einer entsprechenden "Software" vorgegeben. Moderne Heimcomputer sind vorwiegend für Basic vorgesehen.

Das Charakteristische der Maschinensprache ist, daß sowohl alle Informationen als auch alle Befehle in Zahlenform codiert werden müssen. Wie jede andere Sprache erfordert dies ein fleißiges Lernen der Vokabeln. Dem gegenüber steht die einfache "Grammatik" und "Syntax".

Wer also die erste Hürde genommen hat, wird mit Schwung, nach und nach die weiteren nehmen, um dann zum immer größer werdenden Kreis der "Mikroprozessor-Fan's" zu gehören.

## 1. Grundbegriffe der Informationsdarstellung in Mikrocomputern

Um die Arbeitsweise eines Mikrocomputers und seine Programmierung zu verstehen, sind einige Grundkenntnisse notwendig. Diese werden im folgenden kurz erläutert.

### 1.1. Binärelemente

Informationen, wie Zahlen, Buchstaben, Symbole, Operationen mis-

sen so dargestellt werden, daß sie von einem Computer verstanden und verarbeitet werden können. Das ist möglich, wenn Elemente verwendet werden, die zwei gleichberechtigte Zustände annehmen können. Diese Elemente nennt man Binärelemente.

Beispiele dafür sind:

- Schalter, die offen oder geschlossen sein können
- Löcher oder keine Löcher an einer bestimmten Lochkartenstelle
- zwei mögliche Magnetisierungsrichtungen an einer Stelle auf einem Magnetband
- zwei verschiedene Spannungs- oder Strompegel
- abstrakte Symbole 0 und 1

Für die Darstellung der Informationen im Computer werden die Binärelemente 0 und 1 verwendet.

## 1.2. Bit, Byte und Wort

Das Bit ist die kleinste Darstellungseinheit für Informationen. Es ist eine Abkürzung von "binary digit", das mit Binärziffer übersetzt werden kann.

Man kann sich das Bit z. B. als eine Lampe vorstellen, die beliebig ein- oder ausgeschaltet werden kann. Das Zeichen "1" definiert die Lampe im eingeschalteten Zustand. Das Zeichen "0" steht für die ausgeschaltete Lampe.

Ein Bit ist also mit einem binären Zustand 1 bzw. 0 identisch. Informationen bestehen meistens aus mehreren Bits, die in einer Bitfolge zusammengefaßt sind. So stellt z. B. die Bitfolge 1000 die Dezimalziffer 8 dar.

Eine Anzahl von  $n$  aufeinanderfolgenden Bits, die ein Computer gleichzeitig verarbeiten kann, bilden ein Wort mit einer Länge von  $n$ -Bit.

Als Byte bezeichnet man ein Wort mit einer Länge von 8 Bit. Ein Byte besetzt in dem gebräuchlichen Mikrocomputersystem genau einen Speicherplatz. Es gibt bereits Computer, bei denen der Speicherplatz 16 oder sogar 32 Bit aufnehmen kann.

Um die Bezeichnung der einzelnen Bits innerhalb eines Bytes zu erleichtern, werden die Bits von 0 bis 7 nummeriert:

B7 B6 B5 B4 B3 B2 B1 B0

Das höherwertigste Bit ist B7. Das niederwertigste Bit ist B0. Um eine Speicherzelle im Computer auffinden zu können, erhält diese eine "Hausnummer", die Speicheradresse. Diese Speicheradresse besteht aus einem Wort mit einer Länge von 16 Bit, das als Adreß-Wort bezeichnet wird.

### 1.3. Digitale Codes

Der digitale Code ist eine einfache Sprache, die ein Computer oder eine digitale Schaltung verstehen und verarbeiten kann. Er besteht aus einem System von Symbolen. Der digitale Code ermöglicht das Speichern, Verarbeiten und Weitergeben von Daten und Informationen.

So wie es verschiedene Sprachen gibt, unterscheidet man auch verschiedene Codeklassen:

- Codes, mit denen in digitalen Schaltungen bestimmte Operationen, z. B. Zählen, Addieren ausgeführt werden können.

Beispiel: Binär-code (auch binäre Zeichendarstellung)

- Codes, mit denen die Dezimalziffern 0 ... 9 einzeln für die übersichtliche Darstellung im Computer binär verschlüsselt werden:

Beispiel: BCD-Code (Binary Codet Dezimal = Binär-code für Dezimalziffern)

- Codes, mit denen nicht nur Dezimalziffern, sondern auch alle Buchstaben des Alphabets, Operationen, Steuerzeichen usw. verschlüsselt werden können.

Beispiel: ASCII-Code (American Standard Code for Information Interchange = 7-Bit-Code für die Informationsverarbeitung). Zur Darstellung innerhalb des Computers wird das 8. Bit ergänzt und mit 0 belegt.

- Codes, die bewirken, daß der Computer eine vorgeschriebene Operationsfolge ausführt.

Beispiel: U 880-Befehlscode

Die genannten Code-Beispiele:

- Binärcode
- BCD-Code
- ASCII-Code
- U 880-Befehlscode

werden bei der Programmierung des U 880-Mikroprozessors verwendet.

Der ASCII-Code ist nicht für die interne Verarbeitung im Computer gedacht, sondern für den Datenaustausch mit peripheren (anschließbaren) Geräten, z. B. Lochbandleser (hier erscheint das 8. Bit als Paritätsbit).

Die anderen genannten Codes werden in den Abschnitten 1.4. und 1.5. genauer erläutert.

#### 1.4. Zahlensysteme

##### 1.4.1. Das reine Binärsystem

Wir sind gewöhnt, im Dezimalsystem zu denken.

Die einzelnen Ziffern einer Zahl haben hier Werte, die von ihren Positionen abhängen.

$$\begin{aligned} \text{Z. B.} & \quad \quad \quad 1 \quad \quad 9 \quad \quad 8 \quad \quad 4 \\ & \quad \quad \quad \underbrace{\quad\quad\quad} \quad \underbrace{\quad\quad\quad} \quad \underbrace{\quad\quad\quad} \quad \underbrace{\quad\quad\quad} \\ = & \quad \quad \quad 1 \cdot 10^3 + 9 \cdot 10^2 + 8 \cdot 10^1 + 4 \cdot 10^0 \\ = & \quad \quad \quad 1 \cdot 1000 + 9 \cdot 100 + 8 \cdot 10 + 4 \cdot 1 \\ = & \quad \quad \quad 1000 + 900 + 80 + 4 \end{aligned}$$

Das Dezimalsystem umfaßt die Ziffern 0 ... 9.

Die Basis eines Zahlensystems entspricht der Anzahl der verwendeten Ziffern. Die Basis des Dezimalsystems ist also 10. Da zur Darstellung im Computer nur die Binärelemente 0 und 1 benutzt werden können, muß das Dezimalsystem durch ein anderes Zahlensystem ersetzt werden.

Das reine Binärsystem (auch Dualsystem) entspricht der genannten Forderung. Es umfaßt die Binärelemente 0 und 1. Die Basis ist 2. Die einzelnen Ziffern einer Zahl haben ebenfalls positionsabhängige Wertigkeiten.

$$\begin{aligned}
 \text{Z. B.:} & \quad 1 \quad 1 \quad 0 \quad 1 \\
 = & \quad \overbrace{1 \cdot 2^3} + \overbrace{1 \cdot 2^2} + \overbrace{0 \cdot 2^1} + \overbrace{1 \cdot 2^0} \\
 = & \quad 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\
 = & \quad 13
 \end{aligned}$$

Die Binärzahl 1101 entspricht der Dezimalzahl 13.

Da ein und dieselbe Zahl in verschiedenen Zahlensystemen durch unterschiedliche Ziffernfolgen dargestellt wird, muß ein Hinweis auf die Basis des Zahlensystems gegeben werden. Das geschieht durch Anfügen der Basis als Index oder durch Anfügen der Buchstaben

D für das Dezimalsystem

B für das Binärsystem

Unser Umrechnungsbeispiel könnte also auch kürzer geschrieben werden:

$$1101_2 = 13_{10}$$

oder  $1101_B = 13_D$

Soll die Umrechnung in umgekehrter Richtung erfolgen, also die Dezimalzahl in eine Dualzahl umgeformt werden, gibt es verschiedene Verfahren. Das gebräuchlichste ist für ganze Zahlen die wiederholte Division durch 2 (allgemeingültig wird durch die Basis des Zahlensystems, in das umgerechnet wird, dividiert).

13 : 2 = 6	Rest 1	
6 : 2 = 3	" 0	
3 : 2 = 1	" 1	
1 : 2 = 0	" 1	

13<sub>10</sub> = 1 1 0 1<sub>2</sub>

Tabelle 1.1: Dezimal- und Binärzahlen 0 ... 16

Dezimalzahl	Binärzahl
	Wertigkeit 16 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1
16	1 0 0 0 0

Wie aus der Tabelle ersichtlich ist, können in 4 Bit die Dezimalzahlen 0 ... 15 dargestellt werden. Für größere Dezimalzahlen werden in der Binärdarstellung mehr als 4 Bit benötigt.

**Tabelle 1.2.: Überblick über die Zuordnung Dezimalzahl-Binärzahl  
im Bereich von 0 bis 65535**

<u>Dezimalzahl</u>	<u>Binärzahl</u>	
0	0	
1	1	
2	1 0	
3	1 1	
4	1 0 0	
7	1 1 1	
8	1 0 0 0	
15	1 1 1 1	4 Bit
16	1 0 0 0 0	
31	1 1 1 1 1	
32	1 0 0 0 0 0	
63	1 1 1 1 1 1	
64	1 0 0 0 0 0 0	
127	1 1 1 1 1 1 1	
128	1 0 0 0 0 0 0 0	
255	1 1 1 1 1 1 1 1	8 Bit
256	1 0 0 0 0 0 0 0 0	
511	1 1 1 1 1 1 1 1 1	
512	1 0 0 0 0 0 0 0 0 0	
1023	1 1 1 1 1 1 1 1 1 1	
1024	1 0 0 0 0 0 0 0 0 0 0	
2047	1 1 1 1 1 1 1 1 1 1 1	
2048	1 0 0 0 0 0 0 0 0 0 0 0	
4095	1 1 1 1 1 1 1 1 1 1 1 1	12 Bit
4096	1 0 0 0 0 0 0 0 0 0 0 0 0	
8191	1 1 1 1 1 1 1 1 1 1 1 1 1	
8192	1 0 0 0 0 0 0 0 0 0 0 0 0 0	
16383	1 1 1 1 1 1 1 1 1 1 1 1 1 1	
16384	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
32767	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
32768	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
65535	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	16 Bit



Zur Abkürzung wurde definiert:

$$1 \text{ K} = 2^{10} = 1024$$

Mit einem Adreßwort von 16 Bit Länge lassen sich, wie aus der Tabelle leicht ersichtlich wird, genau 65536 (0 ... 65535) Speicherplätze, man könnte auch sagen 64 K Speicherplätze adressieren. Ein Speicherplatz hat eine Länge von 8 Bit = 1 Byte. Deshalb spricht man auch von 64 KByte adressierbarem Speicherbereich.

#### 1.4.2. Das Hexadezimalsystem

Will man sich eine achtstellige Bitreihe (1 Byte) innerhalb kürzester Zeit merken, ist das sehr schwierig. Deshalb hat man nach einer Vereinfachung gesucht. Man benutzt in der Mikrocomputertechnik speziell das Hexadezimalsystem, das eine wesentlich größere Übersicht als das reine Binärsystem bietet. Es wird auch als Sedezimalsystem bezeichnet. Das Hexadezimalsystem hat die Basis  $16 = 2^4$ . Es verwendet die Ziffern 0 ... 9, und da die Elemente nur 1 Zeichen lang sein sollen, ergänzt man die fehlenden 6 Zeichen durch die Buchstaben A ... F. Eine Hexadezimalziffer, oder kurz Hex-Ziffer, entspricht dem Wert einer 4-Bit-Binärzahl.

Tabelle 1.3.: 4- und 8-Bit-Binärzahlen und Hexadezimalzahlen

<u>Dezimalzahl</u>	<u>Binärzahl</u>	<u>Hexadezimalzahl</u>
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F
16	0 0 0 1 0 0 0 0	10
31	0 0 0 1 1 1 1 1	1F
32	0 0 1 0 0 0 0 0	20
255	1 1 1 1 1 1 1 1	FF

Die Umwandlung einer Binärzahl in eine Hexadezimalzahl erfolgt so, daß man die Binärzahl fortlaufend von rechts nach links in Vierergruppen aufteilt. Dann wird jede Vierergruppe entsprechend der Zuordnungsvorschrift aus Tabelle 1.3. in eine Hexadezimalziffer umgerechnet.

Zur Kennzeichnung einer Hexadezimalzahl verwendet man ein nachgestelltes H oder die Fußnote 16.

z. B.  $1\ 0\ 0\ 1\ 1\ 1\ 0\ 1 = 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1 = 9D_{16}$  (oder 9DH)  
 $0\ 0\ 1\ 1\ 1\ 1\ 1\ 1 = 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1 = 3F_{16}$  (oder 3FH)

## 1.5. Darstellungsarten von Zahlen im Byte

Ein Byte kann bei gleichem Inhalt völlig unterschiedliche Informationen enthalten.

### 1.5.1. Darstellungsart positive Binärzahl

Wenn man den Inhalt eines Bytes als reine Binärzahl betrachtet, so können Zahlen zwischen 0 und 255 dargestellt werden. Reine Binärzahlen werden als positive Zahlen gewertet. Soll ein Vorzeichen dazugehören, muß es extra abgespeichert werden. Die Darstellungsart positive Binärzahl wird benutzt für:

- Darstellung von Zahlen und Zählereignissen (Binärcode)
- Codierung von Ziffern, Buchstaben, Operations- und Steuerzeichen (ASCII-Code)
- Befehlscode des Mikroprozessors.

### 1.5.2. Darstellungsart Zweierkomplementzahl

Das Zweierkomplement wird benutzt, um in einem Byte eine negative Zahl darzustellen. Bei dieser Darstellungsart steht im höchstwertigsten Bit (B7) eines Bytes das Vorzeichen. Positive Zahlen werden mit einer 0, negative mit einer 1 gekennzeichnet. Dadurch wird der Zahlenbereich der reinen Binärzahl, der von 0 bis 255 reicht, in 2 Teilbereiche, die unsymmetrisch zu 0 sind, aufgespalten. Die beiden Teilbereiche umfassen dann -128 bis -1 und 0 bis +127. Das Zweierkomplement  $\bar{Z}$  einer 8stelligen Binärzahl ist definiert als Differenz von  $2^8$  und  $Z$

$$\bar{Z} = 2^8 - Z$$

Ist  $Z$  positiv, so ist  $\bar{Z}$  die Darstellung von  $-Z$ .

Ist  $Z$  negativ, so ist  $\bar{Z}$  die Darstellung von  $+Z$ .

Das Zweierkomplement einer Binärzahl  $Z$  wird gebildet, indem  $Z$  bitweise negiert und anschließend eine 1 addiert wird.

Beispiel:  $00011100 = 28_{10}$

bitweise

Negation

$$\begin{array}{r} 11100011 \\ + \quad \quad \quad 1 \\ \hline 11100100 = -28_{10} \end{array}$$

Beispiel:  $10011111 = -97_{10}$

bitweise

Negation

$$\begin{array}{r} 01100000 \\ + \quad \quad \quad 1 \\ \hline 01100001 = 97_{10} \end{array}$$

Als einfache Methode kann man sich merken:

nach der niederwertigsten 1 nach links werden alle Bits negiert.

Beispiel:

$$\begin{array}{r} \text{niederwertigste 1} \\ 00011100 = 28_{10} \\ \hline 11100100 = -28_{10} \end{array}$$

### 1.5.3. Darstellungsart gepacktes BCD-Format

Um Zahlen gut lesbar darzustellen, wurde ein spezielles Format geschaffen, das man gepacktes BCD-Format (Binärcode für Dezimalziffern) nennt.

Dazu werden die 8 Bit eines Bytes in 2 Gruppen zu jeweils 4 Bit aufgeteilt, wie beim Einteilungsprinzip der Hexadezimalschreibweise. Damit könnten mit jedem Halbbyte Werte von 0 ... 15 erfasst werden. Wird der Wertebereich auf 0 ... 9 eingeschränkt, so spricht man vom gepackten BCD-Format, weil jeweils 4 Bit eine Dezimalziffer zwischen 0 und 9 darstellen. In einem Byte können damit Dezimalzahlen zwischen 0 und 99 erfasst werden. Für größere Zahlen müssen mehrere Bytes verwendet werden.

z. B.:

		0 0 0 0		1 0 0 0	=	8 <sub>10</sub>
		0 1 1 0		0 1 1 1	=	67 <sub>10</sub>
0 0 0 1	1 0 0 1	1 0 0 0		0 1 0 0	=	1984 <sub>10</sub>

Für sehr rechnerintensive Aufgaben ist es unzweckmäßig, mit diesem Format zu arbeiten, da zusätzliche Dezimalkorrekturen durchgeführt werden müssen.

## 2. Beschreibung des Mikroprozessorsystems U 880

### 2.1. Übersicht über das Mikroprozessorsystem U 880

Das Mikroprozessorsystem U 880 besteht im wesentlichen aus folgenden Grundbausteinen:

U 880	- <u>C</u> entral <u>P</u> rozessor <u>U</u> nit (zentrale Verarbeitungseinheit)	(CPU)	} Peripherie- bausteine
U 855	- <u>P</u> arallel <u>I</u> nput <u>O</u> utput (parallele Ein-/Ausgabe)	(PIO)	
U 856	- <u>S</u> erial <u>I</u> nput <u>O</u> utput (serielle Ein-/Ausgabe)	(SIO)	
U 857	- <u>C</u> ounter <u>T</u> imer <u>C</u> ircuit (Zähler/Zeitgeber)	(CTC)	

Jeder Mikrorechner besteht aus mindestens einer CPU und Speicherschaltkreisen zur Programm- und Testwertspeicherung (ROM, EPROM). Sinnvollerweise wird dieser Aufbau ergänzt, um verschiedene Ein- und Ausgaben realisieren zu können. Dazu werden ein oder mehrere PIO, SIO und CTC benutzt, die auch Peripherie-Schaltkreise genannt werden. Zusätzliche Speicherschaltkreise dienen als Arbeitsspeicher (RAM).

Der Aufbau der Schaltkreise beruht auf einem einheitlichen Konzept (Systemkonzept des U 880), das folgende Merkmale aufweist:

- Austausch von Adressen zwischen der CPU und den Peripherieschaltkreisen und dem Speicher über den Adressbus.  
Austausch von Daten über den Datenbus, Austausch von Steuerungssignalen über eine Reihe von Steuerleitungen.
- Einheitliches System zur Programmunterbrechung durch die Peripherieschaltkreise (Interrupt) mit der Möglichkeit,

die Peripherieschaltkreise für mehrere gleichzeitige Interrupts und einem bestimmten Vorrangsystem (Prioritätskaskade) zu verketteten.

- Umfangreiche Möglichkeiten zur Programmierung der besonderen Eigenschaften der einzelnen Schaltkreise.
- Gemeinsamer Systemtakt bis zu 4 MHz je nach Ausführungsvariante, durch den die Arbeitsgeschwindigkeit der Grundschrirte bestimmt wird.
- Einheitliche Spannungsversorgung von +5 V, abgestimmte Logikpegel.

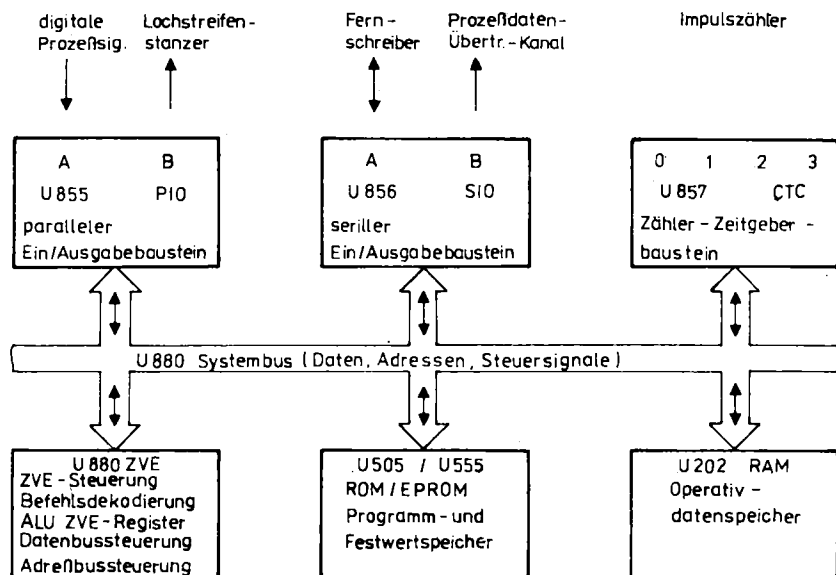


Bild 2.1. Grundstruktur eines mit Bausteinen aus dem System U 880 aufgebauten Mikrorechners

Die U 880-CPU übernimmt als Kernstück des gesamten Mikroprozessorsystems

- die Überwachung des aufgebauten Mikrorechners
- die Befehlsabarbeitung und
- den überwiegenden Teil der Interruptsteuerung.

Es handelt sich beim U 880 um eine CPU, die mit Befehlswörtern variabler Länge (1, 2, 3 oder 4 Byte) standardmäßig Datenwörter von 8 Bit (= 1 Byte) verarbeiten kann, aber auch 1, 4 und 16 Bit.

Neben den Befehlen zur Verarbeitung der Daten stehen eine Fülle von Befehlen zur Programmorganisation zur Verfügung.

Datenquelle oder Datensenke können dabei die CPU, die Peripherieschaltkreise oder die Speicher sein. Der Datentransport (Daten-transfer) innerhalb des Systems erfolgt über die CPU.

(Ausnahme: Datentransfer zur Peripherie mittels direktem Speicherzugriff (DMA-Betrieb))

Eine genaue Beschreibung der CPU und ihrer Arbeitsweise folgt im Abschnitt 2.2.

Die Peripherieschaltkreise werden hier nur kurz vorgestellt und im Abschnitt "Programmierung der Peripheriebausteine" genauer erläutert.

### U 855 - Parallele Ein-/Ausgabe (PIO)

Beim U 855 handelt es sich um einen Schaltkreis, der über zwei 8-Bit-Datenkanäle (Ports) den parallelen Datenaustausch (8 Bit parallel pro Kanal) zwischen Mikroprozessor und Peripherie realisiert.

Parallele Ein-/Ausgabe ist in der Praxis erforderlich zum Anschluß von Lochstreifenlesern, -stanzern, Druckern sowie andere Formen der digitalen Ein- und Ausgabe, z. B. Tastaturen.



Der PIO kann über die Steuerbefehle, die er von der CPU erhält, an die jeweilige Aufgabe (Betriebsart) angepaßt werden. Der U 855 enthält (wie auch der U 856) ein Signal zur Quittung (Handshake) bei Datenübernahme, -übergabe und den restlichen, nicht in der CPU befindlichen Teil des Interruptsystems. Der U 855 wird in der Zusammenschaltung zum Mikrorechner mit den anderen Peripheriebausteinen in eine Prioritätskaskade eingebunden (Daisy chain), die der im jeweiligen Einsatzfall gewünschten Vorrangstruktur der angeschlossenen Peripheriegeräte entsprechen muß.

#### U 856 - Serielle Ein-/Ausgabe (SIO)

Der U 856 dient zum Bit-seriellen Datenaustausch (ein Bit nach dem anderen) zwischen Ein-/Ausgabegeräten und dem Mikroprozessor. Solche Geräte sind u. a. Fernschreiber und Floppy-Disk's, aber auch serielle Prozeßdatenübermittlungskanäle.

Der SIO-Baustein enthält zwei serielle 1-Bit-Kanäle, die unabhängig voneinander sowohl Daten senden als auch empfangen können.

Der Datenaustausch zwischen dem U 856 und dem U 880 erfolgt (wie beim U 855) in Byte-serieller (8-Bit-paralleler) Form. Die Umwandlung Bit-serieller Datenwörter in Byte-serielle Datenwörter und umgekehrt, einschließlich einer doppelten 8-Bit-Empfangspufferung, erfolgt in dem seriellen Ein-/Ausgabe-Baustein (SIO).

Die maximale Datenübertragungsrate, eine wichtige Kenngröße jedes seriellen Datenkanals, liegt bei etwa 500 KBit/s.

Das beim U855 zur Quittungslogik und zum Interruptregime gesagte gilt auch für den U 856.

#### U 857 - Zeitgeber/Zähler (CTC)

Der U 857 kann als Zeitgeber oder als Zähler eingesetzt und programmiert werden.

In der Funktion Zeitgeber ermöglicht er dem Anwender, die über ein Interruptsignal ausgelöste Bearbeitung mehrerer zeitzyklischer Echtzeitaufgaben oder z. B. den Aufbau einer software-gesteuerter Echtzeituhr in der CPU.

Die Funktion Zähler realisiert das Abarbeiten voreingestellter Rückwärtszähler mit maximal 256 externen Einzelimpulsen. Beim Erreichen des Zählerstandes Null erfolgt eine Interruptmeldung.

Der CTC-Baustein ist intern aus vier voneinander unabhängigen 8-Bit-Kanälen aufgebaut, die auch durch entsprechende Schaltungsmaßnahmen miteinander verkettet werden können.

### Speicherbauelemente (ROM, EPROM, RAM)

Für den Aufbau von Mikrorechnersystemen sind neben den Grundbausteinen auch Speicherbauelemente notwendig. Meist werden die generell byteorientierten Programmspeicher, die vor allem als Festwertspeicher (ROM, read only memory) ausgeführt sind, und die Datenspeicher die als RAM-Lese-Schreib-Speicher ausgeführt sind, zu einem einheitlichen Arbeitsspeicher zusammengefaßt, wobei der Adreßraum für ROM- und für RAM-Bereiche genutzt wird.

Die Gruppe der Festwertspeicher ROM hat die Aufgabe, als Nur-Lese-Speicher einmal eingeschriebene Daten oder Befehle zerstörungsfrei für ein beliebig häufiges Lesen bereitzustellen.

Die wichtigsten Arten der Festwertspeicher sind:

#### - ROM (read only memory)

Der eigentliche ROM ist ein im letzten Fertigungsschritt maskiert programmierter Festwertspeicher mit nicht mehr veränderlichem Inhalt (Bitmuster).

Z. B.: U 505

#### - PROM (programmable read only memory)

Vom Anwender mit einem speziellen Programmiergerät programmierbare Festwertspeicher, deren eingeschriebener Inhalt ebenfalls

nicht mehr gelöscht werden kann.

- EPROM (erasable PROM)

Vom Anwender mit einem speziellen Programmiergerät elektrisch programmierbare Festwertspeicher, deren Inhalt mit Hilfe von UV-Licht global gelöscht werden kann.

Z. B.: U 555

- RAM (random access memory)

Speicher mit wahlfreiem Zugriff (lesen oder schreiben), haben die Aufgabe, Daten oder Befehle während des Rechenbetriebes des Mikroprozessors aufzunehmen und wieder bereitzustellen. Mit dem Abschalten der Betriebsspannung verlieren sie ihre Information, wenn nicht spezielle Maßnahmen zur Betriebsspannungspufferung vorgesehen sind. Bezüglich ihrer Systemeigenschaften unterscheiden sich RAM's nicht wesentlich.

Während ROM's fast ausschließlich Byte-organisiert sind, haben RAM's sowohl 1-Bit- als auch 4- und 8-Bit-Verarbeitungsbreite.

Z. B.: U 202, U 214, U 224

Die Speicherbauelemente unterscheiden sich weiterhin nach

- Speicherdichte
- Stromaufnahme
- Geschwindigkeit
- Art des Datenaustausches
- Art der Steuerung
- Art der Datenerhaltung.

International gibt es ein umfangreiches Sortiment von Speicherbauelementen.

## 2.2. Aufbau und Arbeitsweise des Mikroprozessors U 880

Die CPU des Mikroprozessorsystems U 880 ist neben dem leistungsfähigen Befehlsvorrat und der hohen Verarbeitungsgeschwindigkeit durch einen umfangreichen Registersatz (Bild 2.2.) gekennzeichnet.

Register sind schnelle Zwischenspeicher, mit parallelem Zugriff zu den Bits des gespeicherten Wertes.

Die CPU enthält 208 Bits im internen RAM, zu denen der Programmierer Zugriff hat. Bild 2.3. illustriert, wie dieser Speicher in 18 Register zu 8 Bit und 4 Register zu 16 Bit unterteilt ist.

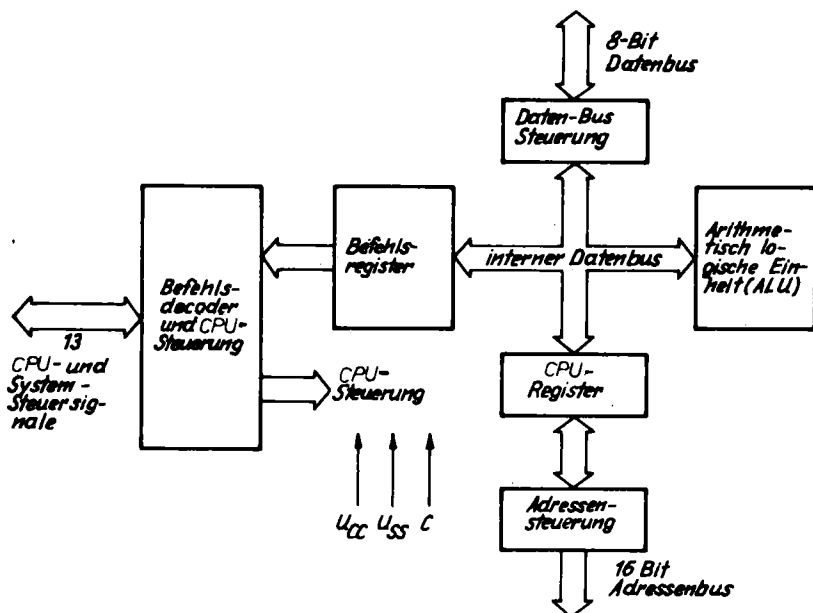


Bild 2.2.: Blockschaltbild der CPU - U 880

Alle U 880-Register sind als statische RAM's ausgeführt. Die Register umfassen zwei Sätze von 6 Allgebrauchsregistern, die individuell als 8-Bit-Register oder in Paaren als 16-Bit-Register verwendet werden können. Ebenfalls sind zwei Akkumulatoren und zwei Flagregister vorhanden (Die Registerstruktur wird in Abschnitt 2.2.1. erläutert.).

## Hauptregistersatz

## Alternativsatz

Akkumu- lator A	Flags F	Akkumu- lator A'	Flags F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

} Allge-  
brauchs-  
regi-  
ster

Interrupt Vektor I	Speicher Refresh R
Index Register	IX
Index Register	IY
Keller-Zeiger	SP
Befehls-Zähler	PC

Spezialregister

Bild 2.3.: Aufbau des CPU-Registers

Neben den allgemeinen Registern existieren zwei Indexregister (IX und IY), ein Keller-Zeiger (Stack-Pointer = SP), ein Befehls-Zähler (PC), ein Interruptvektorenregister (I) und ein Speicher-Refresh-Register (R). Die Kommunikation des Mikroprozessors mit seiner Umwelt läuft über 40 Anschlußbeine (Pins) als 8-Bit-Datenbus, als 16-Bit-Adressenbus und eine Reihe von Steuersignalen (Tabelle: 2.1.).

Tabelle 2.1. Signalbelegung U 880

Signal- bezeichnung	Erläuterung
A <sub>15</sub> ...A <sub>0</sub>	Adressensignale 16 Bit
D <sub>7</sub> ...D <sub>0</sub>	Datensignale 8 Bit
<u>MREQ</u>	MEMORY REQUEST kennzeichnet, daß auf dem Adref bus eine Speicher-Lese- oder eine Speicher-Schreib-Adresse ansteht.
<u>IORQ</u>	IN/OUT REQUEST kennzeichnet, daß auf dem Adref bus eine Ein-/Ausgabe-Portadresse ansteht. IORQ wird außerdem im Interruptakzeptierungszyklus verwendet.
<u>RD</u>	READ Daten lesen
<u>WR</u>	WRITE Daten schreiben
<u>MT</u>	Befehlslesezykluskennzeichen (fetch-cycle)
<u>BUSRQ</u>	BUS REQUEST Mitteilung an den U 880, daß Adref Daten- und Steuersignalleitungen der CPU entzogen werden sollen (z. B. für DMA)
<u>BUSAK</u>	BUS ACKNOWLEDGE Antwort des U 880 auf BUSRQ. Die angeforderten Signalleitungen sind frei.
<u>RFSH</u>	REFRESH Auffrischungssignal für dynamische RAM Speicher
<u>RESET</u>	Rücksetzen der CPU (IFF 1, IFF 2, PC, I, R=0)
<u>HALT</u>	Stop der Befehlsabarbeitung und zyklische Ausführung intern erzeugter NOP-Instruktionen. Der U 880 ist nur durch RESET, NMI oder INT aus dem HALT zu befreien.
<u>WAIT</u>	CPU geht in einen Wartestand
<u>INT</u>	Interruptsignal (durch Programm gesteuert)
<u>NMI</u>	nichtmaskierbares Interruptsignal (nicht durch Programm gesteuert)
C	CLOCK Takteingang
U <sub>CC</sub>	Betriebsspannung +5 V
U <sub>SS</sub> (GND)	Masse

(Begriffserläuterung für Interrupt siehe 2.2.3.)

## 2.2.1. Registerstruktur

### Spezialregister

#### Befehlszähler (PC = Program Counter)

Der Befehlszähler enthält die 16-Bit-Adresse des aktuellen Befehls, der vom Speicher zu holen ist. Der Befehlszähler wird automatisch erhöht, nachdem sein Inhalt in die Adressenleitung überführt worden ist. Wenn Programmsprünge auftreten, wird der neue Wert automatisch in den PC überführt.

Die 16 Bit dieses Registers ermöglichen die direkte Adressierung von 65536 Speicherplätzen (64 K).

#### Keller-Zeiger (SP = Stackpointer)

Der Zeiger enthält die 16-Bit-Adresse des aktuellen obersten Wertes eines Kellers, der irgendwo in einem externen System-RAM untergebracht ist. Die Anordnung und Festlegung seiner Größe wird vom Programmierer übernommen. Dieser externe Kellerspeicher ist als "zuletzt hinein, zuerst heraus" (Last in First Out-LIFO-Datei) organisiert.

Daten können durch die Ausführung der PUSH- oder POP-Befehle, von speziellen CPU-Registern ausgekellert werden oder aus dem Keller in bestimmte CPU-Register zurückgeholt werden.

Die aus dem Keller geholten Daten sind immer die, die als letzte zuvor gekellert wurden. Der Keller ermöglicht eine einfache Gestaltung von Mehrfach-Interrupts, unbegrenzter Unterprogrammtechnik und Vereinfachungen bei vielen Arten von Datenbehandlungen.

#### Zwei Indexregister (IX + IY)

Die zwei unabhängigen Register enthalten eine 16-Bit-Basis-Adresse, die bei indizierter Adressierung (siehe 3.3.) verwendet werden. Dabei wird ein Indexregister als Basis benutzt, um das Gebiet im Speicher festzulegen, von dem aus Daten gespeichert oder ent-

nommen werden sollen. Ein zusätzliches Byte ist in indizierten B fehlen enthalten, um die Entfernung von dieser Basis anzugeben. Diese Entfernung ist als Zweierkomplement der entsprechenden Zahl spezifiziert. Diese Adressierungsart vereinfacht in starkem Maße viele Typen von Programmen, speziell dort, wo Tabellen-Daten benutzt werden.

Außerdem können die Indexregister auch als 16-Bit-Allgebrauchsregister benutzt werden.

### Interrupt-Vektor-Register (I)

Die CPU U 880 kann so betrieben werden, daß ein indirekter Aufruf zu irgendeinem Speicherplatz in Abhängigkeit von einem Interrupt erreicht werden kann. Das Register I enthält dann die höchsten 8 Bits der indirekten Adresse, während die den Interrupt auslösende Schaltung (Peripheriebaustein) die unteren 8 Bits der Adresse liefert. Diese Verfahrensweise ermöglicht es, Interrupt-Routinen dynamisch irgendwo im Speicher mit absolut geringster Zugriffszeit zu dieser Routine abzuspeichern.

### Speicher-Auffrisch-Register (R)

Die CPU U 880 enthält ein Speicher-Auffrisch-Register, das dafür sorgt, daß dynamische Speicher genauso benutzt werden können wie statische. Dieses 7-Bit-Register wird automatisch nach jedem Befehlsaufruf erhöht. Die Daten im Auffrisch-Zähler werden auf den unteren Teil des Adressenbusses mit einem Refresh-Steuersignal abgesetzt, während die CPU den aufgerufenen Befehl dekodiert und ausführt. Diese Art der Auffrischung ist vollständig programmtransparent und senkt nicht die Arbeitsgeschwindigkeit der CPU. Der Programmierer kann das Register R zu Testzwecken laden, aber das Register wird normalerweise nicht vom Programmierer benutzt.

### Akkumulator und Flag-Register

Die CPU enthält zwei unabhängige 8-Bit-Akkumulatoren A, A' und verbunden damit zwei 8-Bit-Flag-Register F, F'. Der Akkumulator



enthält die Ergebnisse der 8-Bit-Rechenoperationen oder logischen Operationen, wohingegen die Flag-Register die speziellen Bedingungen für die 8- oder 16-Bit-Operationen anzeigen, z. B. wenn das Ergebnis einer Operation gleich Null oder ungleich Null ist. Der Programmierer wählt das Akkumulator-Flag-Paar, mit dem er arbeiten möchte, durch einen einzigen Tausch-Befehl, so daß er mit jedem Paar arbeiten kann.

### Allgebrauchsregister

Es gibt zwei zueinander passende Sätze von Allgebrauchsregistern. Jeder Satz enthält sechs 8-Bit-Register, die einzeln als 8-Bit-Register oder paarweise als 16-Bit-Register durch den Programmierer verwendet werden können. Der eine Satz wird mit BC, DE und HL und der Alternativsatz mit BC', DE' und HL' bezeichnet.

Zu jeder beliebigen Zeit kann der Programmierer einen Satz mittels eines Austauschbefehls zur Benutzung auswählen.

In Systemen, wo schnelle Interruptbehandlung erforderlich ist, kann ein Satz von Allgebrauchsregister und ein Akkumulator mit Steuerflags für die Behandlung dieser schnellen Routinen reserviert werden. Nur ein einfacher Austauschbefehl ist erforderlich, um die Anfangsbedingung der Routine zu laden. Das verkürzt wesentlich die Interruptbehandlungszeit. Diese Allgebrauchsregister sind in einem großen Anwendungsbereich durch den Programmierer zu nutzen. Sie vereinfachen auch die Programmierung, besonders bei auf ROM orientierten Systemen, bei denen nur ein kleiner externer Lese-/Schreibspeicherbereich verfügbar ist.

Neben den CPU-Registern existieren, wie aus Bild 2.2. ersichtlich ist, noch weitere Funktionseinheiten.

### Rechenwerk und logische Einheit (ALU= Arithmetical Logical Unit)

Die arithmetischen und logischen 8-Bit-Befehle der CPU werden in der ALU ausgeführt. Intern steht die ALU mit den Registern und dem internen Datenbus in Verbindung. Die Funktionsarten, die durch

die ALU ausgeführt werden, sind folgende:

- Addition
- Subtraktion
- logisches UND
- logisches ODER
- logisches exklusiv ODER
- Vergleichen
- Links-/Rechtsverschiebung oder zyklische Verschiebung (arithmetisch und logisch)
- Erhöhen um 1
- Erniedrigen um 1
- Bit-Setzen
- Bit-Löschen
- Bit-Testen

### Befehlsregister und CPU-Steuerung

Wenn ein Befehl vom Speicher geholt worden ist, wird er ins Befehlsregister geladen und dekodiert. Das Steuerteil führt diese Funktion durch, erzeugt alle Signale die erforderlich sind, um Daten von oder zu den Registern zu lesen oder zu schreiben, die ALU zu steuern, gibt diese Signale aus und liefert alle extern erforderlichen Steuersignale.

### 2.2.2. Interruptsystem

Der Zweck eines Interrupts besteht darin, es den peripheren Geräten zu ermöglichen, die CPU-Operation in einer sinnvollen Weise zu unterbrechen und die CPU zu zwingen, daß eine Routine zur Bedienung der Peripherie (Interruptservice-Routine) gestartet wird. Gewöhnlich sind in diesen Routinen Austauschoperationen für Daten Status- oder Steuerinformationen zwischen der CPU und der Peripherie eingeschlossen. Wenn die Bedienungsroutine abgearbeitet ist, kehrt die CPU zu der Operation zurück, bei der sie unterbrochen wurde.

## Interrupt-Annahme/-Abweisung

Die CPU U 880 hat zwei Interrupteingänge, einen durch die Software maskierbaren (INT) und einen nichtmaskierbaren Interrupt (NMI). Die Annahme des nichtmaskierbaren Interrupts kann durch den Programmierer nicht verhindert werden. Er wird immer angenommen, wenn ein peripheres Gerät ihn fordert. Dieser Interrupt wird i. a. für die wichtigsten Funktionen reserviert, die beim Auftreten sofort bedient werden müssen, z. B. ein bevorstehender Stromausfall.

Neben den zwei Interrupteingängen besitzt die CPU noch den BUSRQ-Steuereingang für die Busübergabe, der speziell bei DMA-Betrieb Verwendung findet. Dieser Steuereingang hat gegenüber allen Interruptanforderungen höchste Priorität.

Insgesamt ergibt sich in der CPU folgende Prioritätsbewertung:

1. Priorität: Bus-Request (BUSRQ)
2. " nichtmaskierbarer Interrupt (NMI)
3. " maskierbarer Interrupt (INT)

Der maskierbare Interrupt kann durch den Programmierer selektiv zugelassen oder abgewiesen werden. Damit hat der Programmierer die Möglichkeit, Interrupte abzuweisen, wenn während bestimmter Perioden ein Verhalten realisieren muß, in dem ein Interrupt nicht zulässig ist.

In der CPU U 880 gibt es ein Annahme-Flip-Flop (IFF1), das durch den Programmierer mit den Befehlen Interrupt-Annahme (Enable interrupt-EI) bzw. Interrupt-Abweisen (Disable interrupt-DI) ein- bzw. ausgeschaltet werden kann. Wenn IFF1 ausgeschaltet ist, kann durch die CPU kein Interrupt angenommen werden.

Die Programmierung von Interrupts wird in Abschnitt 4. ausführlich beschrieben.

Tatsächlich gibt es in der CPU U 880 2 Annahme-Flip-Flops:

IFF1 und IFF2

IFF1

IFF2

Verhindert aktuell  
die Annahme von  
Interrupts

zeitweiliger Speicher-  
platz für IFF1

Die Stellung von IFF1 wird benutzt, um aktuelle Interrupts abzuweisen, während IFF2 nur als zeitweiliger Speicherplatz für IFF1 dient. Der Zweck, IFF1 zu speichern, ist folgender:

Ein RESET der CPU schaltet u. a. IFF1 und IFF2 aus, so daß Interrupts abgewiesen werden. Sie können zu beliebiger Zeit durch den Programmierer mit dem EI-Befehl zugelassen werden. Wenn ein EI-Befehl ausgeführt ist, wird eine schon anliegende Interruptanforderung erst nach der Abarbeitung des auf EI folgenden Befehls angenommen. Diese Verzögerung um einen Befehl ist für den Fall erforderlich, wenn der auf EI folgende ein Rücksprung (return) ist, da ein Interrupt nicht zugelassen werden kann, bis der Rücksprung abgearbeitet ist.

Der EI-Befehl setzt sowohl IFF1 als auch IFF2 in den Annahme-Zustand. Wenn ein Interrupt durch die CPU angenommen wird, werden sowohl IFF1 als auch IFF2 automatisch ausgeschaltet, um weitere Interrupts zu verhindern, bis der Programmierer einen neuen EI-Befehl benutzt, d. h. in den oben genannten Fällen sind IFF1 und IFF2 immer gleich.

Der Zweck, in IFF2 den Zustand von IFF1 zu speichern, wird deutlich, wenn ein nichtmaskierbarer Interrupt auftritt. Wenn ein nichtmaskierbarer Interrupt angenommen wird, wird IFF1 ausgeschaltet, um weitere Interrupts zu verhindern, bis der Programmierer sie wieder zulassen will. Folglich werden, nachdem ein nichtmaskierbarer Interrupt angenommen wurde, maskierbare Interrupts abgewiesen, aber der vorherige Zustand von IFF1 ist gerettet worden, so daß der komplette Zustand der CPU, wie er vor dem nichtmaskierbaren Interrupt bestanden hatte, wieder hergestellt werden kann. Wenn der Befehl "Laden des Akkumulators vom Register

I" (LD A,I) oder der Befehl "Laden des Akkumulators vom Register R" (LD A,R) ausgeführt ist, ist der Zustand von IFF2 in das Paritäts-Flag überführt worden, wo er getestet oder gespeichert werden kann.

Eine zweite Methode, den Zustand von IFF1 zurückzugewinnen, ist die Abarbeitung des Befehls "Rücksprung vom nichtmaskierbaren Interrupt" (RETN). Da dieser Befehl anzeigt, daß die Behandlungsroutine eines nichtmaskierbaren Interrupts abgearbeitet ist, wird der Inhalt von IFF2 nach IFF1 überführt, so daß der Zustand von IFF1 automatisch so wiederhergestellt ist, wie er vor der Annahme des nichtmaskierbaren Interrupts bestanden hatte.

Nachfolgend ist die Wirkung verschiedener Befehle auf die zwei Interrupt-Akknahme-Flip-Flops zusammengestellt:

Aktion	IFF1	IFF2	
RESET	0	0	
DI	0	0	
EI	I	I	
LD A,I	.	.	IFF2 Paritäts-Flag
LD A,R	.	.	IFF2 " "
Annahme von NMI	0	:	
RETN	IFF2	.	IFF2 → IFF1

"." bedeutet keine Veränderung

Da die NMI-Leitung flankengetriggert ist und ihre negativen Flanken ein NMI-Eingangs-Flip-Flop (NMI-EFF) setzen, wird beim Test der NMI-Leitung eigentlich das NMI-Eingangs-Flip-Flop abgefragt und ausgewertet.

Bei einem gesetzten NMI-Eingangs-Flip-Flop werden dann das interne NMI-Flip-Flop gesetzt und das INT-Akknahme-Flip-Flop IFF1 rückgesetzt. Wenn das NMI-Eingangs-Flip-Flop seinerseits nicht gesetzt ist, prüft die CPU nun den Zustand der INT-Leitung und setzt bei einer aktiven INT-Leitung und einer nicht aktiven Interruptverhinderung (IFF1 = 0) das interne INT-Flip-Flop.

Die Abarbeitung erfolgt dann in der Reihenfolge BUSRQ-F/F und INT-F/F.

## Interrupt-Beantwortung

CPU

### - nichtmaskierbar

Ein nichtmaskierbarer Interrupt wird durch die CPU zu jeder Zeit angenommen. Wenn dieser auftritt, ignoriert die CPU den nächsten aufzurufenden Befehl und führt dafür einen RESTART zur Adresse 0066H durch. Folglich verhält sie sich genauso, als hätte sie einen RESTART-Befehl aufgerufen, mit dem Unterschied, daß diese Adresse keine der 8 Software-RESTART-Adressen ist.

Ein RESTART ist ein Unterprogrammaufruf von einer bestimmten Adresse im Anfangsbereich des Speichers.

### - maskierbar

Die CPU kann so programmiert werden, daß sie auf einen maskierbaren Interrupt in einer der drei möglichen Arten (Mode) antwortet.

Mode 0

Bei diesem Mode kann die den Interrupt anfordernde Schaltung einen Befehl auf den Datenbus ausgeben und die CPU wird ihn ausführen. Folglich liefert die den Interrupt anfordernde Schaltung den nächsten abzuarbeitenden Befehl anstelle des Speichers. Meist wird das ein RESTART-Befehl sein, weil die den Interrupt anfordernde Schaltung nur einen Ein-Byte-Befehl einspeisen kann. Mit anderen Worten, es kann kein beliebiger anderer Befehl, wie z. B. ein 3-Byte-Unterprogrammaufruf ausgeführt werden. Die Taktzahl für die Ausführung dieses Befehls ist um 2 Takte größer

als die normale Zahl für diesen Befehl. Das kommt daher, weil die CPU automatisch 2 WAIT-Zustände in den Interrupt-Antwortzyklus einfügt, um der externen Logikkette (daisy chain) genügend Zeit für die Prioritätssteuerung zur Verfügung zu stellen.

Nach dem RESET ist die CPU immer automatisch auf Mode 0 gesetzt.

#### Mode 1

Wenn dieser Mode durch den Programmierer ausgewählt ist, wird die CPU einen Interrupt mit einem RESTART 0038H beantworten. Folglich ist diese Antwort mit der auf einen nichtmaskierbaren Interrupt identisch mit der Ausnahme, daß jetzt die Adresse 0038H aufgerufen wird. Ein weiterer Unterschied besteht darin, daß die erforderliche Zyklenzahl um 2 gegenüber den normalen RESTART entsprechend den zwei eingeführten WAIT-Zuständen vergrößert ist.

#### Mode 2

Dieser Mode ist die leistungsfähigste der Interrupt-Antwort-Varianten.

Mit einem einzigen 8-Bit-Byte kann vom Benutzer ein indirekter Unterprogrammaufruf zu einem beliebigen Speicherplatz durchgeführt werden. Bei diesem Mode stellt der Programmierer eine Tabelle mit 16-Bit-Startadressen für jede Interrupt-Behandlungsroutine auf. Diese Tabelle kann irgendwo im Speicher untergebracht sein. Wenn ein Interrupt angenommen wird, muß ein 16-Bit-Zeiger gebildet werden, um die Startadresse der gewünschten Interrupt-Behandlungsroutine aus der Tabelle holen zu können. Die oberen 8 Bit des Zeigers werden aus dem Inhalt des I-Registers gebildet. Das Register I muß zuvor mit dem vom Programmierer gewünschten Wert geladen werden, d. h. LD I,A. Zu beachten ist, daß ein RESET der CPU auch das Register I zurückstellt, d. h., daß dort eine Null eingeschrieben wird. Die unteren 8 Bit müssen von der den Interrupt anfordernden Schaltung geliefert werden, wobei das niederwertigste Bit eine 0 sein muß. Das ist deshalb notwendig, weil der 16-Bit-Zeiger dazu verwendet wird, zwei aufeinanderfolgende Bytes aus einer Inter-

rupttabelle zu holen, um die vollständige 16-Bit-Startadresse der Behandlungsroutine zu bilden. Die Startadressen der Interrupt-Routinen müssen in der Tabelle immer an geraden Speicherplätzen beginnen.

#### Startadressen-Tabelle der Interrupt-Service-Routinen

unterer Teil	gewünschte Startadresse ausgewählt durch
oberer Teil	
	I-Reg.
	Inhalt
oberer Teil	8 Bits von Peripherie Bit 0 = 0  unterer Teil

Das erste Byte jeder Adresse in dieser Tabelle ist der niederwertige Teil der Adresse. Der Programmierer muß selbstverständlich diese Tabelle mit den gewünschten Adressen füllen, bevor ein Interrupt angenommen werden darf.

Zu beachten ist, daß diese Tabelle zu jeder Zeit durch den Programmierer verändert werden kann, wenn verschiedene periphere Schaltungen mit verschiedenen Behandlungsroutinen bedient werden sollen. Voraussetzung dafür ist, daß die Tabelle in einem Lese-Schreib-Speicher untergebracht ist.

Wenn eine den Interrupt anfordernde Schaltung den unteren Teil des Zeigers liefert, kellert die CPU den Befehlszählerstand automatisch. Sie holt dann die Startadresse aus der Tabelle und führt einen Sprung zu dieser Adresse aus. Dieser Antwort-Mode benötigt 19-Takt-Perioden (7, um die unteren 8 Bit von der den Interrupt anfordernden Schaltung aufzurufen; 6, um den Befehlszähler zu retten und 6, um die Sprungadresse zu bilden).

Zu beachten ist, daß die peripheren Schaltungen des U 880-Systems für die Interrupt-Prioritäten eine Logikkettenstruktur (daisy chain structure) aufweisen. Der Schaltkreis, der den In-



terrupt anfordert, liefert der CPU während der Interrupt-Aufnahme automatisch einen programmierten Vektor. Nähere Informationen dazu können aus dem Abschnitt über die Peripherieschaltkreise entnommen werden.

## 2.3. Aufbau und Arbeitsweise des PIO U 855

### 2.3.1. Schaltkreisbeschreibung

Der PIO U 855 ist ein Parallel-Ein-/Ausgabe-Baustein mit zwei TTL-kompatiblen Kanälen (TTL-Transistor-Transistor-Logik). Er stellt die Verbindung zwischen der CPU und peripheren Geräten her, ohne daß eine zusätzliche Logik erforderlich ist.

Die zwei 8 Bit-bidirektionalen Kanäle (Ports) sind mit Einrichtungen für Quittungsbetrieb ("handshaking") versehen.

Eigenschaften des U 855 D:

- Interruptmöglichkeit im Quittungsbetrieb für schnelle Anforderungsbearbeitung
- Folgende Betriebsarten sind möglich:
  - Byte-Ausgabe (Betriebsart 0)
  - Byte-Eingabe (Betriebsart 1)
  - Byte-Ein/Ausgabe (bidirektionaler Betrieb, nur für Port A möglich, Betriebsart 2)
  - Bit-Ein/Ausgabe (Betriebsart 3)
- Interruptbearbeitung kann den Bedingungen des peripheren Gerätes angepaßt programmiert werden.
- Automatische Interrupt-Vektorerzeugung und Prioritätskodierung durch Kaskadierung der Bausteine (daisy-chain priority interrupt logic).
- Ausgänge des Ports B für den direkten Anschluß von Darlington-Transistoren geeignet.

- Alle Ein- und Ausgänge sind TTL-kompatibel.

In Bild 2.4. ist das Blockschaltbild des U 855 dargestellt.  
Es sind enthalten:

- Interface zur CPU
- Logik für I/O-Port A und B
- interne Steuerlogik
- Interrupt-Steuerlogik

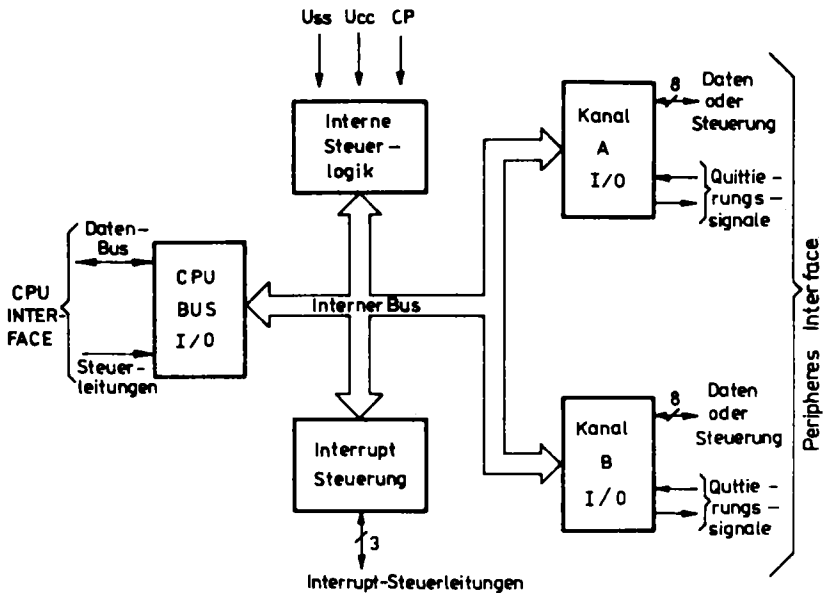


Bild 2.4. : Blockschaltbild der inneren Struktur des U 855 D

Bild 2.5. zeigt einen einzelnen Ein-Ausgabe-Kanal, bestehend aus:

- 2-Bit-Betriebsarten-Register;  
wird von der CPU zur Festlegung auf eine der 4 Betriebsarten geladen
- 8-Bit-Ausgabe-Register;  
dient der Datenübertragung von der CPU an die Peripherie
- 8-Bit-Eingabe-Register;  
dient der Datenübertragung von der Peripherie an die CPU
- 2-Bit-Maskierungssteuerregister (nur Betriebsart 3);  
wird von der CPU geladen, um festzulegen, welcher Zustand der peripheren Schaltung aktiv sein soll (Low oder High) und welche Verknüpfungsbedingung zur Interrupt-Signal-Erzeugung die einflußnehmenden Anschlüsse erfüllen sollen (UND-Bedingung, wenn alle einflußnehmenden Anschlüsse aktiv sind, bzw. ODER-Bedingung, wenn mindestens einer der einflußnehmenden Anschlüsse aktiv ist)
- 8-Bit-Maskierungsregister (nur Betriebsart 3);  
wird von der CPU geladen; mit seinem Inhalt wird festgelegt, welche Port-Anschlüsse auf die Erzeugung einer Interrupt-Anforderung Einfluß nehmen
- 8-Bit-Ein-Ausgabe-Auswahlregister (nur Betriebsart 3);  
wird von der CPU geladen; mit seinem Inhalt wird definiert, welche Anschlüsse des Ports Ausgänge und welche Eingänge sein sollen

Außerdem ist jedem Port ein 7-Bit-Vektorregister zugeordnet, welches von der CPU zur Festlegung des niederwertigen Teils (mit Ausnahme von Bit 0) des Interrupt-Vektors geladen wird. Mit dessen Hilfe wird später die Adresse der zugehörigen Interrupt-Service-Routine aufgesucht.

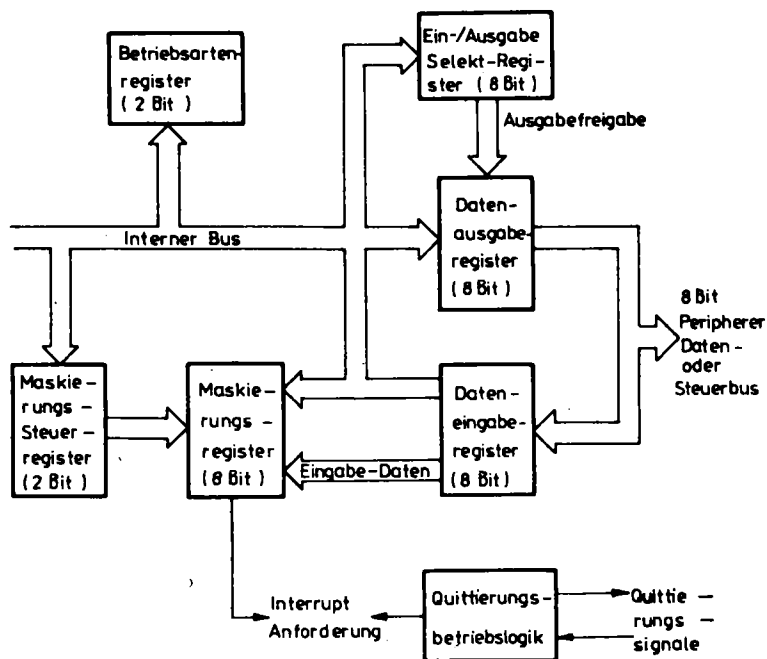


Bild 2.5.: Blockschaltbild eines Kanals

Die genaue Signalbelegung und Bedeutung der einzelnen Anschlüsse enthält Tabelle 2.3.

Tabelle 2.3. Signalbelegung U 855

Signalbezeichnung	Erläuterung
DO ... D7	Datenbus bidirektional, Tri-state
B/ $\bar{A}$	Kanalauswahl Eingang, High = Kanal B
C/ $\bar{D}$	Umschaltung Steuerwort/Datenwort Eingang, High = Steuerwort
$\bar{CE}$	Bausteinauswahl Eingang, Low-aktiv, Aktivierung ist Voraussetzung für E/A-Lese- oder Schreiboperation mit CPU
$\bar{MT}$	Maschinenzyklus 1 Eingang, Low-aktiv, CPU-Steuerbus-Signal
$\bar{TORQ}$	Ein-/Ausgabeanforderung Eingang, Low-aktiv, CPU-Steuerbus-Signal
$\bar{RD}$	Lesen Eingang, Low-aktiv, CPU-Steuerbus-Signal
C	Systemtakt
$\bar{INT}$	Interruptanforderung Ausgang, Open-Drain, Low-aktiv Aktivierung des Ausgangs signalisiert der CPU die Anmeldung eines Interrupts
IEI	Interrupt-Freigabe-Eingang Eingang, High-aktiv Verbindung von IEI mit IEO des nächsthöherpriorisierten E/A-Schaltkreises ermöglicht Interruptprioritäts-Kaskadierung High-Pegel an IEI bedeutet, daß momentan kein Interrupt höherer Priorität abgearbeitet oder angemeldet wird (Ausnahme: bei noch nicht bestätigtem Interrupt eines höherpriorisierten E/A-Schaltkreises, RETI-Dekodierung)
IEO	Interrupt-Freigabe-Ausgang Ausgang, High-aktiv

Signalbezeichnung	Erläuterung
AO ... A7	<p>IEO führt nur dann High-Pegel, wenn der Eingang IEI desselben Schaltkreises High-Pegel erhält und kein eigener Interrupt abgearbeitet oder angemeldet wird (Ausnahme: bei noch nicht bestätigtem Interrupt eines höherpriorisierten E/A-Schaltkreises, RETI-Dekodierung)</p>
ARDY	<p>Ein-/Ausgänge Port A, bidirektional, Tri-state Quittung, Kanal A (A = READY) Ausgang, High-aktiv Bedeutung ist abhängig von Betriebsart:</p> <ol style="list-style-type: none"> <li data-bbox="379 531 985 718">1. MODE 0: Das Signal wird aktiv, um anzuzeigen daß das Ausgaberegister des Kanals geladen ist und daß die Daten abgerufen werden können. Nach quittierter Beendigung der Übernahme durch die periphere Schaltung wird das Signal inaktiv.</li> <li data-bbox="379 733 985 851">2. MODE 1: Das Signal ist aktiv, wenn das Ausgaberegister des Kanals leer und es bereit ist, Daten vom peripheren Gerät zu übernehmen.</li> <li data-bbox="379 866 985 1044">3. MODE 2: Das Signal ist aktiv, wenn Daten i Ausgaberegister vom Kanal A für einen Transfer zum peripheren Gerät verfügbar sind. I dieser Betriebsart liegen die Daten am Kanal A-Datenbus nicht an, sofern nicht <u>ASTB</u> aktiv ist.</li> <li data-bbox="379 1059 985 1118">4. MODE 3: Das Signal ist nicht verwendbar und liegt auf Potential "Low".</li> </ol>
<u>ASTB</u>	<p>Kanal A-Strobe Eingang, Low-aktiv Die Bedeutung dieses Signals hängt von der Betriebsart ab, die für Kanal A gewählt wurde:</p> <ol style="list-style-type: none"> <li data-bbox="379 1262 985 1317">1. MODE 0: Der Strobeimpuls wird von der Peripherie abgegeben, um die Daten aus dem Aus</li> </ol>

Signalbezeichnung	Erläuterung
BO ... B7	<p>gaberegister zu übernehmen. Das Ende des Strobeimpulses (positive Flanke) gilt als Quittung der erfolgten Übernahme.</p> <p>2. MODE 1: Der Strobeimpuls wird von der Peripherie abgegeben, um Daten von der Peripherie in das Eingaberegister des Kanals zu laden. Die Daten werden in den U 855 geladen, wenn das Signal aktiv ist.</p> <p>3. MODE 2: Wenn das Signal aktiv ist, werden Daten vom Ausgaberegister des Kanals A an den bidirektionalen Datenbus des Kanals A gelegt. Die positive Flanke des Strobeimpulses bestätigt den Empfang der Daten.</p> <p>4. MODE 3: Der Strobeimpuls ist intern verboten.</p>
BRDY	<p>Ein-/Ausgänge Port B bidirektional, Tri-state Quittung Kanal B Ausgang, High-aktiv Bedeutung entspricht ARDY mit der folgenden Ausnahme: In der bidirektionalen Betriebsart des Kanals A ist das Signal "High", wenn das Eingaberegister des Kanals A leer und bereit ist, Daten vom peripheren Gerät zu übernehmen.</p>
<u>BSTB</u>	<p>Kanal B-Strobe Eingang, Low-aktiv Bedeutung entsprechend <u>ASTB</u> mit der folgenden Ausnahme: In der bidirektionalen Betriebsart des Kanals A überführt dieses Signal Daten vom peripheren Gerät in das Eingaberegister des Kanals A.</p>

## 2.3.2. Erläuterung der einzelnen Betriebsarten

### 2.3.2.1. Betriebsart Byte-Ausgabe (Mode 0)

Beim Ausführen eines Ausgabebefehls durch die CPU werden die Daten über den Datenbus in das Ausgabe-Register des vorgewählten Ports eingeschrieben. Nach dem Ende des Übergabe-Signals wird nach der nächsten fallenden Flanke von C das Ready-Signal aktiv. Damit wird nach außen hin angezeigt, daß die Daten aus dem Ausgaberegister abgerufen werden können.

Das Ready-Signal bleibt aktiv bis die Übernahme der Daten durch die periphere Schaltung abgeschlossen ist ("Quittung"). Die dem Ende des Strobe-Impulses folgende fallende Flanke von C setzt das Ready-Signal wieder in den inaktiven Zustand.

Mit der steigenden Flanke des Strobe-Signals wird ein INT-Signal ausgelöst, unter der Voraussetzung, daß das Interruptfreigabe-Flip-Flop gesetzt ist und das anfordernde Port im betrachteten Zeitpunkt die höchste Priorität aufweist.

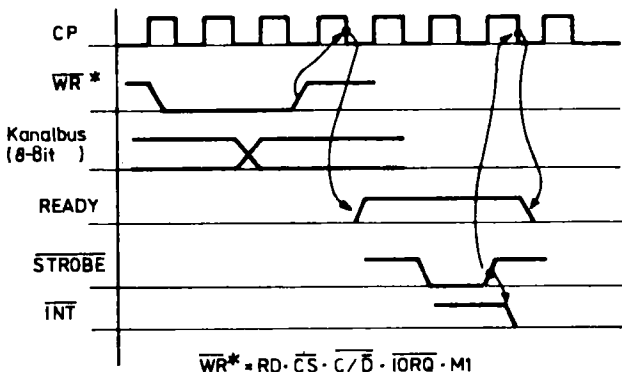


Bild 2.6.: Möglicher Zeitablauf in Mode 0



### 2.3.2.2. Betriebsart Byte-Eingabe (Mode 1)

Während des Low-Zustandes von  $\overline{\text{Strobe}}$  werden die an den Eingängen des Ports anstehenden Daten in das Eingaberegister eingeschrieben. Nach der steigenden Flanke von  $\overline{\text{Strobe}}$  versetzt die nächste fallende Flanke von C das Ready-Signal in den inaktiven Zustand.

Damit wird angezeigt, daß sich im Eingaberegister Daten befinden, die noch nicht von der CPU gelesen wurden.

Falls das Interrupt-Freigabe-Flip-Flop gesetzt ist und höchste Priorität vorliegt, wird mit der steigenden Flanke von  $\overline{\text{Strobe}}$  eine  $\overline{\text{INT}}$ -Anforderung ausgelöst.

Nach Abschluß des Lesevorganges wird von der darauffolgenden fallenden Flanke von C das Ready-Signal aktiv geschaltet, als Zeichen dafür, daß die CPU die Daten gelesen hat und neue Daten zugeführt werden können.

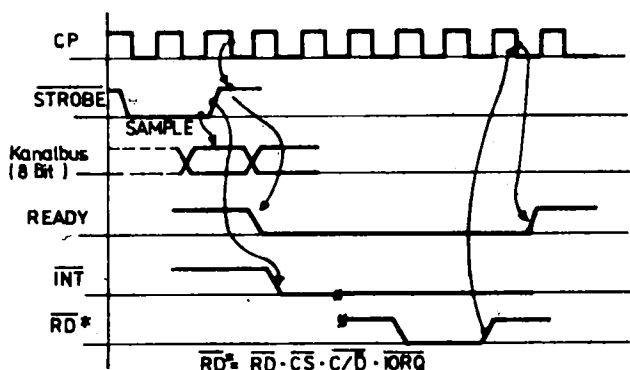


Bild 2.7.: Möglicher Zeitablauf in Mode 1

### 2.3.2.3. Betriebsart Byte-Ein-/Ausgabe (bidirektional) (Mode 2)

Diese Betriebsart kombiniert die Betriebsarten "Byte-Eingabe" und "Byte-Ausgabe", wobei alle 4 Quittungs-Leitungen des U 855 D und die 8 Datenleitungen des Ports A benutzt werden.

Die Quittungsleitungen von Port A werden für die Ausgabe-, die Quittungsleitungen von Port B für die Eingabesteuerung verwendet

Die Datenausgabe an das Port A kann nur während  $\overline{ASTB} = \text{Low}$  erfolgen. Seine steigende Flanke kann zur Übernahme der Daten durch die periphere Schaltung benutzt werden.

Wird Port A in der Betriebsart "Byte-Ein-/Ausgabe" betrieben, ist Port B in Betriebsart "Bit-Ein-/Ausgabe" zu benutzen.

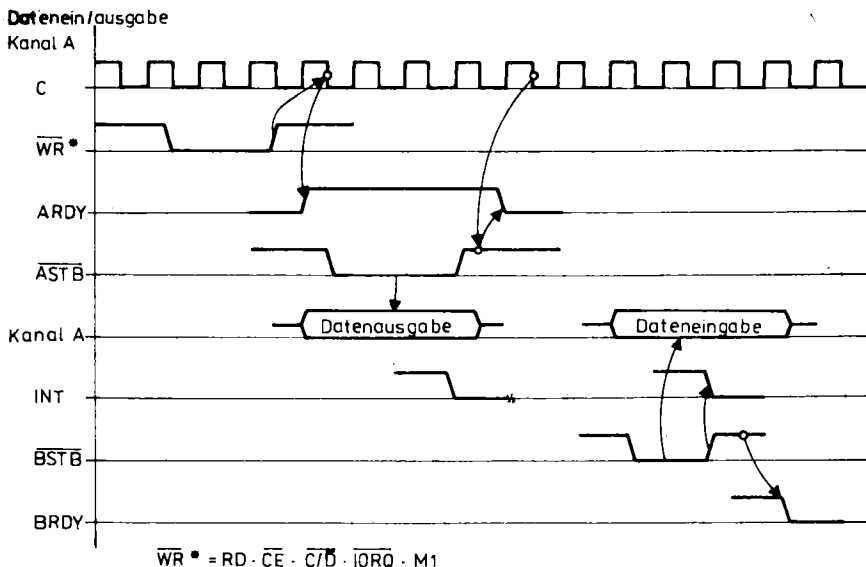


Bild 2.8. : Zeitablauf in Mode 2

#### 2.3.2.4. Bit-Ein-/Ausgabe (Mode 3)

In dieser Betriebsart wird nicht mit Quittungssignalen gearbeitet. Die Ein- und Ausgabe von Daten kann zu jedem beliebigen Zeitpunkt erfolgen. Das Ausgeben von Daten an das Ausgabe-Register erfolgt nach dem gleichen Zeitschema wie in Betriebsart Byte-Ausgabe.

Bei der Eingabe setzen sich die der CPU zugeführten Daten zusammen aus den Daten des Eingaberegisters (dies gilt für die Bits, die im Ein-Ausgabe-Wahl-Register als Eingänge definiert wurden) und aus dem Inhalt des Ausgaberegisters (dies gilt für diejenigen Bits, die im Ein-Ausgabe-Wahl-Register als Ausgänge definiert wurden).

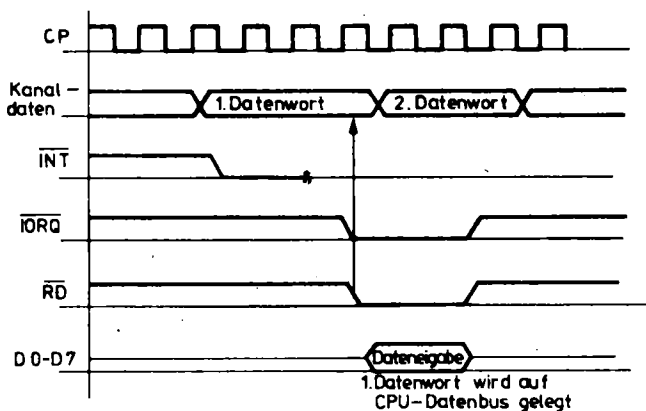


Bild 2.9. : Zeitablauf in Mode 3 (dargestellt "Lesen")

### 2.3.3. Interrupt-Bearbeitung

Während  $\overline{M1}$  kann der Interrupt-Zustand der Steuerlogik einer peripheren Schaltung nicht verändert werden. Dies gibt dem Interruptsignal Zeit, die prioritätsbestimmende Kaskadierung der angeschlossenen peripheren Schaltungen zu durchlaufen ("Daisy chain priority logic").

Derjenige periphere Schaltkreis, bei dem während des Interrupt-Akzeptanzzyklus am Eingang IEI High anliegt und dessen Ausgang IE0 Low liefert, gibt seinen vorher programmierten 8-Bit-Interrupt-Vektor auf den Datenbus aus. Der Ausgang IE0 liefert solange Low, bis von der CPU eine RETI (Return from Interrupt  $\hat{=}$  Rückkehr von der Unterbrechung) - Anweisung bei IEI = High ausgeführt wird.

Die RETI-Anweisung wird im U 855 D dekodiert.

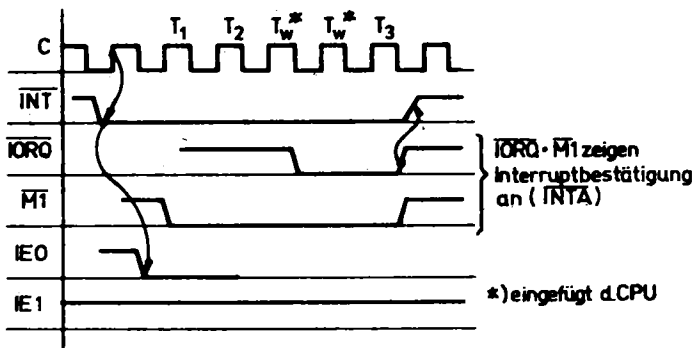


Bild 2.10.: Interrupt-Bestätigungs-Zyklus

### 2.3.4. Programmierung des PIO's

Im Grundzustand, d. h. nach Rücksetzen des PIO's sind vorerst alle Kanäle inaktiv. Erst durch eine Programmierung (Initialisierung genannt) durch die CPU in Form von OUT-Befehlen wird die PIO veranlaßt, die jeweils gewünschte Betriebsart sowie Interruptverhalten einzunehmen.

Die dabei gesendeten Daten (in diesem Falle als Steuerwort bezeichnet) müssen nachfolgende Zusammensetzung besitzen.

#### 2.3.4.1. Wahl der Betriebsart

Dies geschieht über die höchstwertigen 2 Bit (M1 und M0) eines Steuerwortes folgenden Formates an das betreffende zu programmierende Port:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	x	x	1	1	1	1
Definition der Betriebsart		nicht benutzt		Identifikation des Steuerwortes als Betriebsartenauswahlwort			

Betriebsart	M1	M0
Byte-Ausgabe	0	0
Byte-Eingabe	0	1
Byte-Ein-/Ausgabe	1	0
Bit-Ein-/Ausgabe	1	1

#### 2.3.4.2. Ein- bzw. Ausgabedefinition bei Betriebsart Bit-Ein-/Ausgabe

Wurde die Betriebsart Bit-Ein-/Ausgabe gewählt, so wird das als

nächstes übertragene Steuerwort für dieses Port zur Definition der einzelnen Port-Anschlüsse als Ein- bzw. Ausgänge verwendet. Eine "0" entspricht dabei einer Ausgangszuordnung, einer "1" wird ein Eingang zugeordnet.

Format des zusätzlichen Steuerwortes:

D7	D6	D5	D4	D3	D2	D1	D0
I/O <sub>7</sub>	I/O <sub>6</sub>	I/O <sub>5</sub>	I/O <sub>4</sub>	I/O <sub>3</sub>	I/O <sub>2</sub>	I/O <sub>1</sub>	I/O <sub>0</sub>

2.3.4.3. Laden des Interrupt-Vektors

Erfolgt durch Laden eines Steuerwortes folgenden Formates an das ausgewählte Port:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

Durch L-Signal auf der Leitung D0 wird das Steuerwort als Interrupt-Vektor identifiziert.

2.3.4.4. Interrupt-Steuerung

Das Interrupt-Freigabe-Flip-Flop kann durch zwei Steuerworte beeinflusst werden (Die für die Bits D3 ... D0 im folgenden angegebenen Belegungen definieren die beiden Steuerworte als Interrupt-Steuerworte.). In den Betriebsarten 0, 1, und 2 ist folgendes Steuerwort anzuwenden:

D7	D6	D5	D4	D3	D2	D1	D0
Interrupt-freigabe	X	X	X	0	0	1	1

D7  
Low Interrupt-Flip-Flop rückgesetzt, der entsprechende Kanal ist nicht interruptfähig

High Interrupt-Flip-Flop gesetzt, der entsprechende Kanal ist interruptfähig

D6, D5, D4 Bits werden ignoriert

In Betriebsart 3 wird ein anderes Steuerwort verwendet, in dem die Bits D6, D5, D4 mit benutzt werden.

Das Steuerwort hat folgendes Format:

D7	D6	D5	D4	D3	D2	D1	D0
Interrupt-freigabe	UND/ ODER	High/ Low	nächstes Steuerwort ist Maske	0	1	1	1

D7 Interrupt-Flip-Flop rückgesetzt, der entsprechende Kanal ist nicht interruptfähig

High Interrupt-Flip-Flop gesetzt, der entsprechende Kanal interruptfähig

D6 ODER-Funktion der auf die Erzeugung eines Interrupts einflußnehmenden Kanalleitungen

High UND-Funktion der auf die Erzeugung eines Interrupts einflußnehmenden Kanalleitungen

D5 Überwachung der auf die Erzeugung eines Interrupts einflußnehmenden Kanalleitungen auf Low-Zustand (Low-aktiv)

High Überwachung der auf die Erzeugung eines Interrupts einflußnehmenden Kanalleitungen auf High-Zustand (High-aktiv)

D4 Es folgt keine Maske.

**High** Es ist erforderlich, ein weiteres Steuerwort auszugeben (Maske). Dieses dient dazu, diejenigen Kanalleitungen zu definieren, die an der Erzeugung einer Interruptanforderung beteiligt sein sollen. Diejenigen Bits, welche mit Maskierungsbit  $MB_n = 0$  belegt werden, werden zur Erzeugung einer Interruptanforderung herangezogen.

Format des Steuerwortes zur Maskierung:

D7	D6	D5	D4	D3	D2	D1	D0
<hr/>	<hr/>	<hr/>	<hr/>	<hr/>	<hr/>	<hr/>	<hr/>
$MB_7$	$MB_6$	$MB_5$	$MB_4$	$MB_3$	$MB_2$	$MB_1$	$MB_0$

## 2.4. Aufbau und Arbeitsweise des CTC U 857

### 2.4.1. Schaltkreisbeschreibung

Der CTC U 857 ist ein Schaltkreis für Zähler- und Zeitgeberfunktionen.

Eigenschaften des U 857 D:

- 4 voneinander unabhängig, software-programmierbare 8-Bit-Zähler 16-Bit-Zeitgeber-Kanäle
  - jeder Kanal wahlweise als Zähler oder Zeitgeber verwendbar
  - Verteiler durch 16 oder 256 für jeden Kanal (für die Betriebsart Zeitgeber)
  - Rückwärtszähler hält die Anzahl der bis Null auszuführenden Zählsschritte auslesebereit
  - Zeitgeber kann wahlweise von einem positiven oder negativen Triggerimpuls gestartet werden
  - jedem Kanal ist ein Interrupt-Vektor zugeordnet, Kanal-Nr. 0 hat hardwaremäßig die höchste Priorität
- beim Erreichen von programmäßig festlegbaren Zähler- oder Zeitgeberwerten ist die Erzeugung von Interrupts programmierbar



- automatische Interrupt-Vektor-Bereitstellung;  
Prioritätskodierung durch Kaskadierung der Bausteine (ohne zusätzlichen Schaltungsaufwand)
- Die Ausgänge der drei herausgeführten Kanäle (Kanal 0, 1, 2) sind für den Anschluß von Darlington-Transistoren ausgelegt.
- Alle Ein- und Ausgänge sind TTL-kompatibel
- Es wird nur eine +5 V-Versorgungsspannung benötigt.
- Einphasen 5 V-Takt
- Maximale Zählfrequenz in der Betriebsart "Zähler" =  $f_c/2$

Das Bild 2.11. zeigt das Blockschaltbild des U 857 D.

Folgende Funktionseinheiten sind enthalten:

- 4 Zähler/Zeitgeber-Kanäle
- Interface zu Daten- und Steuerbus des U 880 D (CPU)
- Interrupt-Steuerlogik

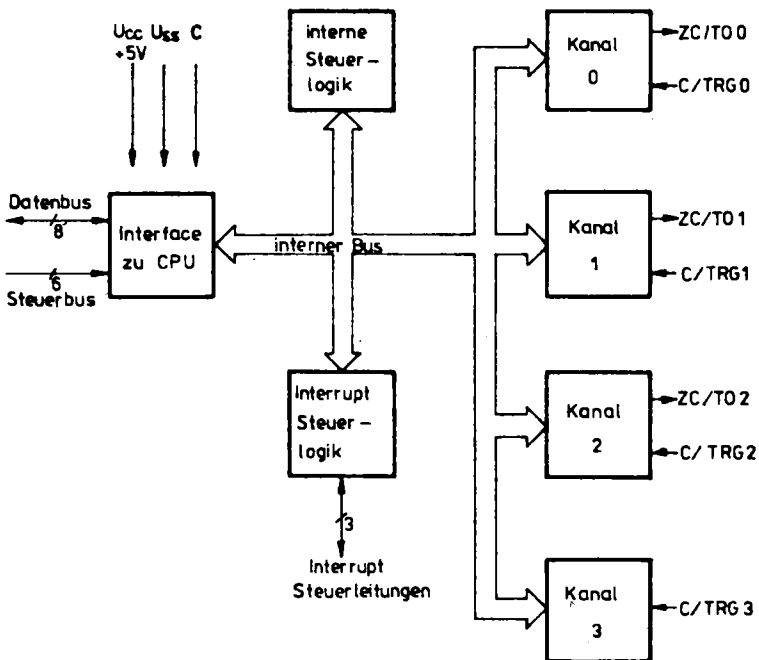


Bild 2.11.: Blockschaltbild

Im Bild 2.12. ist das Blockschaltbild eines einzelnen Kanals dargestellt.

Ein Kanal besteht aus folgenden einzelnen Einheiten:

- Zeitkonstantenregister

- Kanalsteuerregister
- Rückwärtszähler
- Vorteiler

Das Zeitkonstantenregister (8 Bit) wird von der CPU zum Initialisieren und Wiedersetzen des Rückwärtszählers beim Erreichen des Zählerstandes Null geladen.

Das Kanalsteuerregister (8 Bit) wird von der CPU zur Bestimmung der Kanalbetriebsart geladen.

Der Rückwärtszähler (8 Bit) wird entweder mit Hilfe des Anwenderprogramms oder automatisch beim Zählerstand Null auf den im Zeitkonstantenregister stehenden Wert gesetzt. Im Zeitgeberbetrieb wird der Rückwärtszähler über den Vorteiler durch den Systemtakt C, im Zählerbetrieb durch einen Impuls am Eingang C/TRG dekrementiert. Der momentane Wert des Rückwärtszählers kann sowohl im Zähler- als auch im Zeitgeberbetrieb zu jedem beliebigen Zeitpunkt von der CPU ausgelesen werden.

Der Vorteiler (8 Bit) wird nur in der Betriebsart "Zeitgeber" benutzt. Programmierbar sind die Werte 16 oder 256.

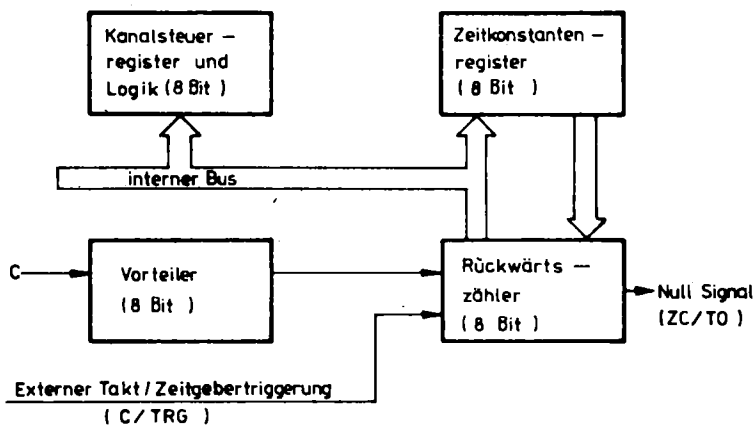


Bild 2.12.: Blockschaltbild eines CTC-Kanals

Die genaue Signalbelegung und Bedeutung der einzelnen Anschlüsse enthält Tabelle 2.4.

Tabelle 2.4.: Signalbelegung U 857

Signalbezeichnung	Erläuterung
DO ... D7	8-Bit-bidirektionaler Datenbus, Tri-state
$\overline{CE}$	Chipauswahl, Eingang, Low-aktiv
KS0; KS1	Kanalauswahl; Eingabe einer 2-Bit-Adresse, des vom Mikroprozessor angesprochenen Kanals
$\overline{MT}$	CPU-Maschinenzyklus M1; Eingang, Low-aktiv
$\overline{TORQ}$	Ein-/Ausgabe-Anforderung; Eingang, Low-aktiv
$\overline{RD}$	CPU-Leseanforderung; Eingang, Low-aktiv
IEI	Interrupt-Freigabe, Eingang
C/TRG0 C/TRG1 C/TRG2 C/TRG3	Takt/Trigger-Eingänge (Kanal 0 bis 3)  Takteingänge für Zähler bzw. Zeitgebertrig- gerung Programmierbar: High- oder Low-aktiv
$\overline{RESET}$	Rücksetzeingang; Low-aktiv  Der Zählvorgang aller Kanäle wird unterbro- chen und die Interrupt-Freigabebits der Steu- erregister aller 4 Kanäle werden zurückgesetzt.  Die Ausgänge ZC/TO...2 und $\overline{INT}$ werden in den inaktiven Zustand gebracht. Der Ausgang IEO wird gleich dem Wert am Eingang IEI gesetzt. Das Interrupt-Vektorregister wird nicht beein- flußt. Die Daten-Ein-/Ausgänge werden in den hochohmigen Zustand gebracht.

Signalbezeichnung	Erläuterung
C	Systemtakt
IEO	Interrupt-Freigabe, Ausgang; High-aktiv für Interrupt-Prioritätskette
ZC/T00	Nulldurchgang/Zeitgebermeldung-Ausgänge (Kanal 0 bis 2)
ZC/T01	
ZC/T02	
	Nullsignal des Rückwärtszählers, bzw. Meldung des Zeitgebers.
<u>INT</u>	Interrupt-Anforderung, Ausgang, Open-Drain-Ausgang, Low-aktiv

#### 2.4.2. Arbeitsweise des Schaltkreises

##### 2.4.2.1. Schreibzyklus

Im Schreibzyklus werden Kanalsteuerwort, Zeitkonstante und Interruptvektor eingeschrieben. Da der U 857 D keinen Schreibeingang (WR) hat, wird ein Schreibsignal intern aus dem RD-Signal genommen. Zu beachten ist, daß außer dem automatisch erzeugten Wartezyklus TW keine weiteren Wartezyklen beim Schreiben in die Register des U 857 D eingefügt werden dürfen.

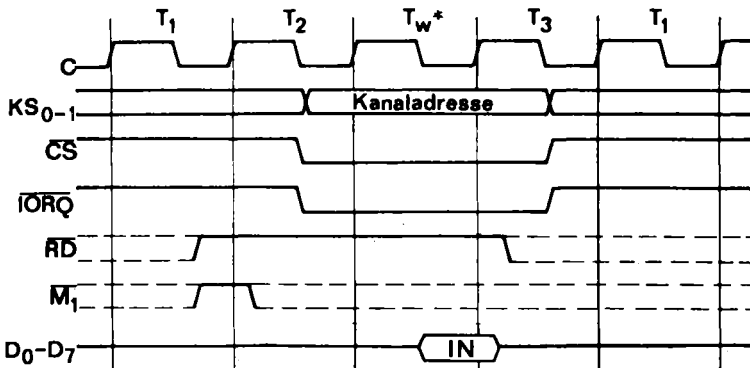


Bild 2.13.: Schreibzyklus

#### 2.4.2.2. Lesezyklus

In beiden Betriebsarten (Zähler/Zeitgeber) kann der Momentanwert jedes Kanalrückwärtszählers zu jedem beliebigen Zeitpunkt ausgelesen werden. Der auf den Datenbus ausgegebene Wert repräsentiert die Anzahl der steigenden Zähltaktflanken vor der steigenden Systemtaktflanke des Zustandes T2 im Zählerbetrieb oder den entsprechenden Zählerstand im Zeitgeberbetrieb. Auch beim Lesezyklus darf kein Wartezyklus eingefügt werden.

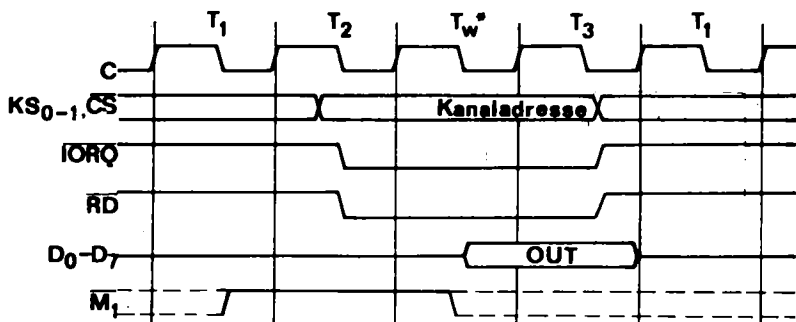


Bild 2.14.: Lesezyklus

#### 2.4.2.3. Interrupt-Quittungs-Zyklus

Die CPU quittiert nach einer gewissen Zeit die Interrupt-Anforderung des U 857 D durch die Signale  $\overline{M\overline{T}}$  und  $\overline{I\overline{O}R\overline{Q}}$ . In dieser Zeit mittelt der U 857 D intern den Kanal mit der höchsten Priorität. Zur Gewährleistung der Interruptkaskadesignale werden alle Interruptanforderungszustände der Kanäle festgehalten, solange  $\overline{M\overline{T}}$  aktiv ist. Ist IEI am Eingang des U 857 D aktiv, so bringt der Kanal mit der höchsten Priorität den Interruptvektor aus dem Vektorregister auf den Datenbus, solange  $\overline{I\overline{O}R\overline{Q}}$  aktiv ist.



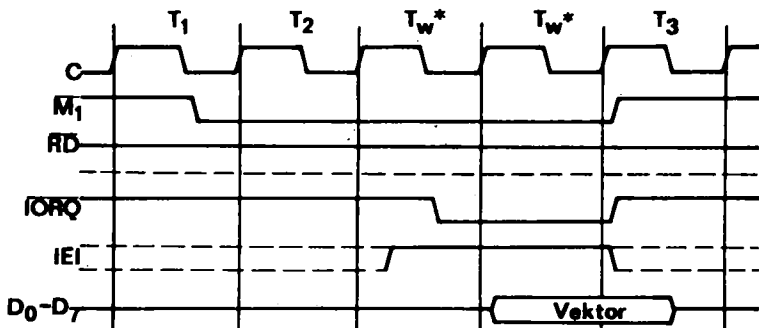


Bild 2.15.: Interrupt-Quittungs-Zyklus

#### Rückkehr vom Interrupt

Von der RETI-Anweisung wird die Interruptkette am Ende einer Interrupt-Bedien-Routine initialisiert. Die 2 Bytes der RETI-Anweisung werden im CTC intern dekodiert. Der CTC erkennt den Befehlscode EDH, daraufhin wird, wenn der IEI-Eingang aktiv ist und als nächster Befehlscode 4DH folgt, der IEO-Ausgang wieder aktiv. Damit ist die Bedienroutine dieses Kanals abgeschlossen.

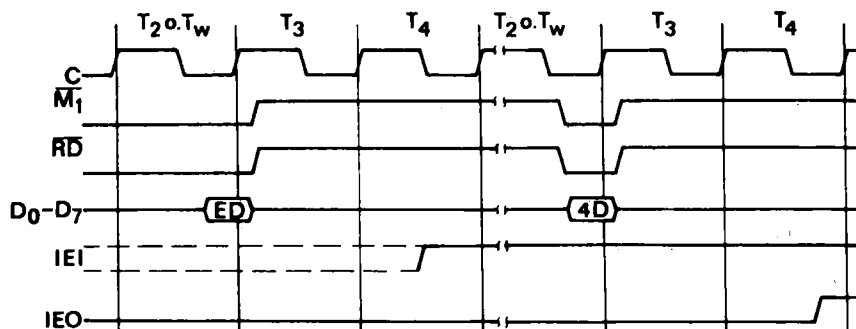


Bild 2.16.: Rückkehr vom Interrupt

#### 2.4.2.4. Prioritätskaskadierung

Bild 2.17. zeigt ein Beispiel der Interruptprioritätskaskadierung. Kanal 2 fordert einen Interrupt an und wird bedient. Danach fordert Kanal 1 ebenfalls einen Interrupt an. Da Kanal 1 höhere Priorität hat, wird er zuerst bedient und Kanal 2 wird in der Abarbeitung unterbrochen. Nachdem die Bedienung von Kanal 1 abgeschlossen ist (mit RETI-Anweisung) wird mit der Abarbeitung von Kanal 2 fortgefahren und die Bedienung zu Ende geführt.

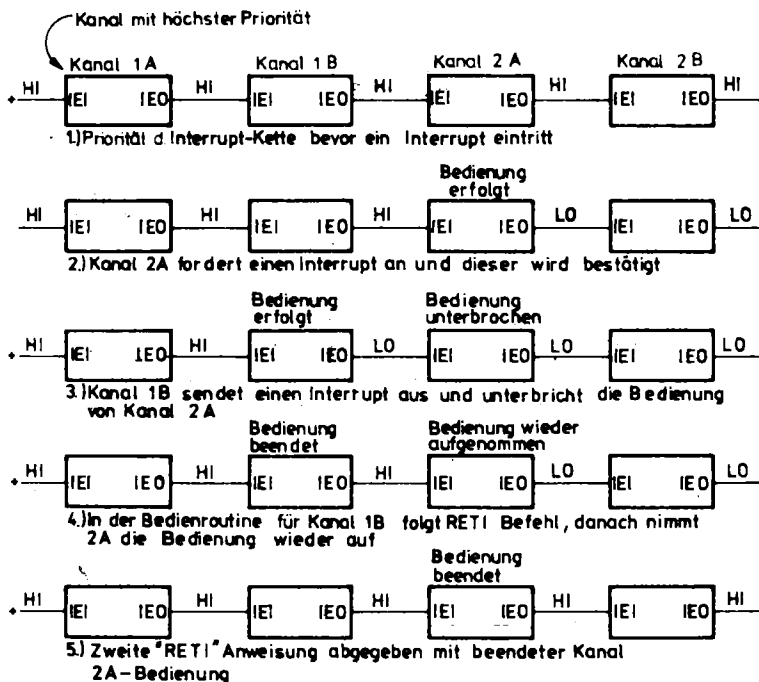


Bild 2.17.: Prioritätskaskadierung (Beispiel)

#### 2.4.2.5. Zähl- und Zeitgeber-Vorgang

In der Betriebsart "Zähler" veranlaßt die programmierte Flanke C/TRG-Eingang ein Dekrementieren des Rückwärtszählers. Das Signal kann vollkommen asynchron empfangen werden, jedoch wird eine gewisse Mindestdauer des Impulses gefordert.

Der Zähler arbeitet synchron mit dem Systemtakt. Der Zähler kann nur dekrementiert werden bei der nachfolgenden steigenden Systemtaktflanke, wenn die Aktivierung des C/TRG-Eingangs eine gewisse Mindestzeit vorher geschieht. In der Betriebsart "Zeitgeber" kann der Zeitgebervorgang von einer steigenden oder fallenden Flanke eingeleitet werden. Genau wie beim Zählerbetrieb werden diese Impulse asynchron empfangen. Ebenso wird eine bestimmte Mindestimpulsdauer und Schaltzeit des TRG-Impulses gefordert.

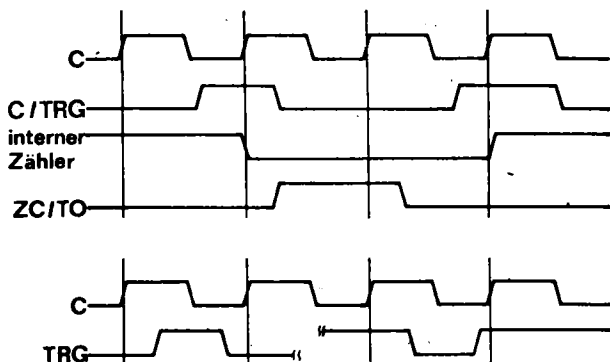


Bild 2.18.: Zähler- und Zeitgebervorgang

#### 2.4.3. Programmierung des CTC

Der Schaltkreis U 857 D wird softwaremäßig gesteuert. Grundsätzlich wird nach einer Aktivierung des Schaltkreises (CE-Freigabe) ein Steuerwort erwartet. Ob ein Wort als Steuer- oder Datenwort (zum Setzen des Zeitkonstantenregisters) erkannt wird, richtet

sich nach der Form des vorangegangenen Steuerwortes.

### 2.4.3.1. Laden des Interrupt-Vektors

Während des Interrupt-Quittungs-Zyklus wird dieser Vektor vom Kanal höchster Priorität auf den Datenbus gelegt.

Der Vektor muß dem CTC vorher über Kanal-Nr. 0 vorgegeben werden.

Format des Interruptvektorsteuerwortes:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	x	x	0
					beliebi- ger Bi- närwert		Identifiziert das Steuerwort als In- terruptvektor

Der Vektor für Kanal 0 ... 3 ergibt sich hieraus wie folgt:

Formate der Interruptvektoren während des Interrupt-Quittungs-Zyklus

für Kanal 0	D7	D6	D5	D4	D3	D2	D1	D0
	V7	V6	V5	V4	V3	0	0	0
für Kanal 1	D7	D6	D5	D4	D3	D2	D1	D0
	V7	V6	V5	V4	V3	0	1	0
für Kanal 2	D7	D6	D5	D4	D3	D2	D1	D0
	V7	V6	V5	V4	V3	1	0	0
für Kanal 3	D7	D6	D5	D4	D3	D2	D1	D0
	V7	V6	V5	V4	V3	1	1	0

Kanal 0 besitzt hardwaremäßig die höchste Priorität.

### 2.4.3.2. Format des Kanalsteuerwortes

Durch die einzelnen Bits des Kanalsteuerwortes wird die Betriebsart des jeweiligen CTC-Kanals festgelegt:

D7	D6	D5	D4
0 Interrupt gesperrt	0 Betr.-Art Zeitgeber	0 Syst.-Takt um Faktor 16 geteilt	0 negative Triggerflanke
1 Interrupt freigegeben	1 Betr.-Art Zähler	1 Syst.-Takt um Faktor 256 geteilt	1 positive Triggerflanke

↑  
Nur für Betr.-Art Zeitgeber

D3	D2	D1	D0
Triggerzeitpunkt	Zeitkonstante laden	Rücksetzen	1 (zur Kennzeichnung als Kanalsteuerwort)

↑  
Nur für Betr.-Art Zeitgeber

- D7    0 Interrupt gesperrt  
       1 Interrupt freigegeben: Interruptanforderung erfolgt jedesmal, wenn der Rückwärtszähler den Wert Null erreicht
- D6    0 Zeitgeber: Rückwärtszähler wird vom Vorteiler gestartet  
           Periode des Zählers  $c = t_c \cdot p \cdot TC$
- |                    |                               |   |
|--------------------|-------------------------------|---|
| /                  |                               | \   |
| System-<br>periode | Vorteiler<br>(16 oder<br>256) | 8-Bit-<br>Zeitkon-<br>stante<br>(max.256) |

- D5 siehe Seite 68
- D4 siehe Seite 68
- D3 0 Zeitmessung beginnt am Anfang des nächsten Maschinenzyklus, der auf das Laden der Zeitkonstante folgt, mit steigender Flanke von  $T_2$ .
- 1 Triggereingang wird zur Veranlassung des Beginns des Zeitgebervorganges freigegeben, nach der steigenden Flanke von  $T_2$  des Maschinenzyklus, der auf das Laden der Zeitkonstante folgt. Der Vorteiler wird nach 2 bzw. 3 Taktzyklen dekrementiert.  
(Zum Zeitverhalten der CPU siehe Literaturhinweis [1] der Bedienungsanleitung)
- D2 0 Auf das Kanalsteuerwort folgt keine Zeitkonstante, die Zeitkonstante ist zum Anlaufenlassen des Zeitmeßvorganges noch einzugeben.
- 1 Nächstes Kontrollwort für den betreffenden Kanal stellt Zeitkonstante für den Rückwärtszähler dar. Wird während des Zeitmeßvorganges eine neue Zeitkonstante eingegeben, so wird die alte Messung erst zu Ende geführt.
- D1 0 Kanal zählt weiter.
- 1 Abbruch der momentanen Operation.  
Falls Bit 2 = 1 ist, wird die Operation nach dem Laden einer Zeitkonstante fortgesetzt. Ist Bit 2 = 0, muß hierfür ein neues Steuerwort an den CTC übermittelt werden.

#### 2.4.3.3. Format des Zeitkonstantensteuerwortes

D7	D6	D5	D4	D3	D2	D1	D0
ZK7	ZK6	ZK5	ZK4	ZK3	ZK2	ZK1	ZK0

Hat Bit 2 des Kanalsteuerwortes den Wert "1", so wird das nachfolgende Steuerwort für den betreffenden Kanal als Zeitkonstante (Datenwort) interpretiert und ins Zeitkonstantenregister geladen. Dabei wird ein Datenwort mit 0000 0000 als Zeitkonstante = 256 interpretiert.

## 3. Befehlsbeschreibung des U 880

### 3.1. Befehlsstruktur

Die Leistungsfähigkeit eines Mikroprozessors wird neben der Operationsgeschwindigkeit wesentlich durch die Struktur und die Flexibilität seines Befehlssatzes bestimmt.

Ein Befehl ist eine Anweisung an den Rechner, eine bestimmte Operation auszuführen. Er besteht aus Operationsteil (1 - 3 Byte) und AdreSteil (1 - 2 Byte). Dabei sind folgende Befehlsstrukturen möglich:

#### 1-Byte-Befehl

Op - Code

Beispiel: 04

Erhöhe Inhalt von  
Register B um 1

#### 2-Byte-Befehl

Op - Code    Op - Code

Beispiel: CB        FF

Bitposition 7 des Re-  
gisters A wird gleich  
1 gesetzt

Op - Code    Direktoperand

Beispiel: 3E        05

Lade Register A mit 05

#### 3-Byte-Befehl

Op - Code    Op - Code    Direktoperand

Beispiel: DD        7E            03

Lade Register A mit In-  
halt der Adresse IX+3



Op - Code	Adresse	Adresse
-----------	---------	---------

Beispiel: 21      00      50      Lade Doppelregister HL  
mit Adresse 5000H

4-Byte-Befehl

Op - Code	Op - Code	Adresse	Adresse
-----------	-----------	---------	---------

Beispiel: DD      21      00      50      Lade Indexregister IX  
mit Adresse 5000H

Op - Code	Op - Code	Direktooperand	Direktooperand
-----------	-----------	----------------	----------------

Beispiel: DD      36      03      05

Lade Inhalt von Register IX+3 mit 05

Op - Code	Op - Code	Direktooperand	Op - Code
-----------	-----------	----------------	-----------

Beispiel: DD      CB      03      46

Bit 0 von Inhalt der Adresse IX+3 wird komplementiert in das Z-Flag geladen

Für den Befehl existiert ein mnemonischer Operationscode (Kürzel). Damit wird die Programmierung für den Anwender einfacher und effektiver als in maschineninterner Form. Der hier verwendete mnemonische Code entspricht der Assemblersprache des Mikroprozessors U 880.

## Beispiele:

Maschinencode	Mnemonic
04	INC B
CB FF	SET 7,A
3E 05	LD A,05H
DD 77 03	LD (IX+3),A
21 00 50	LD HL,5000H
DD 21 00 50	LD IX,5000H
DD 36 03 05	LD (IX+3),05
DD CB 03 46	Bit 0 (IX+3)

### 3.2. Syntax der Assemblersprache

Die Assemblersprache des Mikroprozessors U 880 besteht aus leicht erlernbaren mnemonischen Ausdrücken, die zusammen mit den zugehörigen Operanden jeweils einen Maschinenbefehl darstellen.

Ein Anwender Quellprogramm besteht aus einer Folge von Maschine befehlen (Anweisungen), die jeweils eine Programmzeile belegen

#### Aufbau einer Programmzeile:

	Markenfeld	Op.codefeld	Operandenfeld	Kommentarfeld
Beispiel:	A1:	LD	A,5H	; KONSTANTE LADEN

#### Markenfeld:

- muß nicht unbedingt belegt sein
- Marken werden linksbündig eingetragen
- erstes Zeichen muß ein Buchstabe sein
- Marke stellt den Namen des folgenden Befehls dar und dient zur späteren Bezugnahme auf diesen Befehl
- Abschluß der Marke mit Doppelpunkt

- Operationscodefeld: - enthält einen Befehl des Befehlssatzes des U 880  
 - Abschluß durch Tabulator oder Leerzeichen
- Operandenfeld: - enthält einen oder mehrere Operanden  
 - kann auch leer sein (z. B. bei NOP)
- Kommentarfeld: - beginnt stets mit einem Semikolon  
 - beinhaltet bei Bedarf Kommentar zur Anweisung  
 - Abschluß durch Endekennzeichen:  
 NL = NEW LINES (neue Zeile)  
 CR LF = CARRIAGE RETURN LINEFEED  
 (Wagenrücklauf, Zeilenvorschub)  
 Es handelt sich hier um Tastenfunktionen einer Eingabetastatur.

Eine Zeile eines Quellprogramms darf maximal aus 72 Zeichen bestehen. Eine Zeile, beginnend mit einem Semikolon, stellt eine Kommentarzeile im Quellprogramm dar.

### 3.3. Adressierungsarten

Um den Zugriff zu allen Hauptspeicher-, Register- und Ein-/Ausgabeadressen zu ermöglichen, gibt es unterschiedliche Möglichkeiten der Adressierung.

Der Befehlssatz des U 880 umfaßt insgesamt 6 Adressierungsarten.

#### - Direkte Adressierung

Im Befehl ist eine Speicheradresse als Direktwert angegeben

Beispiel: - LD A,(5000H)  
 - IMP 5004H

#### - Direktooperand

Der Operand steht im Befehl direkt hinter dem Operationscode.

Beispiel: - LD A,5	8-Bit-Konstante
- LD HL,5000H	16-Bit-Konstante

- indirekte Adressierung

Die Adresse wird vor Abarbeitung des Befehls in einem Registerpaar bereitgestellt.

Beispiel: - LD HL,5000H  
          JMP (HL)  
          - LD HL,1000H  
          LD A,(HL)

- implizite Adressierung

Op-Code und Adreßinformation sind in einem Befehlswort verschlüsselt. Der Befehl bezieht sich fest auf bestimmte Register. Dazu gehören die arithmetischen Befehle, bei denen das Ergebnis stets im Akkumulator steht.

Beispiel: - LD A,50H  
          ADD H  
          - ADD 50H

- indizierte Adressierung

Zu einem der beiden Indexregister wird ein Datenbyte addiert, wodurch die endgültige Zieladresse entsteht. Dieses Datenbyte kann einen Wert von -128 bis +127 enthalten.

Beispiel: - LD IX,2000H  
          - LD A,(IX+3)

- Relative Adressierung

Der dem Operationscode folgende Operand enthält einen Wert der die Distanz vom aktuellen Befehlszählerstand (PC) zur

Zieladresse ausgibt.

Beispiel: PC = 2000H      M1: JR + 4

### 3.4. Flag-Bit Technik

Das Flagregister (F-Register) ist ein 8-Bit-Register innerhalb der CPU, bei dem die Bits zu Auswertungszwecken einzeln betrachtet werden.

Die meisten Befehle des U 880 beeinflussen im Ergebnis der Operation das Flagregister

	D7	D6	D5	D4	D3	D2	D1	D0
Flagregister	S	Z	X	H	X	P/V	N	CY

S = Vorzeichenbit (Sign-Flag)

Z = Nullbit (Zero-Flag)

X = keine Bedeutung

H = Halbbyteübertragsbit (Half-Carry-Flag)

P/V = Paritäts/Überlauf-Bit (Parity-Overflow-Flag)

N = Additions/Subtraktions-Bit (Add/Subtract-Flag)

CY = Übertrag-Bit (Carry-Flag)

Mit Hilfe bestimmter Befehle können die einzelnen Bits abgefragt werden und auf Grund des Ergebnisses der weitere Programmablauf gestaltet werden.

Von den 6 Bit des Flagregisters, die von Bedeutung sind, können 4 abgefragt werden.

#### SIGN-Flag (S)

Das S-Flag zeigt nach logischen und arithmetischen Operationen mit ganzen Zahlen den Inhalt des höchstwertigen Bits (D7) des Akkumulators an. Ist das Ergebnis negativ (Bit D7 = 1), so steht im S-Flag eine 1 (S-Flag gesetzt), ansonsten steht eine 0 (S-Flag zurückgesetzt).

### ZERO-Flag (Z)

Das Z-Flag wird gesetzt, wenn im Ergebnis einer Operation der Wert Null entstanden ist, ansonsten wird Z rückgesetzt. Bei Bitbefehlen zeigt es an, ob ein Bit gesetzt ist oder nicht. Das Z-Flag wird gesetzt, wenn das Ergebnis von Such- und Vergleichsbefehlen positiv ist. Bei IN/OUT-Befehlen wird Z-Flag gesetzt, wenn bei Datenübertragungen das Zählregister 0 wird oder die am Eingangstor anliegenden Daten den Wert 0 haben.

### HALF-CARRY-Flag (H)

Das H-Flag wird nach arithmetischen Operationen gesetzt. Sobald ein Übertrag von Bit D3 auf Bit D4 auftritt.

Das wird beim Befehl DAA genutzt, um das Ergebnis einer Addition bzw. Subtraktion von zwei gepackten BCD-Zahlen zu korrigieren. Bei der Addition wird das H-Flag beim Übertrag von Bit D3 zu Bit D4 und bei der Subtraktion beim negativen Übergang von Bit D4 auf Bit D3 gesetzt.

### PARITY/OVERFLOW (P/V)

Das P/V-Flag zeigt an:

- Bei logischen und Verschiebepfehlen die Parität des Ergebnisses.

P/V = 1      gerade Anzahl gesetzter Bits

P/V = 0      ungerade Anzahl gesetzter Bits

- Bei arithmetischen Befehlen zeigt P/V-Flag an, ob das Ergebnis einer Operation mit zwei vorzeichenbehafteten ganzen Zahlen außerhalb des zulässigen Zahlenbereiches liegt, d. h. größer als +127 bzw. kleiner als -128 ist.

### ADD/SUBTRACT-Flag (N)

Das N-Flag zeigt an, ob als letzter Befehl eine Addition (N-Flag = 0) oder Subtraktion (N-Flag = 1) erfolgte. Das N-Flag

wird vom DAA-Befehl ausgewertet.

### CARRY-FLAG (C)

Entsteht bei der Addition ein Übertrag (bzw. bei der Subtraktion ein negativer Übertrag), so wird das C -Flag gesetzt.

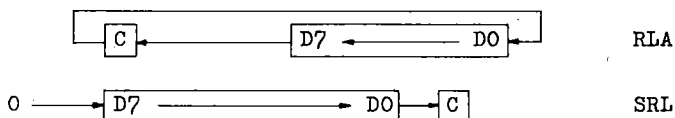
Bei der Addition bzw. Subtraktion von Operanden, bestehend aus mehreren Bytes, wird beim Übertrag vom höchstwertigen Bit (D7) des niederwertigsten Bytes zum niederwertigsten Bit (D0) des höherwertigen Datenbytes das C -Flag gesetzt.

Beispiele:

2. Byte	C -Flag	1. Byte			
00000010		01101011		619	(02 6BH)
00000100		11001100	+	1228	(04 CCH)
1			<hr/>		
00000111		00110111		1847	(07 37H)
00000101		00110101		1333	(05 35H)
00000010		10001100	-	652	(02 8CH)
1			<hr/>		
00000010		10101001		681	(02 A9H)

Außerdem wird das C -Flag von den Verschiebe- und Rotationsbefehlen beeinflusst. Es kann als Ziel und/oder Quelle des höchstwertigsten bzw. niederwertigsten Bits dienen.

Beispiele



Außerdem gibt es noch 2 Flagbits, die nicht abgefragt werden können. Sie haben für die BCD-Arithmetik Bedeutung.

### 3.5. Befehlssatz

Der Befehlssatz des U 880 umfaßt 158 verschiedene Befehle in 696 verschiedenen Modifikationen. Die in diesem Abschnitt enthaltenen Erläuterungen zu den einzelnen Befehlen bzw. Befehlsgruppen sollen dem Nutzer die Wirkungsweise veranschaulichen. Spezielle Angaben zu den einzelnen Befehlen, z. B. Operationscode, Beeinflussung der Flagbits, sind der dem LC 80 beigefügten Befehlskarte zu entnehmen.

Um die Wirkungsweise der Befehle am LC 80 zu erproben und somit die Programmierung zu erlernen, sind besonders die Punkte 2., 3.1., 3.2. und 3.3. der Bedienungsanleitung gründlich durchzuarbeiten und die nachstehenden Hinweise zu beachten.

Der LC 80 wird zunächst auf Stepfunktion vorbereitet (siehe Punkt 3.3.2. Bedienungsanleitung). Danach wird das Programm in den Speicher (RAM) geschrieben (siehe Punkt 3.2. Bedienungsanleitung) und nochmals auf Richtigkeit überprüft. Durch das Betätigen der Taste "NMI" wird der LC 80 auf Registeranzeige und Stepfunktion gestellt. Es erscheint die Adresse der nächsten Speicherzelle und die Buchstaben PC (Programmcounter). Der PC wird auf die Startadresse des Beispielprogramms gesetzt (2000H) Über die "+"-Taste werden die Register aufgerufen und FFFF (Reg. F auf 00) gesetzt. Es ist zu beachten, daß der Stackpoint SP seinen ursprünglichen Inhalt behält. Danach kann in der Stellung PC das Programm schrittweise abgearbeitet werden, nach jedem Schritt kann die Änderung des Registers beobachtet und an Hand der Übersicht zu den Beispielen kontrolliert werden. Gegebenenfalls muß während der Abarbeitung des Programms zur besseren Verständlichkeit das Register F auf 00 gesetzt werden. Die Vorbereitung des LC 80 auf Stepfunktion und Registeranzeige ist nur nach Power-On-Reset notwendig. Der in der Tabelle des Programmbeispiels angegebene Registerinhalt ist nur für die durch den abgearbeiteten Befehl beeinflusste Register angegeben.



Bedeutung der in der Befehlsbeschreibung verwendeten Abkürzungen:

r :	eines der Register	A, B, C, D, E, H, L
r <sub>1</sub> :	eines der Register	A, B, C, D, E, H, L
r <sub>2</sub> :	eines der Register	A, B, C, D, E, H, L
n :	Direktooperand	
nn:	16-Bit-Konstante	
dd:	eines der Registerpaare	BC, DE, HL, SP
qq:	eines der Registerpaare	BC, DE, HL, AF
s :	- eines der Register r:	A, B, C, D, E, H, L
	- Direktooperand n	
	- Speicherinhalt der durch HL festgelegten Adresse	=(HL)=M
	- " " " IX+d "	=(IX+d)
	- " " " IY+d "	=(IY+d)

### 3.5.1. Ladebefehle

#### 3.5.1.1. 8-Bit-Ladebefehle

Sie bewirken die Übertragung eines Bitmusters zwischen zwei Registern, einem Register und einem Speicherplatz oder eines im Programm enthaltenen Festwertes (Direktooperand) auf ein Register.

	Byte	Beschreibung
LD r <sub>1</sub> ,r <sub>2</sub>	1	Laden Register r <sub>1</sub> mit Register r <sub>2</sub>
LD r,(HL)	1	Laden Register r mit Inhalt des durch Registerpaar HL adressierten Speicherplatzes
LD r,n	2	Laden Register r mit einem Direktooperanden n
LD r,(IX+d)	3	Laden Register r mit Inhalt des durch Register IX + Verschiebung d adressierten Speicherplatzes
LD r,(IY+d)	3	Laden Register r mit Inhalt des durch IY + Verschiebung d adressierten Speicherplatzes

	Byte	Beschreibung
LD (HL),r	1	Laden des durch HL adressierten Speicherplatzes mit Inhalt von Register r
LD (HL),n	2	Laden des durch HL adressierten Speicherplatzes mit einem Direktoperanden n
LD (IX+d),r	3	Laden des durch IX + Verschiebung d adressierten Speicherplatzes mit Inhalt des Registers r
LD (IY+d),r	3	Laden des durch IY + Verschiebung d adressierten Speicherplatzes mit Inhalt des Registers r
LD (IX+d),n	4	Laden des durch IX + Verschiebung d adressierten Speicherplatzes mit Direktoperand n
LD (IY+d),n	4	Laden des durch IY + Verschiebung d adressierten Speicherplatzes mit Direktoperand n
LD A,(BC)	1	Laden des Akkumulators (Register A) mit Inhalt des durch Doppelregister BC adressierten Speicherplatzes
LD A,(DE)	1	Laden des Akkumulators mit Inhalt des durch Doppelregister DE adressierten Speicherplatzes
LD A,(nn)	3	Laden des Akkumulators mit Inhalt des durch nn adressierten Speicherplatzes nn : 16-Bit-Konstante
LD (BC),A	1	Laden des durch BC adressierten Speicherplatzes mit Inhalt des Akkumulators
LD (DE),A	1	Laden des durch DE adressierten Speicherplatzes mit Inhalt des Akkumulators
LD I,A	2	Laden des Interruptregisters I mit Akkumulatorinhalt
LD R,A	2	Laden des Refresh-Registers R mit Akkumulatorinhalt

	Byte	Beschreibung
LD A,I	2	Laden des Akkumulators mit Inhalt aus Interruptvektor-Register I
LD A,R	2	Laden des Akkumulators mit Inhalt aus Refresh-Register R

Musterbeispiel: Laden von Registern

### 1. Vorbereiten des LC 80 auf Stepfunktion

Beachte Hinweis auf Seite 4 der Bedienungsanleitung!

Adreßbelegung wie unter Punkt 3.3. der Bedienungsanleitung beschrieben

Adresse	Daten	Tasteneingabe
		ADR
		2340
		DAT
2340H	C3H	C3
		+
2341H	90H	90
		+
2342H	0BH	0B

### 2. Eingabe des Programmes

Adresse	Befehlscode	Mnemonic	Tasteneingabe
			ADR
			2000
			DAT
2000	3E 50	LD A,50H	3E
			+
			50
			+
2002	06 20	LD B,20H	.
2004	60	LD H,B	.
2005	6F	LD L,A	.
2006	5E	LD E,(HL)	+
2007	76	HALT	76
			ADR
			2050
			DAT
2050	AA	AAH	AA

### 3. Übergang zum Stepbetrieb

- . Angabe der Startadresse des Programmes
- . Übergang zum Stepbetrieb
- . Anzeige:

ADR

2000

NMI

2002 PC

1. Befehl des Programmes  
wurde bereits abgearbeitet

### 4. Register auf Anfangswert setzen

		Tasten- eingabe	Anzeige
. PC wird auf 2000H gesetzt, um vor Programmbeginn Anfangs- werte bereitstellen zu können		2000 <input type="checkbox"/> EX	
. Register werden wie folgt belegt:			
AF - FFOOH	<input type="checkbox"/> +	FFOO	XXXXAF FROQAF
		<input type="checkbox"/> EX	FFOOAF
BC - FFFFH	<input type="checkbox"/> +	FFFF	XXXXBC FFFFBC
		<input type="checkbox"/> EX	FFFFBC
DE - FFFFH	<input type="checkbox"/> +		XXXXDE
HL - FFFFH	.		
A'F' - FFFFH	.		
B'C' - FFFFH	.		
D'E' - FFFFH	.		
H'L' - FFFFH	.		
IX - FFFFH	.		
IY - FFFFH	<input type="checkbox"/> +	FFFF	XXXXIY FFFFIY
		<input type="checkbox"/> EX	FFFFIY
	<input type="checkbox"/> +		XXXXSP
	<input type="checkbox"/> +		2000PC

## 5. Programmabarbeitung

Durch Drücken der Taste **ADR** können die Befehle des Programmes einzeln nacheinander abgearbeitet werden.

Nach jedem Befehl kann der Inhalt der Register abgefragt (durch

Drücken der Tasten **+** bzw. **-**

oder geändert werden. Wird nach

Drücken von **+** bzw. **-** das entsprechende Register angezeigt, kann der neue Wert auf der Tastatur eingedrückt und durch Betätigen der Taste

**EX** dem Register fest zugewiesen werden.

<b>ADR</b>	2002PC
<b>+</b>	5000AF
<b>-</b>	2002PC
<b>ADR</b>	2004PC
<b>+</b>	5000AF
<b>+</b>	20FFBC

Reg.Inhalt	Register	PC
5000	AF	2002
20FF	BC	2004
20FF	HL	2005
2050	HL	2006
FFAA	DE	2007

### 3.5.1.2. 16-Bit-Ladebefehle

Sie bewirken die Übertragung eines Bitmusters zwischen einem Doppelregister und 2 aufeinanderfolgenden Speicherplätzen oder eines 16-Bit-Festwertes.

	Byte	Beschreibung
LD dd,nn	3	Laden Registerpaar dd mit 16-Bit-Konstante nn
LD IX,nn	4	Laden des Registers IX mit 16-Bit-Konstante nn
LD IY,nn	4	Laden des Registers IY mit 16-Bit-Konstante nn
LD HL,(nn)	3	Laden des Registerpaares HL mit Inhalt der durch nn und nn+1 adressierten Speicherplätze H : = (nn +1) L : = (nn)

	Byte	Beschreibung
LD dd, (nn)	4	Laden Registerpaar dd mit Inhalt der durch nn und nn+1 adressierten Speicherplätze $d_H$ (höherwertiger Teil) : = (nn+1) $d_L$ (niederwertiger Teil) : = (nn)
LD IX, (nn)	4	Laden Register IX mit Inhalt der durch nn und nn+1 adressierten Speicherplätze $IX_H$ : = (nn+1) $IX_L$ : = (nn)
LD IY, (nn)	4	Laden Register IY mit Inhalt der durch nn und nn+1 adressierten Speicherplätze $IY_H$ : = (nn+1) $IY_L$ : = (nn)
LD (nn), HL	3	Laden des durch nn adressierten Speicherplatzes mit Register HL (nn+1) : = H (nn) : = L
LD (nn), dd	4	Laden des durch nn adressierten Speicherplatzes mit Registerpaar dd (nn+1) : = $dd_H$ (nn) : = $dd_L$
LD (nn), IX	4	Laden des durch nn adressierten Speicherplatzes mit IX (nn+1) : = $IX_H$ (nn) : = $IX_L$
LD (nn), IY	4	Laden des durch nn adressierten Speicherplatzes mit Register IY (nn+1) : = $IY_H$ (nn) : = $IY_L$
LD SP, HL	1	Laden des Stackpointers mit Registerpaar HL $SP_H$ : = H $SP_L$ : = L

	Byte	Beschreibung
LD SP,IX	2	Laden des Stackpointers mit Register IX $SP_H : = IX_H$ $SP_L : = IX_L$
Ld SP,IY	2	Laden des Stackpointers mit Register IY $SP_H : = IY_H$ $SP_L : = IY_L$
PUSH qq	1	Laden des Inhaltes des Registerpaares qq auf den durch den Stackpointer adres- sierten Speicherplatz $(SP-2) : = qq_L$ $(SP-1) : = qq_H$ $SP : = SP-2$
PUSH IX	2	Laden des Inhaltes des Registers IX auf den durch den Stackpointer adressierten Speicherplatz $(SP-2) : = IX_L$ $(SP-1) : = IX_H$ $SP : = SP-2$
PUSH IY	2	Laden des Inhaltes des Registers IY auf den durch den Stackpointer adressierten Speicherplatz $(SP-2) : = IY_L$ $(SP-1) : = IY_H$ $SP : = SP-2$
POP qq	1	Laden des Registers qq mit dem Inhalt des durch den Stackpointer adressierten Speicherplatzes $qq_H : = (SP+1)$ $qq_L : = (SP)$ $SP : = SP+2$
POP IX	2	Laden des Registers IX mit dem Inhalt des vom Stackpointer SP adressierten Speicherplatzes

	Byte	Beschreibung
POP IY	2	$IX_H := (SP+1)$ $IX_L := (SP)$ $SP := SP+2$ Laden des Registers IY mit dem Inhalt des vom Stackpointer SP adressierte Speicherplatzes $IY_H := (SP+1)$ $IY_L := (SP)$ $SP := SP+2$

Beispiele: - Laden des Speicherplatzes 2100H mit dem Wert 30H

Adresse	Befehlscode	Mnemonic
2000	21 00 21	LD HL,2100H
2003	3E 30	LD A,30H
2005	77	LD (HL),A
2006	76	HALT

Reg.Inhalt	Register	PC
2100	HL	2003
30FF	AF	2005



- Laden von Doppelregistern

Adresse	Befehlscode	Mnemonic
2000	01 50 20	LD BC,2050H
2003	DD 21 55 20	LD IX,2055H
2007	2A 60 20	LD HL,(2060H)
200A	22 65 20	LD (2065H),HL
200D	76	HALT
2060	AA	AAH
2061	BB	BHH

Reg.Inhalt	Register	PC
2050	BC	2003
2055	IX	2007
BBAA	HL	200A

Die Taste RES wird betätigt und über ADR und DAT die Speicherplätze

2065     AA            und  
2066     BB

abgefragt.

### 3.5.2. Registertauschbefehle

Sie bewirken den Austausch des Bitmusters zwischen zwei oder mehreren Doppelregister.

	Byte	Beschreibung
<b>EX DE,HL</b>	1	Austausch der Register DE,HL D $\leftrightarrow$ H E $\leftrightarrow$ L
<b>EXAF</b>	1	Austausch von Registersatz AF mit AF' A $\leftrightarrow$ A' F $\leftrightarrow$ F'
<b>EXX</b>	1	Austausch der Registerpaare BC mit BC' DE mit DE' und HL mit HL'
<b>EX (SP),HL</b>	1	Austausch des Registers HL mit Inhalt des Stackpointers SP H $\leftrightarrow$ (SP+1) L $\leftrightarrow$ (SP)
<b>EX (SP),IX</b>	2	Austausch des Registers IX mit Inhalt des Stackpointers SP IX <sub>H</sub> $\leftrightarrow$ (SP+1) IX <sub>L</sub> $\leftrightarrow$ (SP)
<b>EX (SP),IY</b>	2	Austausch des Registers IY mit Inhalt des Stackpointers SP IY <sub>H</sub> $\leftrightarrow$ (SP+1) IY <sub>L</sub> $\leftrightarrow$ (SP)

Beispiele: Austausch Hauptregistersatz - Zweitregistersatz

Adresse	Befehlscode	Mnemonic
2000	3E 20	LD A,20H
2002	01 50 20	LD BC,2050H
2005	11 DD CC	LD DE,C0DDH
2008	21 BB AA	LD HL,AABBH
200B	08	EX AF
200C	D9	EXX
200D	76	HALT

Reg.Inhalt	Register	PC
20FF	AF	2002
2050	BC	2005
CCDD	DE	2008
AABB	HL	200B
FFFF	AF	200C
20FF	AF'	
FFFF	BC	200D
FFFF	DE	
FFFF	HL	
2050	BC'	
CCDD	DE'	
AABB	HL'	

### 3.5.3. Blocktransfer- und Suchbefehle

Die Blocktransferbefehle bewirken die Übertragung einer Datenfolge von einem Speicherbereich in einen anderen.

Die Suchbefehle vergleichen immer den Akkumulatorinhalt mit den Datenbytes in einem festgelegten Speicherbereich.

	Byte	Beschreibung
LDI	2	Transport eines Datenbytes von der durch Registerpaar HL adressierten Speicherzelle nach der durch Registerpaar DE adressierten Speicherzelle. Danach werden die Register HL und DE um 1 erhöht und Register BC um 1 vermindert (BC = Bytezähler). (DE) : = (HL) DE : = DE+1 HL : = HL+1 BC : = BC-1
LDIR	2	Transport von mehreren Datenbytes ab der durch HL adressierten Speicherzelle nach der durch DE adressierten Speicherzelle. Die Anzahl der Bytes enthält das Registerpaar BC. Danach werden die Register HL und DE um 1 erhöht und Register BC um 1 vermindert bis BC = 0. (DE) : = (HL) HL : = HL+1 DE : = DE+1 BC : = BC-1
LDD	2	Transport eines Datenbytes von der durch Registerpaar HL adressierten Speicherzelle nach der durch Registerpaar DE adressierten Speicherzelle. Danach werden die Register HL, DE und BC um 1 vermindert.

	Byte	Beschreibung
		(DE) : = (HL) DE : = DE-1 HL : = HL-1 BC : = BC-1
LDDR	2	Transport mehrerer Datenbytes ab der durch HL adressierten Speicherzelle nach der durch DE adressierten Speicherzelle. Anzahl der Bytes steht im Registrierpaar BC. Danach werden die Register HL, DE und BC um 1 vermindert, bis BC = 0. (DE) : = (HL) DE : = DE-1 HL : = HL-1 BC : = BC-1
CPI	2	Vergleich des Akkumulatorinhaltes mit dem Inhalt der durch HL adressierten Speicherzelle. In Abhängigkeit vom Ergebnis werden die Flags gesetzt. BC wird um 1 vermindert und HL um 1 erhöht. A = (HL) HL : = HL+1 BC : = BC-1
CPIR	2	Vergleich des Akkumulatorinhaltes mit dem Inhalt der durch HL adressierten Speicherzelle. BC wird um 1 vermindert, HL um 1 erhöht. Anzahl der zu vergleichenden Bytes steht im BC. Vergleich ist beendet, wenn BC = 0 oder der Akkumulator = (HL) ist. A = (HL) HL : = HL+1 BC : = BC-1 Flags : Z = 1 bei Gleichheit P/V = 1 bei BC-1 = 0

	Byte	Beschreibung
CPD	2	Vergleich des Akkumulatorinhaltes mit dem Inhalt der durch HL adressierten Speicherzelle. Entsprechend dem Ergebnis werden die Flags gesetzt. Die Registerpaare BC und HL werden um 1 vermindert. A = (HL) HL : = HL-1 BC : = BC-1
CPDR	2	Vergleich des Akkumulatorinhaltes mit der durch HL adressierten Speicherzelle. BC und HL werden um 1 vermindert. Anzahl der zu vergleichenden Bytes wird durch BC festgelegt. Der Vergleich endet, wenn BC = 0 oder der Akkumulator = (HL) ist. A = (HL) HL : = HL-1 BC : = BC-1 Flags : Z = 1 bei Gleichheit P/V = 1 bei BC-1 = 0

Beispiele:

- Transport von Daten von Adresse 2050H - 2055H nach Adresse 2100H - 2105H

Adresse	Befehlscode	Mnemonic
2000	21 50 20	LD HL,2050H
2003	01 06 00	LD BC,0006H
2006	11 00 21	LD DE,2100H
2009	ED B0	LDIR
200B	76	HALT
2050	AA	
2051	BB	
2052	CC	
2053	DD	
2054	EE	
2055	99	

Reg. Inhalt	Register	PC
2050	HL	2003
0006	BC	2006
2100	DE	2009
2051	HL	
0005	BC	
2101	DE	
	.	.
	.	.
	.	.
		6 x <span style="border: 1px solid black; padding: 2px;">ADR</span>
0000	BC	
2106	DE	
2056	HL	

Auf den Adressen 2100 bis 2105 sind die Daten AA BB CC DD EE 99 eingeschrieben.

- Suchen der im Register A angegebenen Daten im Speicher

Adresse	Befehlscode	Mnemonic
2000	3E A9	LD A,0A9H
2002	01 06 00	LD BC,0006H
2005	21 50 20	LD HL,2050H
2008	ED B1	CPIR
200A	7D	LD A,L
200B	C6 05	ADD 05
200D	6F	LD L,A
200E	7E	LD A,(HL)
200F	76	HALT
2050	AA	
2051	BB	
2052	CC	
2053	DD	
2054	EE	
2055	A9	
2056	01	
2057	02	
2058	03	
2059	04	
205A	05	
205B	06	

Statt 0A9H kann als Suchobjekt einer der anderen Speicherinhalte auf den Adressen 2050 bis 2054 verwendet werden. Man erhält dann im Ergebnis des obigen Programmes im Register A die Inhalte der Adressen 2056 bis 2059.



Reg.Inhalt	Register	PC
A900	AF	2002
0006	BC	2005
2050	HL	2008
0005	BC	2008
2051	HL	
.	.	
.	.	
.	.	
0000	BC	200A
2056	HL	
5642	AF	200B
5B08	AF	200D
205B	HL	200E
0608	AF	200F

#### 3.5.4. Arithmetikbefehle

Als arithmetische Operationen stehen im Mikroprozessor nur Addition, Subtraktion, Inkrementierung (Erhöhen um 1) und Dekrementierung (Erniedrigen um 1) zur Verfügung.

##### 3.5.4.1. 8-Bit-Arithmetikbefehle

Die 8-Bit-Arithmetikbefehle beeinflussen im Ergebnis der Operation die entsprechenden Flags.

Die Operationen finden stets zwischen dem Akkumulator und einem zweiten Datenbyte statt.

Dieses zweite Datenbyte kann ein Direktoperand sein, oder es steht in einem Register oder wird durch dieses adressiert.

Das Ergebnis steht immer im Akkumulator. Der zweite Operand bleibt unverändert, außer bei den Inkrement- und Dekrementbefehlen.

	Byte	Beschreibung
ADD r	1	Addiere Inhalt Register r zum Akkumulatorinhalt
ADD M	1	Addiere Inhalt der durch HL adressierten Speicherzelle zum Akkumulatorinhalt
ADD n	2	Addiere Direktoperand n zum Akkumulatorinhalt
ADD (IX+d)	3	Addiere die durch Register IX + Verschiebung d adressierte Speicherzelle zum Akkumulatorinhalt (-128 = d = 127)
ADD (IY+d)	3	Addiere die durch Register IY + Verschiebung d adressierte Speicherzelle zum Akkumulatorinhalt (-128 = d = 127)
ADC r	1	Addiere zum Akkumulatorinhalt den Inhalt von Register r + Carry-Flag
ADC M	1	Addiere zum Akkumulatorinhalt Inhalt der durch Registerpaar HL adressierten Speicherzelle + Carry-Flag
ADC n	2	Addiere zum Akkumulatorinhalt Direktoperand n + Carry-Flag
ADC (IX+d)	3	Addiere zum Akkumulatorinhalt die durch Register IX + Verschiebung d adressierte Speicherzelle + Carry-Flag
ADC (IY+d)	3	Addiere zum Akkumulatorinhalt den Inhalt der durch Register IY + Verschiebung d adressierte Speicherzelle + Carry-Flag
SUB s	1-3	Subtrahiere s vom Inhalt des Akkumulators s kann sein: r: A, B, C, D, E, H, L n: Direktoperand (IX+d) (IY+d), wie bei ADD-Befehl
SBC s	1-3	Subtrahiere vom Inhalt des Akkumulators s und Carry-Flag

	Byte	Beschreibung
INC r	1	Erhöhung des Registerinhaltes um 1
INC (HL)	1	Erhöhung des Inhaltes des Registerpaares HL um 1
INC (IX+d)	3	Erhöhung des Inhaltes der durch IX + Verschiebung d adressierten Speicherzelle (-128 = d = 127)
INC (IY+d)	3	Erhöhung des Inhaltes der durch IY + Verschiebung d adressierten Speicherzelle (-128 = d = 127)
DEC f	1 o. 3	Inhalt von f wird um 1 vermindert. f kann sein: r: A, B, C, D, E, H, L M: (HL) (IX+d) (IY+d) wie bei INC-Befehl

### 3.5.4.2. 16-Bit-Arithmetikbefehle

Die 16-Bit-Arithmetikbefehle für Addition und Subtraktion beeinflussen die Flags.

Bei den INC- und DEC-Befehlen der Doppelregister werden keine Flags gesetzt.

Die Operationen finden immer mit dem Inhalt von zwei Doppelregistern statt.

	Byte	Beschreibung
ADD HL,dd	1	Addition von Registerpaar dd zu Registerpaar HL
ADC HL,dd	2	Addition von Registerpaar dd und Carry-Flag zu Registerpaar HL
SBC HL,dd	2	Subtraktion von Registerpaar dd und Carry-Flag von Registerpaar HL

	Byte	Beschreibung
ADD IX,pp	2	Addition von Registerpaar pp zu Register IX
ADD IY,pp	2	Addition von Registerpaar pp zu Register IY
INC dd	1 (2)	Addition von 1 zum Inhalt des Doppelregisters dd: BC, DE, HL, SP, IX, IY
DEC dd	1 (2)	Subtraktion von 1 vom Inhalt des Doppelregisters dd: BC, DE, HL, SP, IX, IY

Beispiel: Addition der Doppelregister HL und BC, danach Subtraktion von 1

Adresse	Befehlscode	Mnemonic
2000	01 10 10	LD BC,1010H
2003	21 00 00	LD HL,0000H
2006	09	ADD HL,BC
2007	2B	DEC HL
2008	76	HALT

Reg.Inh.	Register	F-Reg. SZXHP/VNC	PC
1010	BC		2003
0000	HL		2006
1010	HL		2007
100F	HL		2008

### 3.5.5. Sprungbefehle

Ein Sprungbefehl im Programm bewirkt, daß die normale Reihenfolge der Abarbeitung unterbrochen und statt beim nächsten bei einem anderen Befehl fortgesetzt wird. Zur Kennzeichnung dieses Befehls wird dessen Adresse als Sprungziel angegeben.

	Byte	Beschreibung
JMP nn	3	Unbedingter Sprung nach Adresse nn. Das Programm wird mit dem Befehl fortgesetzt, der auf Adresse nn steht.
JPNZ nn	3	Sprung nach Adresse nn, wenn Z-Flag = 0
JPZ nn	3	Sprung nach Adresse nn, wenn Z-Flag = 1
JFNC nn	3	Sprung nach Adresse nn, wenn C -Flag = 0
JPC nn	3	Sprung nach Adresse nn, wenn C -Flag = 1
JPPO nn	3	Sprung nach Adresse nn, wenn P/V-Flag = 0
JPPE nn	3	Sprung nach Adresse nn, wenn P/V-Flag = 1
JPP nn	3	Sprung nach Adresse nn, wenn S-Flag = 0
JPM nn	3	Sprung nach Adresse nn, wenn S-Flag = 1
JR e	2	Unbedingter relativer Sprung e: Abstand zwischen aktuellen Befehlszählerstand und Zieladresse (-126 = e = 129)
JRNZ e	2	Bei Z-Flag = 0 relativer Sprung mit Abstand e (-126 = e = 129) Z-Flag = 1, kein Sprung
JRZ e	2	Bei Z-Flag = 1 relativer Sprung mit Abstand e (-126 = e = 129) Z-Flag = 0, kein Sprung
JRNC e	2	bei C -Flag = 0 relativer Sprung mit Abstand e (-126 = e = 129) C -Flag = 1, kein Sprung

	Byte	Beschreibung
JRC e	2	Bei C -Flag = 1 relativer Sprung mit Abstand e (-126 = e = 129) C -Flag = 0, kein Sprung
JMP M	1	Unbedingter Sprung zu der im Registerpaar HL angegebenen Adresse
JMP (IX)	2	Unbedingter Sprung zu der in IX angegebenen Adresse
JMP (IY)	2	Unbedingter Sprung zu der in IY angegebenen Adresse
DJNZ e	2	Subtraktion 1 von Register B, - ist B ≠ 0 Sprung zur Adresse, die sich aus Abstand e ergibt - ansonsten Abarbeitung beim nächsten Befehl

### 3.5.6. 1-Byte-Logik-Befehle

Sie realisieren logische Verknüpfungen vom Akkumulatorinhalt mit einem Register bzw. mit dem Inhalt einer Adresse. Man verwendet diese Befehle im Programm, um Negationen auszuführen, Masken zu setzen, einzelne Bits zu prüfen, zu setzen oder rückzusetzen usw.:

	Byte	Beschreibung
AND s	1-3	Der Inhalt von s wird konjunktiv (log. UND) mit dem Inhalt des Akkumulators A verknüpft Abarbeitung nach folgender Tabelle:

Akku \ Operand	0	1
0	0	0
1	0	1

	Byte	Beschreibung																				
OR s	1-3	<p>Der Inhalt von s wird disjunktiv (log. ODER) mit dem Inhalt des Akkumulators A verknüpft</p> <p>Abarbeitung nach folgender Tabelle:</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="border: none;"></td> <td style="border: none;">Akku</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;">0</td> <td style="border: none;">1</td> </tr> <tr> <td style="border: none;">Operand</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">0</td> <td style="border: none;">0</td> <td style="border: none;">1</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">1</td> <td style="border: none;">1</td> <td style="border: none;">1</td> </tr> </table>		Akku					0	1	Operand					0	0	1		1	1	1
	Akku																					
		0	1																			
Operand																						
	0	0	1																			
	1	1	1																			
XOR s	1-3	<p>EXKLUSIVES ODER von s und Akkumulator</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="border: none;"></td> <td style="border: none;">Akku</td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;">0</td> <td style="border: none;">1</td> </tr> <tr> <td style="border: none;">Operand</td> <td style="border: none;"></td> <td style="border: none;"></td> <td style="border: none;"></td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">0</td> <td style="border: none;">0</td> <td style="border: none;">1</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none;">1</td> <td style="border: none;">1</td> <td style="border: none;">0</td> </tr> </table>		Akku					0	1	Operand					0	0	1		1	1	0
	Akku																					
		0	1																			
Operand																						
	0	0	1																			
	1	1	0																			
CMP s	1-3	<p>Vergleich von s mit dem Akkumulator. Inhalte von s und Akkumulator bleiben erhalten. In Abhängigkeit vom Ergebnis der Operation werden die Flags wie folgt gesetzt:</p> <p>S: 1, falls Ergebnis negativ, sonst 0</p> <p>Z: 1, falls Ergebnis 0, sonst 0</p> <p>H: 0, falls von Bit 4 "geborgt" wurde, sonst 1</p> <p>P/V: 1, falls Überlauf, sonst 0</p> <p>N: 1 gesetzt</p> <p>d: 1, falls geborgt wurde, sonst 0</p>																				

Beispiel:

(Der Inhalt des Registers F ist zu Beginn Q zu setzen)

Adresse	Befehlscode	Mnemonic
2009	3E FF	LD A,FFH
200B	0E 02	LD C,02H
200D	A9	XOR C
200E	B9	CMP C
200F	0C	INC C
2010	76	HALT

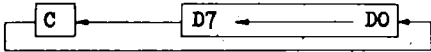
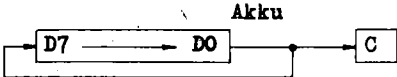
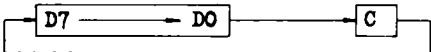
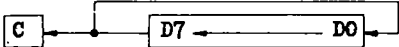
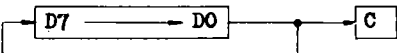
Reg.Inhalt	Register	F-Register	PC
		S Z X H XP/VN C	
			F : = C
FF 00	AF		200B
FF 02	BC		200D
FD A8	AF	1 0 1 0 1 0 0 0	200E
FD 82	AF	1 0 0 0 0 0 1 0	200F
FD 00	AF		2010
FF 03	BC		

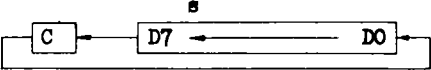
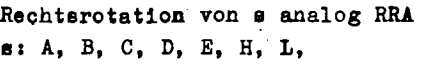
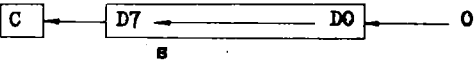
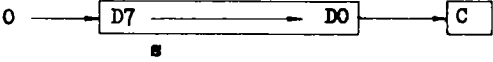
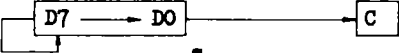
### 3.5.7. Rotations- und Schiebepfehle

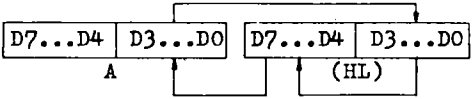
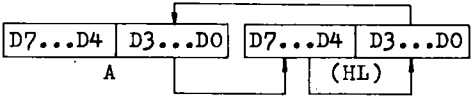
Diese Befehle beziehen sich auf ein Register r und/oder auf eine adressierte Speicherzelle. Sie verschieben den Inhalt nach rechts oder links.

Byte	Beschreibung
RLCA	<p>1</p> <p>Linksrotation des Akkumulatorinhalts um eine Bitposition. Der Akkumulatorinhalt wird um eine Bitposition nach links verschoben. Bit D7 wird zum Inhalt von Bit D0 und des Carry-Flags.</p> <p style="text-align: center;">Akku</p>



	Byte	Beschreibung
RLA	1	<p>Linksrotation des Akkumulatorinhalts um eine Bitposition. Bit D7 wird ins C - Flag geschoben, C -Flag ersetzt D0.</p> <p>Akku</p> 
RRCA	1	<p>Rechtsrotation des Akkumulatorinhalts um eine Bitposition. Bit D0 ersetzt D7 und C -Flag.</p> <p>Akku</p> 
RRA	1	<p>Rechtsrotation des Akkumulatorinhalts um eine Bitposition. Bit D0 ersetzt C -Flag, C -Flag ersetzt Bit D7.</p> <p>Akku</p> 
RLC r	2	
RLC (HL)	2	
RLC (IX+d)	4	Linksrotation um eine Bitposition. C := D7, DO := D7
RLC (IY+d)	4	
		 <p>r, (HL), (IX+d) (IY+d)</p>
RRC s	2 o. 4	<p>Rechtsrotation von s analog RRCA</p> <p>s: A, B, C, E, H, L, (HL), (IX+d), (IY+d)</p> <p>s</p> 

	Byte	Beschreibung
RL s	2 o. 4	<p>Linksrotation von s analog RLA  s: A, B, C, D, E, H, L,  (HL), (IX+d), (IY+d)</p> 
RR s	2 o. 4	<p>Rechtsrotation von s analog RRA  s: A, B, C, D, E, H, L,  (HL), (IX+d), (IY+d)</p> 
SLA s	2 o. 4	<p>Linksverschiebung von s um 1 Bit.  Bit D0 := 0,  C-Flag := D7  s: A, B, C, D, E, H, L,  (HL), (IX+d), (IY+d)</p> 
SRL s	2 o. 4	<p>Rechtsverschiebung von s um 1 Bit.  Bit D7 := 0, C := D0</p>  <p>s: A, B, C, D, E, H, L,  (HL), (IX+d), (IY+d)</p>
SRA s	2 o. 4	<p>Rechtsverschiebung von s um 1 Bit.  Bit D7 bleibt erhalten. C := D0</p> 

	Byte	Beschreibung
RLD	2	<p>Linksverschiebung zwischen Akkumulator und dem Inhalt des durch Registerpaar HL adressierten Speicherplatzes.</p> <p>Die unteren 4 Bits von (HL) werden in die oberen 4 Bits von (HL) und diese wiederum in die unteren 4 Bits des Akkumulators übertragen. Gleichzeitig erfolgt eine Übertragung der unteren 4 Bits des Akkumulators in die unteren 4 Bits von (HL).</p> 
RRD	2	<p>Rechtsverschiebung zwischen Akkumulator und dem Inhalt der durch Registerpaar HL adressierten Speicherzelle.</p> <p>Die unteren 4 Bits von (HL) werden in die unteren 4 Bits des Akkumulators und diese wiederum in die oberen 4 Bits von (HL) übertragen.</p> <p>Gleichzeitig erfolgt eine Übertragung der oberen 4 Bits von (HL) in die unteren 4 Bits von (HL).</p> 

Beispiele:

- Adresse	Befehlscode	Mnemonic
2000	3E FE	LD A,0FEH
2002	07	RLCA
2003	38 FD	JRC FD
2005	3E 01	LD A,01H
2007	07	RLCA
2008	D2 07 20	JPNC 2007
200B	3E FF	LD A,0FFH
200D	0E FF	LD C,0FFH
200F	C3 14 20	JMP 2014
2012	0E FB	LD C,0FBH
2014	B9	CMP C
2015	CA 12 20	JPZ 2012
2018	3E 01	LD A,01H
201A	06 03	LD B,03H
201C	87	ADD A
201D	10 FD	DJNZ FD
201F	76	HALT

Reg.-Inhalt	Register	A-Register	F-Register								PC
			S	Z	X	H	XP/VN	C			
FE 00	AF	1111 1110	0	0	0	0	0	0	0	0	2002
FD 29	AF	1111 1101	0	0	1	0	1	0	0	1	2003
											2002
FB 29	AF	1111 1011	0	0	1	0	1	0	0	1	2003
											.
											. 6x
											.
FE 28	AF	1111 1110	0	0	1	0	1	0	0	0	2005
											F:=00
01 00	AF	0000 0001	0	0	0	0	0	0	0	0	2007
02 00	AF	0000 0010	0	0	0	0	0	0	0	0	2008
											2007
04 00	AF	0000 0100	0	0	0	0	0	0	0	0	2008
											.
											. 6x
											.
01 01	AF	0000 0001	0	0	0	0	0	0	0	1	200B
											F:=00

Reg.-Inhalt	Register	A-Register	F-Register								PC
			S	Z	X	H	XP	VN	C		
FF 00	AF										200D
FF FF	BC										200F 2014
FF 6A	AF		0	1	1	0	1	0	1	0	2015 2012 2014
FF 6A	AF										2014
FF FB	BC										
FF 2A	AF		0	0	1	0	1	0	1	0	2015
FF 2A	AF										2018
											F:=00
01 00	AF										201A
03 FB	BC										201C
02 00	AF										201D
02 FB	BC										201C
08 08	AF		1	0	0	0	1	0	0	0	201F
00 FB	BC										

- Adresse	Befehlscode	Mnemonic
2000	3E 40	LD A,40H
2002	07	RLCA
2003	17	RLA
2004	3E 01	LD A,01H
2006	0F	RR CA
2007	1F	RRA
2008	3E 01	LD A,01H
200A	CB 2F	SRA A
200C	CB 3F	SRL A
200E	76	HALT

Reg.-Inhalt	Register	A-Register	F-Register							PC
			S	Z	X	H	XP/VN	C		
40 00	AF	01000000	0	0	0	0	0	0	0	2002
80 00	AF	10000000	0	0	0	0	0	0	0	2003
00 01	AF	00000000	0	0	0	0	0	0	1	2004
										F:=00
01 00	AF	00000001	0	0	0	0	0	0	0	2006
80 01	AF	10000000	0	0	0	0	0	0	1	2007
00 00	AF	11000000	0	0	0	0	0	0	0	2008
01 00	AF	00000001	0	0	0	0	0	0	0	200A
00 45	AF	00000000	0	1	0	0	0	1	0	200C
00 44	AF	00000000	0	1	0	0	0	1	0	200E

### 3.5.8. Bitmanipulationsbefehle

Mit dieser Befehlsgruppe können einzelne Bits eines Bytes gesetzt oder rückgesetzt werden.

	Byte	Beschreibung
SET b,r	2	Die durch b gekennzeichnete
SET b,(HL)	2	Bitposition wird 1 gesetzt.
SET b,(IX+d)	4	b : 7,6,...,0
SET b,(IY+d)	4	
RES b,r	2	Die durch b gekennzeichnete
RES b,(HL)		Bitposition wird 0 gesetzt.
RES b,(IX+d)	4	
RES b,(IY+d)	4	
BIT b,r	2	Die durch b gekennzeichnete
BIT b,(HL)	2	Bitposition wird komplementiert und ins Z-Flag geladen.
BIT b,(IX+d)	4	
BIT b,(IY+d)	4	

**Beispiel:**

Setzen und Rücksetzen der einzelnen Bitpositionen des A-Registers

Adresse	Befehlscode	Mnemonic
2000	3E 00	LD A,00
2002	CB 6F	Bit 5,A
2004	CB E7	SET 4,A
2006	CB A7	RES 4,A
2008	76	HALT

Reg.-Inhalt	Register	A-Register	F-Register								PC
			S	Z	X	H	XP/VN	C			
00 00	AF	00000000	0	0	0	0	0	0	0	0	2002
00 54	AF	00000000	0	1	0	1	0	1	0	0	2004
10 54	AF	00010000	0	1	0	1	0	1	0	0	2006
00 54	AF	00000000	0	1	0	1	0	1	0	0	2008

**3.5.9. Spezielle Akkumulator- und Flagbefehle**

	Byte	Beschreibung												
DAA	1	<p>Korrigiert nach der Addition/Subtraktion zweier gepackter BCD-Zahlen den Inhalt des Akkumulators so, daß wieder gepacktes BCD-Format entsteht.</p> <p>Bsp.:</p> <table style="margin-left: 40px;"> <tr> <td>0010</td> <td>0110</td> <td>26</td> </tr> <tr> <td>+ 0101</td> <td>1001</td> <td>+ 59</td> </tr> <tr> <td>falsch</td> <td>0111 1111</td> <td>7F</td> </tr> <tr> <td>nach DAA</td> <td>1000 0101</td> <td>85</td> </tr> </table>	0010	0110	26	+ 0101	1001	+ 59	falsch	0111 1111	7F	nach DAA	1000 0101	85
0010	0110	26												
+ 0101	1001	+ 59												
falsch	0111 1111	7F												
nach DAA	1000 0101	85												
NEG	2	Subtraktion des Akkumulatorinhaltes von 0 (Zweierkomplement)												

	Byte	Beschreibung
CCF	1	Komplementieren des C -Flags setzen des C -Flags
CPL	1	bitweises Negieren des Akkumulatorin- haltes

Beispiel:

Adresse	Befehlscode	Mnemonic
2000	3E 00	LD A,0
2002	2F	CPL
2003	ED 44	NEG
2005	3F	CCF
2006	37	SCF
2007	76	HALT

Reg.-Inhalt	Register	F-Register							PC
		S	Z	X	H	XP/VN	C		
00 00	AF	0	0	0	0	0	0	0	2002
FF 3A	AF	0	0	1	1	1	0	1	2003
01 13	AF	0	0	0	1	0	0	1	2005
01 10	AF	0	0	0	1	0	0	0	2006
01 01	AF	0	0	0	1	0	0	0	2007

### 3.5.10. Unterprogrammaufruf- und Rücksprungbefehle

Unterprogramme sind Befehlsfolgen, die bei der Abarbeitung des Programmes ein- oder mehrmals aktiviert werden. Sie können von mehreren Programmen, sowohl Hauptprogrammen als auch Unterprogrammen, aufgerufen werden.

Bei dieser Programmierungs-Technik wird der im Prozessor vorhandene Kellerspeicher (Stackpointer) genutzt. In den Stackpointer wird beim Sprung in ein Unterprogramm der aktuelle Programmzähler (PC) und damit die Rückkehradresse (Adresse des im Hauptprogramm nächsten Befehls) automatisch eingeschrieben.



Der Einsprung in ein Unterprogramm erfolgt über den Befehl CALL. Das Unterprogramm endet mit einem RET-Befehl.

	Byte	Beschreibung
CALL nn	3	Unbedingter Unterprogrammaufruf. Im Stackpointer wird aktueller Programmzähler (PC) gerettet, der dann mit Adresse nn geladen wird. (SP-1):= PC <sub>H</sub> (SP-2):= PC <sub>L</sub> PC := nn
CANZ nn	3	Unterprogrammaufruf, wenn Z-Flag=0
CAZ nn	3	Unterprogrammaufruf, wenn Z-Flag=1
CANC nn	3	Unterprogrammaufruf, wenn C -Flag=0
CAC nn	3	Unterprogrammaufruf, wenn C -Flag=1
CAPO nn	3	Unterprogrammaufruf, wenn P/V-Flag=0
CAPE nn	3	Unterprogrammaufruf, wenn P/V-Flag=1
CAP nn	3	Unterprogrammaufruf, wenn S-Flag=0
CAM nn	3	Unterprogrammaufruf, wenn S-Flag=1
RST p	1	Spezieller Unterprogrammaufruf. Dabei erfolgt ein Sprung zu der angegebenen Adresse p. p kann sein: 00H, 08H, 10H, 18H 20H, 28H, 30H, 38H Die damit aktivierte Befehlsfolge muß mit einem RET-Befehl enden.
RET	1	Unbedingte Rücksprung. Der Inhalt der durch den Stackpointer adressierten Speicherzelle wird zum aktuellen Programmzähler. Anschließend wird der Stackpointer um 2 erhöht. PC <sub>L</sub> := (SP) PC <sub>H</sub> := (SP+1) SP := SP+2

	Byte	Beschreibung
RNZ	1	Unterprogrammrücksprung, wenn Z-Flag = 0
RZ	1	Unterprogrammrücksprung, wenn Z-Flag = 1
RNC	1	Unterprogrammrücksprung, wenn C-Flag = 0
RC	1	Unterprogrammrücksprung, wenn C-Flag = 1
RPO	1	Unterprogrammrücksprung, wenn P/V-Flag = 0
RPE	1	Unterprogrammrücksprung, wenn P/V-Flag = 1
RP	1	Unterprogrammrücksprung, wenn S-Flag = 0
RM	1	Unterprogrammrücksprung, wenn S-Flag = 1
RETI	2	Rückkehr aus einer Interrupt- serviceroutine im Mode 2
RETN	2	Rücksprung aus einer nicht- maskierbaren Interrupt- serviceroutine

Beispiel:

Sprung ins Unterprogramm ab Adresse 2050 und Rücksprung  
aus dem Unterprogramm

Adresse	Befehlscode	Mnemonic
2000	CD 50 20	CALL 2050
2003	76	HALT
2050	00	NOP
2051	00	NOP
2052	C9	RET
PC		
2000		
2050		
2051		
2052		
2003		

### 3.5.11. Allgemeine Steuerbefehle

	Byte	Beschreibung
NOP	1	Die CPU führt keine Operation aus.
HALT	1	CPU führt solange NOP-Befehle aus, bis ein Interrupt- oder der RESET- Eingang aktiv wird.
DI	1	Interrupt abweisen IFF 1 = 0, IFF 2 = 0 IFF 1: Interrupt-Annahme-Flip-Flop IFF 2: Interrupt-Zwischenspeicher-Flip- Flop
EI	1	Interrupt annehmen IFF 1 = 1, IFF 2 = 1
IMO	2	Setzen Interruptmodes 0
IM1	2	Setzen Interruptmodes 1
IM2	2	Setzen Interruptmodes 2

### 3.5.12. Ein- und Ausgabebefehle

	Byte	Beschreibung
<u>PORT → CPU/Reg.</u>		
IN n	2	A: = (n) Akkumulatorinhalt wird mit Inhalt des durch n adressierten Eingabekanals geladen n: Direktwert
IN r	2	r: = (C) Register r wird mit Inhalt des durch C adressierten Eingabekanals geladen. C: enthält Adresse des Eingabekanals
IN F	2	Das F-Register wird mit dem Zustand des durch C adressierten Eingabekanals geladen
<u>PORT → Speicher</u>		
INI	2	(HL): = (C)    B: = B-1 HL:= HL+1 Durch HL adressierter Speicherplatz wird mit Inhalt des durch C adressierten Eingabekanals geladen. HL wird um 1 erhöht, B um 1 erniedrigt. C: wird vor Abarbeitung von INI geladen
INIR	2	(HL): = (C)    B: = B-1 HL:= HL+1 analog INI; Wiederholung der Befehlsausführung bis B = 0
IND	2	(HL): = (C)    B: = B-1 HL:= HL-1 analog INI; HL wird um 1 verringert

	Byte	Beschreibung
INDR	2	(HL): = (C)    B: = B-1 HL: = HL-1  analog INIR; HL wird um 1 verringert
<u>CPU/Reg. → PORT</u>		
OUT n	2	(n): = A Inhalt des Akkumulators wird an den Ausgabekanal mit der Adresse n gegeben n: Direktwert
OUT r	2	(C): = r Registerinhalt wird an den durch C adressierten Ausgabekanal gegeben. Adresse des Ausgabekanal muß vorher in C geladen werden.
<u>Speicher → PORT</u>		
OUTI	2	(C): = (HL)    B: = B-1 HL: = HL+1  Der Inhalt, der durch HL adressierten Speicherzelle wird an den durch C adressierten Ausgabekanal ausgegeben. Danach wird B um 1 verringert und HL um 1 erhöht.
OTIR	2	(C): = (HL)    B: = B-1 HL: = HL+1  analog OUTI; Befehl wird so oft wiederholt bis B = 0 ist.
OUTD	2	(C): = (HL)    B: = B-1 HL: = HL-1  analog OUTI; aber HL wird um 1 verringert

	Byte	Beschreibung
OTDR	2	(C): = (HL)      B: = B-1 HL: = HL-1  analog OTIR, aber HL wird um 1 verringert.

**Bemerkung:**

Bei den Eingabe- und Ausgabebefehlen liegt die Kanaladresse auf dem unteren Adreßbus A0 ... A7. Auf dem oberen Adreßbus A8 bis A15 liegt bei IN n und OUT n der Akkuinhalt A; bei den Befehlen IN r, OUT r der Inhalt von B; bei den Befehlen INI, INIR, IND, INDR der nicht dekrementierte Wert von B und bei den Befehlen OUT, OTIR, OUTD, OTDR der dekrementierte Wert von B.

Anwendungsbeispiele hierzu sind im Abschnitt 3. enthalten.

## 4. Programmierung der Peripherieschaltkreise des IC 80

Nachfolgend soll anhand einiger Beispiele die Programmierung von PIO und CTC erläutert werden. Diese Programmierung erfolgt auf der Grundlage der in den Abschnitten 2.3.4. und 2.4.3. aufgeführten schaltkreisspezifischen Programmiervorschriften.

Es wurde hierbei nicht angestrebt, alle möglichen Varianten zu erfassen, sondern eher den prinzipiellen Aufbau dieser Programmteile zu verdeutlichen.

Die Lösung eigenständiger Anwendungsaufgaben ist dann unter Nutzung o. g. Abschnitte leicht möglich.

Zuerst soll nochmals kurz auf die programmäßig notwendigen Aktivitäten zur Organisation einer Interruptserviceroutine (ISR- (vgl. Abschnitt 2.2.2.) in Verbindung mit CTC und PIO eingegangen werden.

### 4.1. Programmäßige Organisation einer Interruptserviceroutine

Nach dem im Abschnitt 2.2.2. dargelegten Ablaufes für den hierbei betrachteten Interruptmode 2 (IM 2) sind folgende Aktivitäten erforderlich:

- Im Grundzustand nach RESET ist die Interruptannahme (INT) gesperrt und muß mittels EI-Befehl erst freigegeben werden. Gleiches gilt nach einer erfolgten Interruptannahme.
- Prinzipiell ist ein Interrupt-Aufruf in Mode 2 mit einem Unterprogrammaufruf mittels CALL-Befehl vergleichbar.

In der Regel wird es notwendig sein, die in der ISR verwendeten Register entweder durch Arbeit mit dem Alternativregistersatz (EXX, EXAF) oder durch PUSH-Befehle zu retten. Am Ende der ISR ist der Rücktausch (EXAF, EXX) bzw. die Rückkellerung der Register (POP) notwendig.

- Das im Rahmen einer ISR abzuarbeitende Unterprogramm kann an beliebiger Stelle im RAM abgespeichert werden und muß zur Kennzeichnung des programmäßigen Endes dieser ISR mit dem Befehl RETI (bei NMI : RETN) abgeschlossen werden.
- Die Startadresse dieses Programms kann ebenfalls an beliebiger Stelle im RAM abgespeichert werden. Die hierfür gewählte RAM-Adresse bestimmt den Inhalt des zu ladenden I-Registers (oberer Adreßteil) sowie des Interruptvektors (unterer Adreßteil) des interruptauslösenden Peripheriekanals.

Ein kurzes Prinzipbeispiel soll die erforderliche Programmstruktur verdeutlichen:

Drei Peripheriekanäle sollen interruptfähig sein. Die jeweiligen Interrupt-Unterprogramme seien im RAM nacheinanderfolgend abgespeichert.

(Die verwendeten Adressen wurden beliebig gewählt.)

```

      .
      .
LD A,23H      oberer Adreßteil der Startadressentafel
LD I,A        der INT-Unterprogramme in I-Register laden
IM 2

      .
      .
OUT (Peripheriekanal 1, Steueradresse),00H  Laden der jeweiligen
      .                                     Interruptvektoren
      .                                     im Rahmen der Ini-
OUT (Peripheriekanal 2, Steueradresse),02H  tialisierung der
      .                                     Peripherieschalt-
      .                                     kreise (Beachte spe-
OUT (Peripheriekanal 3, Steueradresse),04H  zielle Interruptvek-
      .                                     torerzeugung der
      .                                     CTC-Kanäle 1 - 3)

```



```

      .
      .
      .
Schleife: EI
          HALT
          JR Schleife

```

```

      .
      .
      .
2060H: . . . .
          (INT-Unterprogramm 1)

```

```

      .
      .
      .
          RETI
2073H: . . . .
          (INT-Unterprogramm 2)

```

```

      .
      .
      .
          RETI
20A9H: . . . .
          (INT-Unterprogramm 3)

```

```

      .
      .
      .
          RETI

```

```

      .
      .
      .

```

2300H:DEFW	2060H	Startadressentafel der
2302H:DEFW	2073H	INT- Unterprogramme
2304H:DEFW	20A9H	





Beispiel 3: PIO D207/Port B soll in Bit-Mode arbeiten, dabei sollen B0 - B3 Ausgänge und B4 - B7 Eingänge sein, Interrupt gesperrt. Anschließend soll ein Daten-schreibe- und -lesezyklus erfolgen:

•  
•  
•

D A,CFH Betriebsarten-Steuerwort: 1 1 0 0 1 1 1 1  
Bit-Mode

JT (FBH),A

D A,FOH Definierung der Ein-/Ausgänge: 1 1 1 1 0 0 0 0  
Eingänge Ausgänge

JT (FBH),A

D A,O7H INT-Steuerwort: 0 0 0 0 0 1 1 1  
INT gesperrt Identifikation als INT-Steuerwort  
Maske folgt nicht

JT (FBH),A

•  
•  
•

D A,00

JT (F9H),A über Port B (Datenadresse) wird der Inhalt vom A-Register ausgegeben, d. h. hierbei wirksam nur für die als Ausgänge programmierten Anschlüsse B0 - B3

•

IN A, (F9H)

Einlesen von Port B (Datenadresse) in das A-Register der CPU. Dabei werden die an B4 - B7 (Eingänge) anliegenden Daten sowie die im Ausgaberegister von B0 - B3 (Ausgänge) enthaltenen Werte (hierbei B0 - B3 = 0) gelesen.

Beispiel 4: PIO D207/Port B soll in Bit-Mode arbeiten, B0 - B7 sollen Eingänge sein, wobei ein Interrupt unter folgender Bedingung ausgelöst werden soll:

$\overline{B0} \cdot \overline{B1} \cdot \overline{B3}$  (Low-Pegel an B0, B1 und B3 → UND-Verknüpfung)

(INT-Vektor = 70H gewählt) :

.  
.  
.

LD A, 70H

INT-Vektor: 0 1 1 1 0 0 0 0

Identifikation als INT-Vektor

OUT (FBH), A

LD A, CFH

Betriebsarten-Steuersatz

OUT (FBH), A

LD A, FFH

Definition der Ein-/Ausgänge

OUT (FBH), A

LD A, D7H

INT-Steuersatz: 1 1 0 1 0 1 1 1

INT-Freigabe    UND-Verknüpfung    Low-aktiv    Maske folgt

OUT (FBH), A

l, F4H

Maskierung: 1 1 1 1

0 1 0 0  
| | | |  
B3, B1, B0 für INT  
ausgewertet

(FBH), A

.  
.  
.

. CTC-Programmierung (vgl. hierzu Abschnitt 2.4.3.)

spiel 1: CTC D208/Kanal 1 soll als Zeitgeber arbeiten, wobei bei der Systemtakt (Bezugsbasis 900 kHz) durch den Vorteilerfaktor 256 geteilt wird.

Die Zeitkonstante soll 09H betragen. Die Zeitgeberoperation soll mit der steigenden Systemtaktflanke des CPU-Maschinenzyklusses, der auf das Laden der Zeitkonstante folgt (Software-Triggerung), beginnen.

INT freigegeben, INT-Vektor Kanal 0 = 50H gewählt.

.  
.  
.

A, 50H

INT-Vektor: 0 1 0 1

0 0 0 0

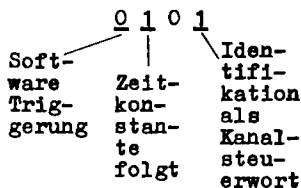
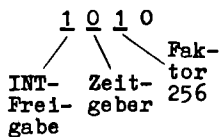
Identifikation als  
INT-Vektor

(ECH), A

INT-Vektor in Kanal 0 (!) einschreiben  
(gemäß Abschnitt 2.4.3. wird damit bei INT-Annahme von Kanal 1 der INT-Vektor = 52H gesendet!)

LD A, A5H

Kanalsteuerwort:



OUT (EDH), A      Kanalsteuerwort in Kanal 1 einschreiben

LD A, 09H          Zeitkonstante 09H

OUT (EDH), A

Beginn der Zeitgeberoperation

.  
. .

Die Dauer bis zur INT-Auslösung (Nulldurchgang des Rückwärtszählers) beträgt damit gemäß Abschnitt 2.4.3.2.

$$t = \frac{1}{900} \text{ ms} \cdot 256 \cdot 9 = 2,56 \text{ ms}$$

Beispiel 2: Kanal 3 soll als Zähler arbeiten, wobei die am CTC-Eingang C/TRG3 ankommenden Impulse mit der positiven Flanke (L → H) gezählt werden. Beim Nulldurchgang des Rückwärtszählers soll ein Interrupt erzeugt werden (INT-Vektor Kanal 0 = 50H gewählt).

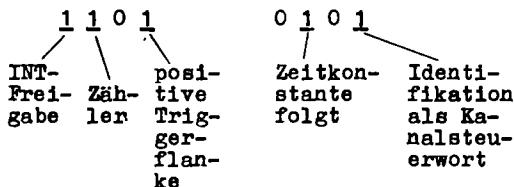
Als Zeitkonstante soll der maximale Wert = 0 verwendet werden (entspricht einer Zeitkonstante von 256, d. h. nach 256 Eingangsimpulsen erfolgt INT-Anmeldung).

•  
•  
•

D A,50H

UT (ECH),A      INT-Vektor in Kanal 0 einschreiben (damit  
wird bei INT-Annahme von Kanal 3 der INT-  
Vektor = 56H gesendet)

D A,D5H      Kanalsteuerwort:



UT (EFH),A      Kanalsteuerwort in Kanal 3 einschreiben

D A,00

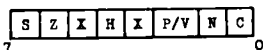
UT (EFH),A      Zeitkonstante 0: nach 256 Eingangsimpulsen  
erfolgt INT-Anmeldung  
durch CTC



## 5. Befehlsliste U880

Symbol	Bedeutung
C	Übertragsflag. C = 1, wenn die Operation einen Übertrag vom MSB des Operanden oder des Ergebnisses erzeugt.
Z	Null-Flag. Z = 1, wenn das Ergebnis der Operation Null ist.
S	Vorzeichen-Flag. S = 1, wenn das MSB des Ergebnisses eins ist.
P/V	Paritäts- oder Überlauf-Flag. Parität (P) und Überlauf (V) benutzen das gleiche Flag. Logische Operationen beeinflussen das Flag entsprechend der Parität des Ergebnisses, arithmetische Operationen stellen dieses Flag entsprechend dem Überlauf des Ergebnisses. P/V = 1, wenn das Ergebnis paarig ist, P/V = 0, wenn das Ergebnis unpaarig ist. P/V = 1, wenn das Ergebnis einen Überlauf enthält.
H	Halbbyte-Übertragsflag. H = 1, wenn Addition oder Subtraktion einen Übertrag innerhalb von 4 Akkumulatorbits erzeugen.
N	Additions-/Subtraktionsflag. N = 1, wenn vorangegangene Operation eine Subtraktion war. H- und N-Flags werden für die Dezimalkorrektur (DAA) benutzt, um das Ergebnis einer Addition oder Subtraktion von gepackten BCD-Zahlen in das Format gepackter BCD-Zahlen zu wandeln.
‡	Flag wird entsprechend dem Ergebnis der Operation gestellt
.	Flag wird durch die Operation nicht beeinflusst
0	Flag wird durch die Operation gelöscht
1	Flag wird durch die Operation gesetzt
X	Flag unbestimmt
V	P/V-Flag entspricht dem Ergebnis-Überlauf der Operation
P	P/V-Flag entspricht der Parität des Ergebnisses der Operation
r	eines der U880D - Register A, B, C, D, E, H, L.
s	ein 8-Bit-Speicherplatz, der durch eine der für den jeweiligen Befehl zulässigen Adressierungsarten definiert ist.
dd	ein 16-Bit-Speicherplatz, der durch eine der für diesen Befehl zulässigen Adressierungsarten definiert ist.
ii	eines der zwei Indexregister IX oder IY
R	Auffrischzähler
n	8-Bit im Bereich 0 - 255
nn	16-Bit im Bereich 0 - 65535
A	P/V-Flag ist 0, wenn das Ergebnis von BC-1 = 0, sonst P/V = 1
B	Z-Flag ist 1, wenn A = M, sonst Z = 0
IFF1	Interrupt-Annahme-Flip-Flop
IFF2	Interrupt-Zwischenspeicher-Flip-Flop
e	stellt die Abstandsangabe in der relativen Adressierungsart dar, bezogen auf das 1. Byte des Sprungbefehls, e ist ein Zweierkomplement mit Vorzeichen im Bereich -126 bis +129
e-2	ergibt im Operationscode die tatsächliche Adresse PC+e, da der Befehlszähler vor der Addition von e um 2 erhöht worden ist.
s <sub>b</sub>	bezeichnet das Bit b (0...7) des Speicherplatzes s
s	falls B-1 = 0, wird Z = 1 gesetzt, sonst Z = 0

Befehl	Flags						Bemerkungen
	C	Z	P/V	S	N	H	
ADD s, ADC s	‡	‡	V	‡	0	‡	8-Bit-Addition oder Addition mit Übertrag
SUB s, SBC s, CMP s, NEG	‡	‡	V	‡	1	‡	8-Bit-Subtraktion, Subtraktion mit Übertrag, Vergleich und Negation des Akkumulators
AND s	0	‡	P	‡	0	1	Logische Operationen
OR s, XOR s	0	‡	P	‡	0	0	
INC s	.	‡	V	‡	0	‡	8-Bit-Erhöhung
DEC s	.	‡	V	‡	1	‡	8-Bit-Erniedrigung
ADD HL,dd	‡	.	.	.	0	X	16-Bit-Addition
ADC HL,dd	‡	‡	V	‡	0	X	16-Bit-Addition mit Übertrag
SBC HL,dd	‡	‡	V	‡	1	X	16-Bit-Subtraktion mit Übertrag
RLA, RLCA, RRA, RRCA	‡	.	.	.	0	0	zyklische Verschiebung - Akkumulator
RL s, RLC s, RR s, RRC s	‡	‡	P	‡	0	0	zyklische Verschiebung - Speicherplatz s
SLA s, SRA s, SRL s	‡	‡	P	‡	0	0	Verschiebung - Speicherplatz s
RLD, RRD	.	‡	P	‡	0	0	zyklische Verschiebung - Zahl links und rechts
DAA	‡	‡	P	‡	.	‡	Dezimalen Einrichtung - Akkumulator
CPL	.	.	.	.	1	1	Komplement des Akkumulators
SCF	1	.	.	.	0	0	Setzen des Übertrags
CCF	‡	.	.	.	0	X	Komplement des Übertrags
IN r, INF	.	‡	P	‡	0	0	Eingabe, indirekte Registeradresse
INI, IND, OUTI, OUTD	.	‡	X	X	1	X	Block-Ein- und Ausgabe, Z = 0 wenn B ≠ 0, sonst Z = 1
INIR, INDR, OTIR, OTDR	.	1	X	X	1	X	Z = 0 wenn B ≠ 0, sonst Z = 1
LDI, LDD	.	X	‡	X	0	0	Blocktransfer-Befehle
LDIR, LDDR	.	X	0	X	0	0	P/V = 1 wenn BC ≠ 0, sonst P/V = 0
CPI, CPIR, CPD, CPDR	.	‡	‡	X	1	X	Block-Such-Befehle, Z = 1 wenn A = (HL), sonst Z = 0; P/V = 1 wenn BC ≠ 0, sonst P/V = 0
LD A,I; LD A,R	.	‡	IFF2	‡	0	0	Inhalt des Interrupt-Akzeptanz-Flip-Flops 2 (IFF2) ins P/V-Flag überführt
BIT b,s	.	‡	X	X	0	1	Zustand des Bits b im Speicherplatz s ins Z-Flag überführt



X: Bit hat keine Bedeutung

Format des Flag-Registers

Assembler Sprache	symbolische Operation	Flags					Operationscode	Bytes	M-Zyklen	Takte	Bemerkung
		C	Z	P/V	S	N					
8 - Bit - Ladegruppe											
LD $r_1, r_2$	$r_1 \leftarrow r_2$	.	.	.	.	.	01 $r_1$ $r_2$	1	1	4	$r_1, r_2$   Register
LD $r, n$	$r \leftarrow n$	.	.	.	.	.	00 $r$ 110 - n -	2	2	7	000   B 001   C 010   D 011   E 100   H 101   L 111   A
LD $r, M$	$r \leftarrow M$	.	.	.	.	.	01 $r$ 110	1	2	7	
LD $r, (IX+d)$	$r \leftarrow (IX+d)$	.	.	.	.	.	11 011 101 01 $r$ 110 - d -	3	5	19	
LD $r, (IY+d)$	$r \leftarrow (IY+d)$	.	.	.	.	.	11 111 101 01 $r$ 110 - d -	3	5	19	
LD $M, r$	$M \leftarrow r$	.	.	.	.	.	01 110 $r$	1	2	7	r <sub>1</sub> , r <sub>2</sub> steht für eines der Register A, B, C, D, E, H, L.
LD $(IX+d), r$	$(IX+d) \leftarrow r$	.	.	.	.	.	11 011 101 01 110 $r$ - d -	3	5	19	
LD $(IY+d), r$	$(IY+d) \leftarrow r$	.	.	.	.	.	11 111 101 01 110 $r$ - d -	3	5	19	
LD $M, n$	$M \leftarrow n$	.	.	.	.	.	00 110 110 - n -	2	3	10	
LD $(IX+d), n$	$(IX+d) \leftarrow n$	.	.	.	.	.	11 011 101 00 110 110 - d - - n -	4	5	19	
LD $(IY+d), n$	$(IY+d) \leftarrow n$	.	.	.	.	.	11 111 101 00 110 110 - d - - n -	4	5	19	
LD A, (BC)	$A \leftarrow (BC)$	.	.	.	.	.	00 001 010	1	2	7	
LD A, (DE)	$A \leftarrow (DE)$	.	.	.	.	.	00 011 010	1	2	7	
LD A, (nn)	$A \leftarrow (nn)$	.	.	.	.	.	00 111 010 - n - - n -	3	4	13	
LD (BC), A	$(BC) \leftarrow A$	.	.	.	.	.	00 000 010	1	2	7	
LD (DE), A	$(DE) \leftarrow A$	.	.	.	.	.	00 010 010	1	2	7	
LD (nn), A	$(nn) \leftarrow A$	.	.	.	.	.	00 110 010 - n - - n -	3	4	13	
LD A, I	$A \leftarrow I$	.	‡	IFF	‡	0 0	11 101 101 01 010 111	2	2	9	
LD A, R	$A \leftarrow R$	.	‡	IFF	‡	0 0	11 101 101 01 011 111	2	2	9	
LD I, A	$I \leftarrow A$	.	.	.	.	.	11 101 101 01 000 111	2	2	9	
LD R, A	$R \leftarrow A$	.	.	.	.	.	11 101 101 01 001 111	2	2	9	

16 - Bit - Ladegruppe

LD $dd, nn$	$dd \leftarrow nn$	.	.	.	.	.	00 $dd$ 001 - n - - n -	3	3	10	$dd$   Paar 00   BC 01   DE 10   HL 11   SP
LD $IX, nn$	$IX \leftarrow nn$	.	.	.	.	.	11 011 101 00 100 001 - n - - n -	4	4	14	
LD $IY, nn$	$IY \leftarrow nn$	.	.	.	.	.	11 111 101 00 100 001 - n - - n -	4	4	14	
LD $HL, (nn)$	$H \leftarrow (nn+1)$ $L \leftarrow (nn)$	.	.	.	.	.	00 101 010 - n - - n -	3	5	16	

Assembler Sprache	symbolische Operation	Flags						Operations-code 76 543 210	By-tes	M-Zyk-len	Takte	Bemerkung										
		C	Z	P/V	S	N	H															
LD dd, (nn)	ddH←(nn+1) ddL←(nn)	.	.	.	.	.	.	11 101 101 01 dd1 011 - n - - n -	4	6	20	dd ist eines der Registerpaare BC, DE, HL, SP										
LD IX, (nn)	IXH←(nn+1) IXL←(nn)	.	.	.	.	.	.	11 011 101 00 101 010 - n - - n -	4	6	20											
LD IY, (nn)	IYH←(nn+1) IYL←(nn)	.	.	.	.	.	.	11 111 101 00 101 010 - n - - n -	4	6	20											
LD (nn), HL	(nn+1)←H (nn)←L	.	.	.	.	.	.	00 100 010 - n - - n -	3	5	16											
LD (nn), dd	(nn+1)←ddH (nn)←ddL	.	.	.	.	.	.	11 101 101 01 dd0 011 - n - - n -	4	6	20											
LD (nn), IX	(nn+1)←IXH (nn)←IXL	.	.	.	.	.	.	11 011 101 00 100 010 - n - - n -	4	6	20											
LD (nn), IY	(nn+1)←IYH (nn)←IYL	.	.	.	.	.	.	11 111 101 00 100 010 - n - - n -	4	6	20	<table border="0"> <tr> <td>qq</td> <td>Paar</td> </tr> <tr> <td>00</td> <td>BC</td> </tr> <tr> <td>01</td> <td>DE</td> </tr> <tr> <td>10</td> <td>HL</td> </tr> <tr> <td>11</td> <td>AF</td> </tr> </table> <p>qq ist eines der Registerpaare AF, BC, DE, HL.</p> <p>(Paar)H bzw. (Paar)L bezieht sich auf die oberen bzw. unteren 8 Bits d. entspr. Registerpaars, d.h. BCL=C, AFH=A.</p>	qq	Paar	00	BC	01	DE	10	HL	11	AF
qq	Paar																					
00	BC																					
01	DE																					
10	HL																					
11	AF																					
LD SP, HL	SP←HL	.	.	.	.	.	.	11 111 001	1	1	6											
LD SP, IX	SP←IX	.	.	.	.	.	.	11 011 101 11 111 001	2	2	10											
LD SP, IY	SP←IY	.	.	.	.	.	.	11 111 101 11 111 001	2	2	10											
PUSH qq	(SP-2)←qqL (SP-1)←qqH SP←SP-2	.	.	.	.	.	.	11 qq0 101	1	3	11											
PUSH IX	(SP-2)←IXL (SP-1)←IXH SP←SP-2	.	.	.	.	.	.	11 011 101 11 100 101	2	4	15											
PUSH IY	(SP-2)←IYL (SP-1)←IYH SP←SP-2	.	.	.	.	.	.	11 111 101 11 100 101	2	4	15											
POP qq	qqH←(SP+1) qqL←(SP) SP←SP+2	.	.	.	.	.	.	11 qq0 001	1	3	10											
POP IX	IXH←(SP+1) IXL←(SP) SP←SP+2	.	.	.	.	.	.	11 011 101 11 100 001	2	4	14											
POP IY	IYH←(SP+1) IYL←(SP) SP←SP+2	.	.	.	.	.	.	11 111 101 11 100 001	2	4	14											

#### Austausch-, Blocktransfer- und Suchgruppe

EX DE, HL	DE↔HL	.	.	.	.	.	.	11 101 011	1	1	4	
EX AF	AF↔AF'	.	.	.	.	.	.	00 001 000	1	1	4	
EXX	BC↔DE DE↔BC HL↔HL'	.	.	.	.	.	.	11 011 001	1	1	4	Vertauschung Registeratz/ Alternativregisteratz
EX (SP), HL	H↔(SP+1) L↔(SP)	.	.	.	.	.	.	11 100 011	1	5	19	
EX (SP), IX	IXH↔(SP+1) IXL↔(SP)	.	.	.	.	.	.	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IYH↔(SP+1) IYL↔(SP)	.	.	.	.	.	.	11 111 101 11 100 011	2	6	23	

Assembler Sprache	symbolische Operation	Flags						Operationscode 76 543 210	Ry-tes	M-Zyk-len	Takte	Bemerkung
		C	Z	P	V	S	H					
LDI	(DE)←M DE←DE+1 HL←HL+1 BC←BC-1	. X		X	O	O		11 101 101 10 100 000	2	4	16	
LDIR	(DE)←M DE←DE+1 HL←HL+1 BC←BC-1 Wiederholung bis BC=0	. X	O	X	O	O		11 101 101 10 110 000	2 2	5 4	21 16	wenn BC≠0 wenn BC=0
LDD	(DE)←M DE←DE-1 HL←HL-1 BC←BC-1	. X			X	O	O	11 101 101 10 101 000	2	4	16	
LDDR	(DE)←M DE←DE-1 HL←HL-1 BC←BC-1 Wiederholung bis BC=0	. X	O	X	O	O		11 101 101 10 111 000	2 2	5 4	21 16	wenn BC≠0 wenn BC=0
CPI	A-M HL←HL+1 BC←BC-1	.				X	1 X	11 101 101 10 100 001	2	4	16	
CFIR	A-M HL←HL+1 BC←BC-1 Wiederholung bis BC=0 oder A=M	.				X	1 X	11 101 101 10 110 001	2 2	5 4	21 16	wenn BC≠0 u. A=M wenn BC=0 o. A=M
CPD	A-M HL←HL-1 BC←BC-1	.				X	1 X	11 101 101 10 101 001	2	4	16	
CPDR	A-M HL←HL-1 BC←BC-1 Wiederholung bis BC=0 oder A=M	.				X	1 X	11 101 101 10 111 001	2 2	5 4	21 16	wenn BC≠0 u. A=M wenn BC=0 o. A=M

#### 8-Bit-Arithmetik und logische Gruppe

ADD r	A←A+r					V			0	10	000	r	1	1	4	r	Regi-ster
ADD n	A←A+n					V			0	11	000	110	2	2	7	000	
											- n -					001	C
ADD M	A←A+M					V			0	10	000	110	1	2	7	010	D
ADD (IX+d)	A←A+(IX+d)					V			0	11	011	101	3	5	19	011	E
										10	000	110				100	H
											- d -					101	L
																111	A
ADD (IY+d)	A←A+(IY+d)					V			0	11	111	101	3	5	19		
										10	000	110					
											- d -						
ADC s	A←A+s+CY					V			0		001						
SUB s	A←A-s					V			1		010						
																	s ist eines der r, n, M, (IX+d), (IY+d) wie beim ADD-Befehl
																	umrandete Bits ersetzen 000 in ADD-Befehl
SBC s	A←A-s-CY					V			1		011						
AND s	A←A&s					P			0 1		100						
OR s	A←A s					P			0 0		110						
XOR s	A←A⊕s					P			0 0		101						
CMP s	A - s					V			1		111						
INC r	r←r+1	.				V			0	00	r	100	1	1	4		

Assembler Sprache	symbolische Operation	Flags				Operationscode	Bytes	M-Zyklen	Takte	Bemerkung			
		C	Z	P/V	S/H/H								
INC M	M ← M+1	.	‡	V	‡	0	‡	00 110	100	1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	.	‡	V	‡	0	‡	11 011 101		3	6	23	
								00 110	100				
								- d -					
INC (IY+d)	(IY+d) ← (IY+d)+1	.	‡	V	‡	0	‡	11 111 101		3	6	23	
								00 110	100				
								- d -					
DEC f	f ← f-1	.	‡	V	‡	1	‡		101				f ist eines der r. M. (IX+d), (IY+d) wie bei INC; gleiches Format u. Zustände wie INC. 100 durch 101 im Operationscode ersetzen.

### Allgemeine Arithmetik und USSOD - Steuergruppe

DAA	Wandelt AC- Inhalt in gepackt. BCD- Format nach Add o. Sub- traktion v. gepackt. BCD- Zahlen	.	.	.	.	.	.	00 100 111		1	1	4	Dezimalkorrektur im Akkumulator
CPL	A ← $\bar{A}$	.	.	.	.	.	.	00 101 111		1	1	4	Komplement d. Akkumulators; Einerkomplement
NEG	A ← $\bar{A}+1$	‡	‡	V	‡	1	‡	11 101 101 01 000 100		2	2	8	Negation AC; Zweierkomplement
CCF	CY ← $\bar{CY}$	‡	.	.	.	.	0	01 111 111		1	1	4	Komplement d. Übertrags-Flag
SCF	CY ← 1	1	.	.	.	.	0	00 110 111		1	1	4	Setzen d. Übertrags-Flag
NOP	keine Operation	.	.	.	.	.	.	00 000 000		1	1	4	
HALT	USSOD im HALT-Zustand	.	.	.	.	.	.	01 110 110		1	1	4	
DI	IFF1 ← 0, IFF2 ← 0	.	.	.	.	.	.	11 110 011		1	1	4	
EI	IFF1 ← 1, IFF2 ← 1	.	.	.	.	.	.	11 111 011		1	1	4	
IM0	Setzen d. Interrupt-Mode 0	.	.	.	.	.	.	11 101 101 01 000 110		2	2	8	
IM1	Setzen d. Interrupt-Mode 1	.	.	.	.	.	.	11 101 101 01 010 110		2	2	8	
IM2	Setzen d. Interrupt-Mode 2	.	.	.	.	.	.	11 101 101 01 011 110		2	2	8	



### 16-Bit-Arithmetik

ADD HL, dd	HL ← HL+dd	‡	.	.	.	.	0	01 dd1 001		1	3	11	dd Register
ADC HL, dd	HL ← HL+dd +CY	‡	‡	V	‡	0	‡	11 101 101 01 dd1 010		2	4	15	00 BC 01 DE 10 HL 11 SP
SBC HL, dd	HL ← HL-dd -CY	‡	‡	V	‡	1	‡	11 101 101 01 dd0 010		2	4	15	dd ist eines d. Registerpaars; BC, DE, HL, SP

Assembler Sprache	symbolische Operation	Flags				Operationscode 76 543 210	Ry-tes	M-Zyk-len	Takte	Bemerkung	
		C	Z	P/V	S N H						
ADD IX, pp	IX ← IX + pp	‡	.	.	.	0 X	11 011 101 00 pp1 001	2	4	15	pp   Register 00   BC 01   DE 10   IX 11   SP  pp ist eines d. Registerpaare: BC, DE, IX, SP
ADD IY, pp	IY ← IY + pp	‡	.	.	.	0 X	11 111 101 00 pp1 001	2	4	15	pp   Register 00   BC 01   DE 10   IX 11   SP  pp ist eines d. Registerpaare: BC, DE, IY, SP
INC dd	dd ← dd + 1	.	.	.	.	.	00 dd0 011	1	1	6	
INC IX	IX ← IX + 1	.	.	.	.	.	11 011 101 00 100 011	2	2	10	
INC IY	IY ← IY + 1	.	.	.	.	.	11 111 101 00 100 011	2	2	10	
DEC dd	dd ← dd - 1	.	.	.	.	.	00 dd1 011	1	1	6	
DEC IX	IX ← IX - 1	.	.	.	.	.	11 011 101 00 101 011	2	2	10	
DEC IY	IY ← IY - 1	.	.	.	.	.	11 111 101 00 101 011	2	2	10	

Befehlsgruppe: Verschiebung und zyklische Verschiebung

RLCA	$\boxed{C} \leftarrow \boxed{7 \rightarrow 0}$ A	‡	.	.	.	0 0	00 000 111	1	1	4	zyklische Verschiebung AC, linksherum
RLA	$\boxed{C} \leftarrow \boxed{7 \rightarrow 0}$ A	‡	.	.	.	0 0	00 010 111	1	1	4	zyklische Verschiebung AC, nach links
RRCA	$\boxed{7 \rightarrow 0} \leftarrow \boxed{C}$ A	‡	.	.	.	0 0	00 001 111	1	1	4	zyklische Verschiebung AC, rechtsherum
RRA	$\boxed{7 \rightarrow 0} \leftarrow \boxed{C}$ A	‡	.	.	.	0 0	00 011 111	1	1	4	zyklische Verschiebung AC, nach rechts
RLC r		‡ ‡	P	‡	0 0	0	11 001 011 00 $\boxed{000}$ r	2	2	8	zyklische Verschiebung, Register, linksherum
RLC M		‡ ‡	P	‡	0 0	0	11 001 011 00 $\boxed{000}$ 110	2	4	15	r   Register
RLC (IX+d)	$\boxed{C} \leftarrow \boxed{7 \rightarrow 0}$ r, M (IX+d) (IY+d)	‡ ‡	P	‡	0 0	0	11 011 101 11 001 011 - d - 00 $\boxed{000}$ 110	4	6	23	000   B 001   C 010   D 011   E 100   H 101   L 111   A
RLC (IY+d)		‡ ‡	P	‡	0 0	0	11 111 101 11 001 011 - d - 00 $\boxed{000}$ 110	4	6	23	
RL s	$\boxed{C} \leftarrow \boxed{7 \rightarrow 0}$ s=r, M (IX+d), (IY+d)	‡ ‡	P	‡	0 0	0	$\boxed{010}$				Befehlsformat u. Zustände wie bei RLC s; im Operationscode 000 durch d. umrandeten Bits ersetzen
RRC s	$\boxed{7 \rightarrow 0} \leftarrow \boxed{C}$ s=r, M (IX+d), (IY+d)	‡ ‡	P	‡	0 0	0	$\boxed{001}$				
RR s	$\boxed{7 \rightarrow 0} \leftarrow \boxed{C}$ s=r, M (IX+d), (IY+d)	‡ ‡	P	‡	0 0	0	$\boxed{011}$				

Assembler Sprache	symbolische Operation	Flags					Operationscode	Bytes	M-Zyklen	Takte	Bemerkung	
		C	Z	P	V	S						N
SLA s	$C \leftarrow \overline{r} \oplus 0$ s=r, M (IX+d), (IY+d)	0	0	1	0	0	0	100				
SRA s	$\overline{r} \oplus 0 \leftarrow C$ s=r, M (IX+d), (IY+d)	0	0	1	0	0	0	101				
SRL s	$0 \leftarrow \overline{r} \oplus C$ s=r, M (IX+d), (IY+d)	0	0	1	0	0	0	111				
RLD		0	0	1	0	0	0	11 101 101 01 101 111	2	5	18	zyklische Zahlenverschiebung links u. rechts zwischen AC u. M. Inhalt d. oberen Hälfte d. AC wird nicht beeinflusst
RRD		0	0	1	0	0	0	11 101 101 01 100 111	2	5	18	

Gruppe Bit setzen, löschen und testen

Assembler Sprache	symbolische Operation	Flags					Operationscode	Bytes	M-Zyklen	Takte	Register	
		C	Z	P	V	S					N	H
BIT b,r	$Z \leftarrow \overline{r}_b$	0	0	1	0	0	1	11 001 011 01 b r	2	2	8	000 E 001 C 010 D 011 E 100 H 101 L 111 A
BIT b,M	$Z \leftarrow \overline{M}_b$	0	0	1	0	0	1	11 001 011 01 b 110	2	3	12	
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	0	0	1	0	0	1	11 011 101 11 001 011 - d - 01 b 110	4	5	20	
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	0	0	1	0	0	1	11 111 101 11 001 011 - d - 01 b 110	4	5	20	b getestet Bit
SET b,r	$r_b \leftarrow 1$	.	.	.	.	.	.	11 001 011 11 b r	2	2	8	000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b,M	$M_b \leftarrow 1$	.	.	.	.	.	.	11 001 011 11 b 110	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	.	.	.	.	.	.	11 011 101 11 001 011 - d - 11 b 110	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	.	.	.	.	.	.	11 111 101 11 001 011 - d - 11 b 110	4	6	23	
RES b,s	$s_b \leftarrow 0$ s=r, M (IX+d), (IY+d)	.	.	.	.	.	.	10				Zur Bildung d. neuen Operationscodes 11 in SET durch 10 ersetzen. Flags u. Zeiten wie bei SET



Befehlsgruppe: Sprünge

Assembler Sprache	symbolische Operation	Flags C Z P/V S N H	Operations-code 76 543 210	Bytes	M-Zyk-len	Tak-te	Bemerkung
JMP nn	PC←nn	. . . . .	11 000 011 - n - - n -	3	3	10	
JPcc nn	wenn Bedin- gung cc wahr ist, PC←nn, sonst weiter	. . . . .	11 cc 010 - n - - n -	3	3	10	cc Bedingung 000 NZ nicht Null 001 Z Null 010 NC kein Übertrag 011 C Übertrag 100 PO unpar. 101 PE paarig 110 P Vorzeich. positiv 111 M Vorzeich. negativ
JR e	PC←PC+e	. . . . .	00 011 000 - e-2 -	2	3	12	
JRC e	wenn C=0 kein Sprung wenn C=1 PC←PC+e	. . . . .	00 111 000 - e-2 -	2	2	7	Beding. nicht erfüllt
				2	3	12	Beding. erfüllt.
JRNC e	wenn C=1 kein Sprung wenn C=0 PC←PC+e	. . . . .	00 110 000 - e-2 -	2	2	7	Beding. nicht erfüllt
				2	3	12	Beding. erfüllt.
JRZ e	wenn Z=0, kein Sprung wenn Z=1 PC←PC+e	. . . . .	00 101 000 - e-2 -	2	2	7	Beding. nicht erfüllt
				2	3	12	Beding. erfüllt
JRNZ e	wenn Z=1, kein Sprung wenn Z=0 PC←PC+e	. . . . .	00 100 000 - e-2 -	2	2	7	Beding. nicht erfüllt
				2	3	12	Beding. erfüllt
JMP M	PC←M	. . . . .	11 101 001	1	1	4	
JMP (IX)	PC←IX	. . . . .	11 011 101 11 101 001	2	2	8	
JMP (IY)	PC←IY	. . . . .	11 111 101 11 101 001	2	2	8	
DJNZ e	B←B-1 wenn B=0, kein Sprung wenn B≠0 PC←PC+e	. . . . .	00 010 000 - e-2 -	2	2	8	wenn B=0
				2	3	13	wenn B≠0

Befehlsgruppe: Unterprogrammaufruf und Rücksprung

CALL nn	(SP-1)←PCH (SP-2)←PCL PC←nn SP←SP-2	. . . . .	11 001 101 - n - - n -	3	5	17	
CA cc nn	wenn Bed. cc falsch ist, kein Sprung, sonst wie CALL nn	. . . . .	11 cc 100 - n - - n -	3	3	10	wenn cc falsch ist
				3	5	17	wenn cc wahr ist
RET	PCL←(SP) PCH←(SP+1) SP←SP+2	. . . . .	11 001 001	1	3	10	
Rcc	wenn Bed. cc falsch ist, kein Sprung, sonst wie RET	. . . . .	11 cc 000	1	1	5	wenn cc falsch ist
				1	3	11	wenn cc wahr ist

Assembler Sprache	symbolische Operation	Flags C Z P/V S N H	Operationscode 76 543 210	Bytes	M-Zyklen	Takte	Bemerkung
RETI	Rücksprung v. Interrupt	. . . . .	11 101 101 01 001 101	2	4	14	cc Bedingung 000 NZ nicht Null
RETH	Rücksprung v. nicht maskierb. Interrupt	. . . . .	11 101 101 01 000 101	2	4	14	001 Z Null 010 NC kein Übertrag 011 C Übertrag
RST p	(SP-1)←PCH (SP-2)←PCL PCH←0 PCL←p SP←SP-2	. . . . .	11 t 111	1	3	11	100 PO unpaar. 101 PE paarig 110 P Vorz.pos. 111 M Vorz.neg.
							t   p 000 00H 001 08H 010 10H 011 18H 100 20H 101 28H 110 30H 111 38H

Befehlsgruppe: Ein- und Ausgabe

IN n	A←(n)	. . . . .	11 011 011 - n -	2	3	11	n zu AO - A 7 AC zu AB - A15
IN r	r←(C)	. † P † 0 0	11 101 101 01 r 000	2	3	12	C zu AO - A 7 B zu AB - A15
INI	M←(C) B←B-1 HL←HL+1	. † X X 1 X a	11 101 101 10 100 010	2	4	16	C zu AO - A 7 B zu AB - A15
INIR	M←(C) B←B-1 HL←HL+1 Wiederholg. bis B=0	. 1 X X 1 X	11 101 101 10 110 010	2	5 B≠0 4 B=0	21 16	C zu AO - A 7 B zu AB - A15
IND	M←(C) B←B-1 HL←HL-1	. † X X 1 X a	11 101 101 10 101 010	2	4	16	C zu AO - A 7 B zu AB - A15
INDR	M←(C) B←B-1 HL←HL-1 Wiederholg. bis B=0	. 1 X X 1 X	11 101 101 10 111 010	2	5 B≠0 4 B=0	21 16	C zu AO - A 7 B zu AB - A15
OUT n	(n)←A	. . . . .	11 010 011 - n -	2	3	11	n zu AO - A 7 AC zu AB - A15
OUT r	(C)←r	. . . . .	11 101 101 01 r 001	2	3	12	C zu AO - A 7 B zu AB - A15
OUTI	(C)←M B←B-1 HL←HL+1	. † X X 1 X a	11 101 101 10 100 011	2	4	16	C zu AO - A 7 B zu AB - A15
OTIR	(C)←M B←B-1 HL←HL+1 Wiederholg. bis B=0	. 1 X X 1 X	11 101 101 10 110 011	2	5 B≠0 4 B=0	21 16	C zu AO - A 7 B zu AB - A15
OUTD	(C)←M B←B-1 HL←HL-1	. † X X 1 X a	11 101 101 10 101 011	2	4	16	C zu AO - A 7 B zu AB - A15
OTDR	(C)←M B←B-1 HL←HL-1 Wiederholg. bis B=0	. 1 X X 1 X	11 101 101 10 111 011	2	5 B≠0 4 B=0	21 16	C zu AO - A 7 B zu AB - A15

Befehlsliste  
des USGO D -  
sortiert nach  
dem OP-Code

OP-Code	Mnemonic
00	HOP
01 nn	LD BC, nn
02	LD (BC), A
03	INC BC
04	INC B
05	DEC B
06 n	LD B, n
07	RLCA
08	RIAF
09	ADD HL, BC
0A	LD A, (BC)
0B	DEC BC
0C	INC C
0D	DEC C
0E n	LD C, n
0F	RRCA
10 e	DJNZ e
11 nn	LD DE, nn
12	LD (DE), A
13	INC DE
14	INC D
15	DEC D
16 n	LD D, n
17	RLA
18 e	JR e
19	ADD HL, DE
1A	LD A, (DE)
1B	DEC DE
1C	INC E
1D	DEC E
1E n	LD E, n
1F	RRA
20 e	JRNC e
21 nn	LD HL, nn
22 nn	LD (nn), HL
23	INC HL
24	INC H
25	DEC H
26 n	LD H, n
27	DAA
28 e	JRZ e
29	ADD HL, HL
2A nn	LD HL, (nn)
2B	DEC HL
2C	INC L
2D	DEC L
2E n	LD L, n
2F	CPL
30 e	JRNC e
31 nn	LD SP, nn
32 nn	LD (nn), A
33	INC SP
34	INC M
35	DEC M
36 n	LD M, n
37	SCF
38 e	JRZ e
39	ADD HL, SP
3A nn	LD A, (nn)
3B	DEC SP
3C	INC A
3D	DEC A
3E n	LD A, n
3F	CCF
40	LD B, B
41	LD B, C
42	LD B, D
43	LD B, E
44	LD B, H
45	LD B, L
46	LD B, M

OP-Code	Mnemonic
47	LD B, 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
4E	LD C, M
4F	LD C, 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
56	LD D, M
57	LD D, A
58	LD E, B
59	LD E, C
5A	LD E, D
5B	LD E, E
5C	LD E, H
5D	LD E, L
5E	LD E, M
5F	LD E, 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
66	LD H, M
67	LD H, 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
6E	LD L, M
6F	LD L, A
70	LD M, B
71	LD M, C
72	LD M, D
73	LD M, E
74	LD M, H
75	LD M, L
76	HALT
77	LD M, A
78	LD A, B
79	LD A, C
7A	LD A, D
7B	LD A, E
7C	LD A, H
7D	LD A, L
7E	LD A, M
7F	LD A, A
80	ADD B
81	ADD C
82	ADD D
83	ADD E
84	ADD H
85	ADD L
86	ADD M
87	ADD A
88	ADC B
89	ADC C
8A	ADC D
8B	ADC E
8C	ADC H
8D	ADC L
8E	ADC M
8F	ADC A
90	SUB B
91	SUB C
92	SUB D
93	SUB E

OP-Code	Mnemonic
94	SUB H
95	SUB L
96	SUB M
97	SUB A
98	SBC B
99	SBC C
9A	SBC D
9B	SBC E
9C	SBC H
9D	SBC L
9E	SBC M
9F	SBC A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
A6	AND M
A7	AND A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR M
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR M
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP M
BF	CP A
CO	RNZ
C1	POP BC
C2 nn	JPHZ nn
C3 nn	JMZ nn
C4 nn	CANZ nn
C5	PUSH BC
C6 n	ADD A, n
C7	RST 0
C8	RZ
C9	RET
CA nn	JPZ nn
CC nn	CAZ nn
CD nn	CALL nn
CE n	ADC n
CF	RST 8
DO	RNC
D1	POP DE
D2 nn	JPNC nn
D3	OUT n
D4 nn	CANZ nn
D5	PUSH DE
D6 n	SUB n
D7	RST 10H
D8	RC
D9	EXX
DA nn	JPC nn
DB nn	IN A, n
DC nn	CAC nn
DE n	SBC n
DF	RST 18H
E0	RPO
E1	POP HL
D2 nn	JPPO nn

OP-Code	Mnemonic
FDCB d86	RES 6, (IY+d)
FDCB d8E	RES 7, (IY+d)
FDCB dC6	SET 0, (IY+d)
FDCB dC8	SET 1, (IY+d)
FDCB dD6	SET 2, (IY+d)
FDCB dDE	SET 3, (IY+d)
FDCB dE6	SET 4, (IY+d)
FDCB dEE	SET 5, (IY+d)
FDCB dF6	SET 6, (IY+d)
FDCB dFE	SET 7, (IY+d)

Befehlsliste - sortiert  
nach der Mnemonik

8E	ADC A, (HL)
DB8E d	ADC (IX+d)
FD8E d	ADC (IY+d)
8F	ADC A
88	ADC B
89	ADC C
8A	ADC D
8B	ADC E
8C	ADC H
8D	ADC L
CE n	ADC n
EDAA	ADC HL, BC
ED5A	ADC HL, DE
ED6A	ADC HL, HL
ED7A	ADC HL, SP
86	ADD M
DD86 d	ADD (IX+d)
FD86 d	ADD (IY+d)
87	ADD A
80	ADD B
81	ADD C
82	ADD D
83	ADD E
84	ADD H
85	ADD L
C6 n	ADD 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
F009	ADD IY, BC
FD19	ADD IY, DE
FD29	ADD IY, IY
FD39	ADD IY, SP
A6	AND M
DDA6 d	AND (IX+d)
FDA6 d	AND (IY+d)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E6 n	AND n
CB46	BIT 0, M
DDCB d46	BIT 0, (IX+d)
FDCB d46	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

OP-Code	Mnemonic
CB4E	BIT 1, M
DDCB d4E	BIT 1, (IX+d)
FDCB d4E	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, M
DDCB d56	BIT 2, (IX+d)
FDCB d56	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, M
DDCB d5E	BIT 3, (IX+d)
FDCB d5E	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, M
DDCB d66	BIT 4, (IX+d)
FDCB d66	BIT 4, (IY+d)
CB67	BIT 4, A
CB60	BIT 4, B
CB61	BIT 4, C
CB62	BIT 4, D
CB63	BIT 4, E
CB64	BIT 4, H
CB65	BIT 4, L
CB6E	BIT 5, M
DDCB d66	BIT 5, (IX+d)
FDCB d6E	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, M
DDCB d76	BIT 6, (IX+d)
FDCB d76	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, M
DDCB d7E	BIT 7, (IX+d)
FDCB d7E	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
DC nn	CAC nn
FC nn	CAM nn
D4 nn	CANZ nn
CD nn	GALL nn
C4 nn	CANZ nn
F4 nn	CAP nn
EC nn	CAPE nn

OP-Code	Mnemonic
E4 nn	CAPO nn
CAZ nn	CAZ nn
3F	CGP
RE	CMP M
DDRE d	CMP (IX+d)
FDRE d	CMP (IY+d)
B8	CMP A
B9	CMP B
BB	CMP C
BA	CMP D
BB	CMP E
BC	CMP H
BD	CMP L
FE n	CMP n
EDA9	CPD
EDB9	CPDR
EDA1	CPI
EDB1	CPIR
2F	CPL
27	DAZ
DD35 d	DEC (IX+d)
FD35 d	DEC (IY+d)
3D	DEC A
05	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD2B	DEC IX
FD2B	DEC IY
2D	DEC L
3B	DEC SP
F3	DI
10 e	DJNZ e
FB	EI
E3	EI (SP), HL
DDE3	EI (SP), IX
FDE3	EI (SP), IY
08	EXAF
EB	EX DE, HL
D9	EXX
76	HALT
ED46	IM0
ED56	IM1
ED5E	IM2
ED78	IN A
DE n	IN n
ED40	IN B
ED48	IN C
ED50	IN D
ED58	IN E
ED60	IN H
ED68	IN L
34	INC M
DD34 d	INC (IX+d)
FD34 d	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
EDAA	IND
EDBA	INDR
ED70	INP
EDA2	INI

OP-Code	Mnemonic
EDB2	INIR
E9	JMP M
DDE9	JMP (IX)
FDE9	JMP (IY)
DA nn	JPC nn
FA nn	JPM nn
D2 nn	JPHC nn
C3 nn	JMP nn
C2 nn	JPMZ nn
F2 nn	JPP nn
EA nn	JPPE nn
E2 nn	JPPO nn
CA nn	JPZ nn
38 e	JRC e
18 e	JR e
30 e	JRNC e
20 e	JRNZ e
28 e	JRZ e
02	LD (BC),A
12	LD (DE),A
77	LD M,A
70	LD M,B
71	LD M,C
72	LD M,D
73	LD M,E
74	LD M,H
75	LD M,L
36 n	LD M,n
DD77 d	LD (IX+d),A
DD70 d	LD (IX+d),B
DD71 d	LD (IX+d),C
DD72 d	LD (IX+d),D
DD73 d	LD (IX+d),E
DD74 d	LD (IX+d),H
DD75 d	LD (IX+d),L
DD36 dn	LD (IX+d),n
FD77 d	LD (IY+d),A
FD70 d	LD (IY+d),B
FD71 d	LD (IY+d),C
FD72 d	LD (IY+d),D
FD73 d	LD (IY+d),E
FD74 d	LD (IY+d),H
FD75 d	LD (IY+d),L
FD36 dn	LD (IY+d),n
32 nn	LD (nn),A
ED43 nn	LD (nn),BC
ED53 nn	LD (nn),DE
22 nn	LD (nn),HL
DD22 nn	LD (nn),IX
FD22 nn	LD (nn),IY
ED73 nn	LD (nn),SP
0A	LD A,(BC)
1A	LD A,(DE)
7E	LD A,(HL)
DD7E d	LD A,(IX+d)
FD7E d	LD A,(IY+d)
3A nn	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
3E n	LD A,n
ED5F	LD A,R
46	LD B,M
DD46 d	LD B,(IX+d)
FD46 d	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

OP-Code	Mnemonic
06 n	LD B,n
ED48 nn	LD BC,(nn)
01 nn	LD BC,nn
4E	LD C,M
DD4E d	LD C,(IX+d)
FD4E d	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
0E n	LD C,n
56	LD D,M
DD56 d	LD D,(IX+d)
FD56 d	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
16 n	LD D,n
ED5B nn	LD DE,(nn)
11 nn	LD DE,nn
5E	LD E,M
DD5E d	LD E,(IX+d)
FD5E d	LD E,(IY+d)
5F	LD E,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
1E n	LD E,n
66	LD H,M
DD66 d	LD H,(IX+d)
FD66 d	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
26 n	LD H,n
2A nn	LD HL,(nn)
21 nn	LD HL,nn
ED47	LD I,A
DD2A nn	LD IX,(nn)
DD21 nn	LD IX,nn
FD2A nn	LD IY,(nn)
FD21 nn	LD IY,nn
6E	LD L,M
DD6E d	LD L,(IX+d)
FD6E d	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
2E n	LD L,n
ED4F	LD R,A
ED7B nn	LD SP,(nn)
F9	LD SP,HL
DDF9	LD SP,IX
FDf9	LD SP,IY
31 nn	LD SP,nn
EDAB	LDD
EDBB	LDDR
EDAO	LDI
EDBO	LDIR

OP-Code	Mnemonic
ED44	NEG
00	NOP
86	OR M
DDB6 d	OR (IX+d)
FDB6 d	OR (IY+d)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
P6 n	OR n
EDBB	OTDR
EDB3	OTIR
ED79	OUT A
ED41	OUT B
ED49	OUT C
ED51	OUT D
ED59	OUT E
ED61	OUT H
ED69	OUT L
D3 n	OUT n
EDAB	OUTD
EDA3	OUTI
F1	POP AF
C1	POP BC
D1	POP DE
E1	POP HL
DD81	POP IX
FD81	POP IY
P5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DD85	PUSH IX
FD85	PUSH IY
CB86	RES 0,M
DDCB d86	RES 0,(IX+d)
FDcB d86	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 0,M
DDCB d8E	RES 1,(IX+d)
FDcB d8E	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,M
DDCB d96	RES 2,(IX+d)
FDcB d96	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 2,M
DDCB d9E	RES 3,(IX+d)
FDcB d9E	RES 3,(IY+d)
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
CB9E	RES 3,M

OP-Code	Mnemonic
DDCB dA6	RES 4, (IX+d)
FDCB dA6	RES 4, (IY+d)
CBA7	RES 4, A
CBA0	RES 4, B
CBA1	RES 4, C
CBA2	RES 4, D
CBA3	RES 4, E
CBA4	RES 4, H
CBA5	RES 4, L
CBAE	RES 5, M
DDCB dAE	RLS 5, (IX+d)
FDCB dAE	RES 5, (IY+d)
CBAF	RES 5, A
CBA8	RES 5, B
CBA9	RES 5, C
CBA A	RES 5, D
CBA B	RES 5, E
CBAC	RES 5, H
CBAD	RES 5, L
CBBE	RES 6, M
DDCB dB6	RES 6, (IX+d)
FDCB dB6	RES 6, (IY+d)
CBB7	RES 6, A
CBB0	RES 6, B
CBB1	RES 6, C
CBB2	RES 6, D
CBB3	RES 6, E
CBB4	RES 6, H
CBB5	RES 6, L
CBBE	RES 7, M
DDCB dBE	RES 7, (IX+d)
FDCB dBE	RES 7, (IY+d)
CBBF	RES 7, A
CBB6	RES 7, B
CBB9	RES 7, C
CBBA	RES 7, D
CBBB	RES 7, E
CBBC	RES 7, H
CBBD	RES 7, L
C9	RET
D6	RC
F6	RM
DO	RMC
GO	RNZ
FO	RP
EB	RPE
BO	RPO
CB	RB
ED4D	RTI
ED4E	RTM
CB16	RL M
DDCB d16	RL (IX+d)
FDCB d16	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 M
DDCB d06	RLC (IX+d)
FDCB d06	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 M
DDCB d1E	RR (IX+d)
FDCB d1E	RR (IY+d)
CB1F	RR A

OP-Code	Mnemonic
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
1F	RRA
CBOE	RRC M
DDCB d0E	RRC (IX+d)
FDCB d0E	RRC (IY+d)
CBOF	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 0
D7	RST 10H
DP	RST 18H
E7	RST 20H
EP	RST 28H
FP	RST 30H
FP	RST 38H
CP	RST 8
9E	SBC M
DD9E d	SBC (IX+d)
FD9E d	SBC (IY+d)
9P	SBC A
98	SBC B
99	SBC C
9A	SBC D
9B	SBC E
9C	SBC H
9D	SBC L
DE n	SBC n
ED42	SBC HL, BC
ED52	SBC HL, DE
ED62	SBC HL, HL
ED72	SBC HL, SP
37	SCF
CB06	SET 0, M
DDCB dC6	SET 0, (IX+d)
FDCB dC6	SET 0, (IY+d)
CB07	SET 0, A
CB00	SET 0, B
CB01	SET 0, C
CB02	SET 0, D
CB03	SET 0, E
CB04	SET 0, H
CB05	SET 0, L
CB0E	SET 1, M
DDCB dCE	SET 1, (IX+d)
FDCB dCE	SET 1, (IY+d)
CB0F	SET 1, A
CB0C	SET 1, B
CB09	SET 1, C
CB0A	SET 1, D
CB0B	SET 1, E
CB0D	SET 1, H
CB0E	SET 1, L
CB06	SET 2, M
DDCB dD6	SET 2, (IX+d)
FDCB dD6	SET 2, (IY+d)
CB07	SET 2, A
CB00	SET 2, B
CB01	SET 2, C
CB02	SET 2, D
CB03	SET 2, E
CB04	SET 2, H
CB05	SET 2, L
CB0E	SET 3, B
CB0E	SET 2, M
DDCB dDE	SET 3, (IX+d)
FDCB dDE	SET 3, (IY+d)

OP-Code	Mnemonic
CBDF	SET 3, A
CB09	SET 3, C
CBDA	SET 3, D
CBDB	SET 3, E
CBDC	SET 3, H
CBDD	SET 3, L
CBE6	SET 4, M
DDCB dE6	SET 4, (IX+d)
FDCB dE6	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
CBEE	SET 5, M
DDCB dEE	SET 5, (IX+d)
FDCB dEE	SET 5, (IY+d)
CBEF	SET 5, A
CBEB	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, M
DDCB dF6	SET 6, (IX+d)
FDCB dF6	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, M
DDCB dFE	SET 7, (IX+d)
FDCB dFE	SET 7, (IY+d)
CBFF	SET 7, A
CBFB	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 M
DDCB d26	SLA (IX+d)
FDCB d26	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 M
DDCB d2E	SRA (IX+d)
FDCB d2E	SRA (IY+d)
CB2F	SRA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB2E	SRL M
DDCB d3E	SRL (IX+d)
FDCB d3E	SRL (IY+d)
CB3F	SRL A
CB3A	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB M

OP-Code	Mnemonic
DD 96d	SUB (IX+d)
FD 96 d	SUB (IY+d)
97	SUB A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
D6 n	SUB n
AE	XOR M

OP-Code	Mnemonic
DDA Ed	XOR (IX+d)
FDA Ed	XOR (IY+d)
AP	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE n	XOR n

### Programmierung der PIO

#### Interruptrektor

V7 V6 V5 V4 V3 V2 V1 0

#### Betriebsartenauswahl

M1 M0 X X 1 1 1 1

#### M1 M0 Betriebsart

0 0 Byteausgabe  
0 1 Byteeingabe  
1 0 Byte-Ein-/Ausgabe  
1 1 Bit-Ein-/Ausgabe

#### Ein-/Ausgabe Wahlwort, wenn Bit-Ein-/Ausgabe

I07 I06 I05 I04 I03 I02 I01 I00

#### Interruptkontrollwert

EI A/O H/L MF 0 111

EI Interruptfreigabe

A/O AND bzw. OR-Verknüpfung

H/L High oder Low-Pegel der Port

MF Maskierungswort folgt als nächstes Steuerwort

#### Maskierungswort, wenn MF=1

MB7 MB6 MB5 MB4 MB3 MB2 MB1 MBO

#### Interruptkontrollwort ohne Auswahlinformation

EI X X X 0011

### Programmierung der CTC

#### Interruptrektor

V7 V6 V5 V4 V3 X X 0

#### Kanalsteuerwort

EI M V F T TC R 1

EI Interruptfreigabe T Triggerzeitpunkt

M Betriebsart TC Zeitkonstante folgt

V Verteilerfaktor R Rücksetzen

F Triggerflanke

#### Zeitkonstantenwort, wenn TC=1 im Kanalsteuerwort

ZK7 ZK6 ZK5 ZK4 ZK3 ZK2 ZK1 ZK0



Dezimal-Hexadezimal-Tabelle von -128 bis +127  
negative Zahlen als 2er Komplement

DEZ	HEX	CPL	DEZ	HEX	CPL	DEZ	HEX	CPL
0	00	00	44	2C	D4	88	58	A8
1	01	FF	45	2D	D3	89	59	A7
2	02	FE	46	2E	D2	90	5A	A6
3	03	FD	47	2F	D1	91	5B	A5
4	04	FC	48	30	D0	92	5C	A4
5	05	FB	49	31	CF	93	5D	A3
6	06	FA	50	32	CE	94	5E	A2
7	07	F9	51	33	CD	95	5F	A1
8	08	F8	52	34	CC	96	60	A0
9	09	F7	53	35	CB	97	61	9F
10	0A	F6	54	36	CA	98	62	9E
11	0B	F5	55	37	C9	99	63	9D
12	0C	F4	56	38	C8	100	64	9C
13	0D	F3	57	39	C7	101	65	9B
14	0E	F2	58	3A	C6	102	66	9A
15	0F	F1	59	3B	C5	103	67	99
16	10	F0	60	3C	C4	104	68	98
17	11	EF	61	3D	C3	105	69	97
18	12	EE	62	3E	C2	106	6A	96
19	13	ED	63	3F	C1	107	6B	95
20	14	EC	64	40	C0	108	6C	94
21	15	EB	65	41	BF	109	6D	93
22	16	EA	66	42	BE	110	6E	92
23	17	E9	67	43	BD	111	6F	91
24	18	E8	68	44	BC	112	70	90
25	19	E7	69	45	BB	113	71	8F
26	1A	E6	70	46	BA	114	72	8E
27	1B	E5	71	47	B9	115	73	8D
28	1C	E4	72	48	B8	116	74	8C
29	1D	E3	73	49	B7	117	75	8B
30	1E	E2	74	4A	B6	118	76	8A
31	1F	E1	75	4B	B5	119	77	89
32	20	E0	76	4C	B4	120	78	88
33	21	DF	77	4D	B3	121	79	87
34	22	DE	78	4E	B2	122	7A	86
35	23	DD	79	4F	B1	123	7B	85
36	24	DC	80	50	B0	124	7C	84
37	25	DB	81	51	AF	125	7D	83
38	26	DA	82	52	AE	126	7E	82
39	27	D9	83	53	AD	127	7F	81
40	28	D8	84	54	AC	128	80	80
41	29	D7	85	55	AB			
42	2A	D6	86	56	AA			
43	2B	D5	87	57	A9			

Tabelle der Hexadezimalwerte

6		5		4		3		2		1	
HEX = DEC		HEX = DEC		HEX = DEC		HEX = DEC		HEX = DEC		HEX = DEC	
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
0123		4567		0123		4567		0123		4567	
BYTE				BYTE				BYTE			

Potenzen von 2

$2^n$	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

Potenzen von 16

$16^n$	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

## A

---

Adreßbus	‡ Bus zur Übertragung von Adressen. Die Anzahl der Leitungen entspricht dem maximal adressierbaren Speicherbereich (16 Bit für 64 K Speicherplätze)
Adreßregister	‡ Register eines Mikrocomputers, dessen Inhalt zur Berechnung einer Operandenadresse benutzt wird.
Adreßteil	Teil des Befehlswortes. Der A. enthält die absolute ‡ Adresse einer Information oder Angabe darüber, wie die absolute Adresse zu bilden ist (z. B. durch Summation von Registerinhalten).
Adresse	<ol style="list-style-type: none"><li>1. Adresse einer Information ist die Nummer des Speicherplatzes, auf dem diese Information untergebracht ist.</li><li>2. Statt der Nummer wird ein aus alphanumerischen Zeichen gebildetes Wort benutzt (symbolische Adresse).</li></ol>
Adressierungsart	Art und Weise, wie die ‡ Adresse eines Speicherplatzes in einem Programm dargestellt bzw. wie vom Adreßrechenwerk die Adresse eines Speicherplatzes aus der Operandenangabe im ‡ Maschinenbefehl errechnet wird.
Akkumulator	‡ Register, mit dessen Inhalt Operationen durchgeführt werden können. Der A. repräsentiert meist einen Operanden, das

Ergebnis der Operation steht wiederum im A. Jeder Mikroprozessor hat mindestens einen A., der für alle Datenübertragungen über den Datenbus vom und zum Prozessor herangezogen wird.

- ALU (Arithmetical Logical Unit)  
Arithmetisch-logische Einheit  
Funktionsblock des Rechenwerkes des Mikrorechners. Sie führt arithmetische und logische Operationen und Verschiebeoperationen aus. Sie verfügt über ein Adresswerk und kann Flags belegen.
- Anwenderprogramm Spezielles Programm, durch das ein Anwenderproblem gelöst wird.
- Anweisung Abgeschlossene Vorschrift, die in einer beliebigen Programmiersprache formuliert ist.
- Arbeitsweise, asynchrone
1. Ablauf von zwei oder mehreren voneinander unabhängigen Prozessen (Vorgängen), die so gestaltet sind, daß sie zu keinem Zeitpunkt aufeinander angewiesen sind.
  2. Eine Betriebsart bei der seriellen Datenübertragung, bei der die Start- und Stopkennzeichnungen (Signale) zur zeichenweisen Synchronisation der Datenübertragung benutzt werden. Dadurch können sich Verzerrungen über Frequenzabweichungen nicht über mehrere Zeichen summieren.

Arbeitsweise,  
synchrone

1. Ablauf von zwei oder mehreren voneinander unabhängigen Prozessen, die so gestaltet sind, daß zu bestimmten Zeitpunkten eine Synchronisation stattfindet (Einsatz für Datentransport).
2. Eine Betriebsart bei der seriellen Datenübertragung, bei der die Information mit einem konstanten Takt für den Kanal gegeben wird und vom Empfänger mit dem gleichen Takt (gleich in Frequenz und Phase) zu empfangen ist.

Arbeitsspeicher

↑Register oder ↑RAM für Programm und Zwischenergebnisse.

ASCII

(American Standard Code for Information Interchange)  
amerikanischer Standard-Code für Informationsaustausch,  
Bezeichnung eines USA-Standards, der international benutzt wird. Es ist ein 7-Bit-Code für 128 Zeichen (alle Zeichen, die eine Schreibmaschine besitzt und einige Sonderzeichen). Er wird zur Ausgabe dieser Zeichen auf periphere Geräte benutzt.

Assembler

Programm zur Übersetzung eines in einer ↑Assemblersprache geschriebenen Programmes in eine ↑Maschinensprache.  
Aus jedem Assemblerbefehl wird ein Maschinenbefehl erzeugt.

Assemblersprache

Maschinenorientierte Programmiersprache, die vom ↑Assembler in die ↑Maschi-

**B**

---

BASIC	(Beginners All-purpose Symbolic Introduction-Code) Allzweckprogrammiersprache für Anfänger. Leicht erlernbare höhere Programmiersprache. In einem Befehl werden mehrere Maschinenbefehle zusammengefaßt realisiert.
BCD-Arithmetik	(Binary Coded Decimals-Arithmetik) binär kodierte Dezimalzahlen-Arithmetik, Möglichkeit, bei arithmetischen Verknüpfungen mit ziffernweise kodierten Dezimalzahlen zu arbeiten. Jede Ziffer wird dual in einem Halbbyte (4Bit) dargestellt (BCD-Format).
bedingter Sprung	Ein Sprung, der ausgeführt wird, wenn im Programmverlauf bestimmte Bedingungen erfüllt sind.
Befehl	Anweisung an einen Rechner zur Ausführung einer Operation. Ein B. wird durch einen $\uparrow$ Assembler in den zugeordneten $\uparrow$ Maschinenbefehl übersetzt, der vom $\uparrow$ Mikroprozessor verarbeitet werden kann.
Befehlsadresse	$\uparrow$ Adresse, unter der ein $\uparrow$ Befehl im $\uparrow$ Speicher steht.
Befehlsregister	$\uparrow$ Register, das den Befehl zum Zwecke der Ausführung speichert.

Befehlsvorrat	Gesamtheit der ↑Befehle, die vom ↑Steu- erwerk eines Mikrorechners verarbeitet werden können.
Befehlszähler	Spezielles Register der ↑CPU, das wäh- rend der Abarbeitung eines Programmes stets die ↑Adresse des nächsten zu ver- arbeitenden Befehls enthält. Abkürzung: PC (Program Counter) Programm-Zähler
Betriebssystem	Programmpaket, das die Bearbeitung von Programmen durch den Rechner ohne mensch- liche Hilfe ermöglicht.
bidirektional	Übertragung von Daten auf einer Leitung kann in beiden Richtungen erfolgen.
Binärdarstellung	Darstellung von Informationen durch die Elemente 0 und 1 (Binärelemente)
Binärwort	Endliche Folge der Binärelemente 0 und 1
Bit	(Binary Digit) Binärziffer, 1. einzelnes Zeichen eines ↑Binärwortes 2. kleinste Einheit der Speicherkapazi- tät eines Speicherbausteines
Bus	Verbindungsleitung zwischen den Bau- gruppen eines Mikrorechners. Man unterscheidet: ↑Datenbus ↑Adreßbus ↑Steuerbus
Byte	Zusammenfassung von 8 ↑Bit zu einer Ein- heit. Ein Byte ist im Speicher die

kleinste adressierbare Einheit. Die Speicherkapazität wird in KByte angegeben.

---

-Flag (Carry-Flag)  
Übertragsflag;  
↑Flag, das mit 1 belegt wird, wenn bei Addition oder Subtraktion ein Übertrag über die höchstwertigste Stelle des Akkumulators erfolgt.

PU (Central Processing Unit)  
Zentrale Verarbeitungseinheit - ZVE;  
Steuert die Abarbeitung des Programmes und die Realisierung eines jeden ↑Maschinenbefehls.

TC (Counter/Timer Circuit)  
Zähler/Zeitgeber-Baustein;  
Spezieller Schaltkreis eines Mikrorechners, der zwei Funktionen übernehmen kann:

1. Zähler, der im Sinne eines Weckers Unterbrechungen (↑Interrupts) erzeugt.
2. Zeitgeber, der im Sinne einer Uhr Zeitangaben zur Verfügung stellen kann.

---

aisy-Chain      ↑Prioritätskaskade



Datenbus	Bus zur Übertragung von Daten zwischen den einzelnen Baugruppen des Mikrorechners. Die Anzahl der Leitungen entspricht der Anzahl der Bits eines Wortes (z. B. 4, 8 oder 16)
Datentransfer	Datentransport
Dekrementierung	Vermindern eines Zählers oder einer Programmgröße um 1
DMA-Baustein	(Direct Memory Access) Direkter Speicherzugriff, Schaltkreis, der sehr schnell und selbstständig die Suche nach einem Datenbyte oder Datenübertragungen zwischen Speicher und peripheren Geräten ausführen kann, ohne Umweg über die CPU. Dadurch wird die CPU von routinemäßigen Übertragungsaufgaben entlastet.

## E

---

Eingabe-Ausgabe-Baustein	Realisiert Datenfluß vom oder zum Mikroprozessor zur oder von der Peripherie.
EPROM	(Erasable Programmable Read-Only-Memory) löschbarer programmierbarer Nur-Lese-Speicher, Speichertyp, der zur Speicherung von festen Programmen und Konstanten dient. Er wird elektrisch programmiert und ist durch UV-Licht löschtbar.
Echtzeituhr	Baugruppe eines Mikroprozessorsystems zur Erzeugung periodischer Taktimpulse.

Die E. dient zum Steuern von Programmabläufen.

---

ag	Speicher der Länge ein Bit innerhalb der CPU. Sie werden benutzt, um beim Rechnerlauf aufgetretene Zustände (z. B. Überlauf, 0 im Akkumulator) zu fixieren. Der Inhalt der Flags kann in Sprungbefehlen abgefragt und so für Programmverzweigungen genutzt werden. Der Inhalt vom F. wird auch in andere Operationen einbezogen.
agregister	Register innerhalb der CPU. Das F. besteht aus 8 Flags.
ormat	Anordnung von Daten, Adressen oder Befehlen.

---

lbyte	Gruppe von 4 Bits Synonym: Tetrade In einem H. können die Zahlen 0 ... 15 in dualer Form dargestellt werden.
rdware	Gesamtheit aller technischen Einheiten eines Mikroprozessorsystems.
auptprogramm	Eigentliches Programm, in dem Unterprogramme aufgerufen werden können.
Flag	(Half-Carry-Flag)

t

### Halbbyte-Übertragsflag

♣Flag, das mit 1 belegt wird, wenn bei arithmetischen Operationen im Akkumulator ein Übertrag von Bit D3 auf Bit D4 auftritt.

**Hexadezimalsystem** In der Rechentechnik sehr häufig benutztes Zahlensystem mit der Basis 16.  
**Verwendete Symbole:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

**High** hoch,  
entspricht dem binären Zustand 1

## I

---

**Index-Register** ♣Register, dessen Inhalt vor oder während der Ausführung eines Befehls zum (oder vom) Adreßoperanden addiert (oder subtrahiert) werden kann.

**Inkrementierung** Erhöhen eines Zählers oder einer Programmgröße um 1.

**Interface** Schnittstelle, elektronische Schaltung, die zwei Geräte oder Bausteine einander anpaßt.

**Interrupt** In der Mikrorechentechnik benutzter Fachbegriff für Unterbrechung eines Programmes und Bearbeitung eines vorbereiteten Unterprogrammes (♣Interruptserviceroutine). Danach wird die Abarbeitung des unterbrochenen Programmes fortgesetzt.

rruptmode	Betriebsart der Bausteine eines Mikroprozessorsystems, durch die die Unterbrechungsbehandlung festgelegt wird. Zu jedem Interrupt gehört eine Interruptserviceroutine. Diese Routine (Unterprogramm) wird automatisch aufgesucht, wenn der Mikroprozessor die Interruptforderung annimmt.
rruptregister	Register, mit dessen Hilfe die Adresse einer Interruptserviceroutine (ISR) aufgebaut wird. Es enthält den höherwertigen Teil (A8 ... A15) eines Zeigers, der auf die Anfangsadresse der jetzt zu beginnenden ISR zeigt.
rruptservice- ine	Unterbrechungsbehandlungsroutine, Unterprogramm, das durch die CPU abgearbeitet wird, sobald ein Interrupt von einer peripheren Schaltung angenommen wurde
rruptvektor	Information der Länge 16 Bit, die zur Steuerung der Interruptbehandlung dient. Der I. wird vom interruptenden Peripherieschaltkreis ausgesendet. Der niederwertige Teil (A0 ... A7) bildet zusammen mit dem Inhalt des Interruptregisters den Zeiger auf die Anfangsadresse der ISR.

---

Abkürzung für "Kilo", Mengeneinheit für Anzahl von Speicherplätzen

1 KByte = 1024 Byte

**Kanal** Verbindung, über die Daten gesandt oder empfangen werden können (Tor)

**Kellerspeicher**  $\uparrow$ Stack

**Kellerzeiger**  $\uparrow$ Stackpointer

## L

---

**Low** tief,  
entspricht dem binären Zustand 0

## M

---

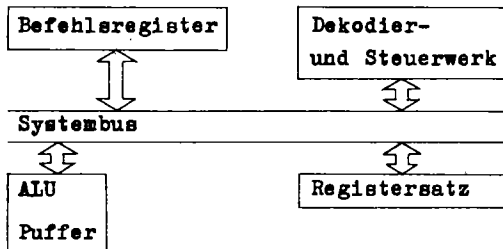
**Marke** Zeichen ("Name") zur Identifizierung einer Anweisung oder eines bestimmten Datenwortes zur Auffindung in einem Computerprogramm.

**Maschinenprogramm** Folge von  $\uparrow$ Befehlen in  $\uparrow$ Maschinsprache, die von einem Mikrorechner direkt gelesen und ausgeführt werden können.

**Maschinsprache** Menge von  $\uparrow$ Binärwörtern, die von einem Mikroprozessor unmittelbar verarbeitet werden können.

**Maske** Gruppe von Bits, mit denen durch logische  $\uparrow$  Befehle die zugeordneten Bits eines Operanden herausgelöst bzw. gesondert betrachtet werden können.

**Mikroprozessor** Zentraleinheit ( $\uparrow$ CPU) eines Mikrorechners mit folgendem schematischen Aufbau



oprozessor-  
sm

Informationsverarbeitungssystem, das  
im wesentlichen aus folgenden Grund-  
bausteinen besteht: †CPU

- †PIO
- †SIO
- †CTC
- †ROM
- †RAM
- †DMA

rechner

†Mikroprozessorsystem

onik

maschinenorientierte †Programmierspra-  
che, in der alphanumerische Abkürzungen  
zur Notation von †Befehlen verwendet  
werden. Sie werden durch den †Assembler  
in die †Maschinensprache übersetzt.

Beispiel: LD (HL),E ist der mnemonische  
Kode für einen Ladebefehl  
(Transport aus dem Register E  
in den Hauptspeicher).

Übersetzt in die Maschinensprache ent-  
steht das Bitmuster 01110011.

## N

---

**N-Flag**                    Additions-/Subtraktions-Flag  
♣Flag, das mit 1 belegt wird, wenn als  
letzter Befehl eine Subtraktion erfolgte,  
bei Addition N-Flag = 0.

## O

---

**Operandenadresse**       ♣Adresse, unter der ein Operand im  
♣Speicher steht.

## P

---

**Peripheriebaustein**      Spezieller Schaltkreis für den Anschluß  
externer (peripherer) Geräte an einen  
Mikroprozessor.

**PIO**                        (Parallel-Input/Output)  
Parallele Ein/Ausgabe,  
spezieller Schaltkreis eines Mikrorechners,  
der parallelen Datenaustausch zwischen dem  
Mikroprozessor und der Peripherie realisiert.  
Der PIO besitzt zwei Datenkanäle mit  
einer Breite von je 8 Bit = 1 Byte.

**Polling**                    Periodische Abfrage der ♣Peripheriebausteine,  
um festzustellen, ob eine ♣Interruptanforderung  
vorliegt.

**Port**                        1. Mit P. wird die Stelle bezeichnet,  
an der der ♣Bus mit dem Mikroprozessor  
(oder einem anderen Schaltkreis) verbunden  
ist. Man unterscheidet Datenport, Adressenport.

2. Zugriffsstelle für das Einlesen  
(Eingabeport) oder das Ausgeben  
(Ausgabeport) von Daten.  
Synonyme: Tor, Kanal.

- loritätskaskade (Daisy-Chain)  
Spezielle Behandlungsweise von Unterbrechungen, die von Peripheriebausteinen an die CPU gesandt werden. Die einzelnen Peripheriebausteine sind dabei fest in eine Vorrangstruktur eingebunden.
- ogramm Folge von \*Anweisungen und Vereinbarungen, die einen in sich abgeschlossenen Algorithmus darstellen.
- ogrammsähler (PC: Program Counter)  
\*Befehlszähler
- OM (Programmable Read Only Memory)  
programmierbarer Nur-Lese-Speicher, vom Programmierer mit einem speziellen Programmiergerät elektrisch programmierbarer Festwertspeicher, nicht löschar.
- udotetrade Halbbytebelegung (4 Bit) der \*BCD-Darstellung, die keiner Dezimalziffer entspricht (A, B, C, D, E, F).  
Sie kann durch Addition einer Sechs normalisiert werden (BCD-Korrektur).
- ffer Speicherbereiche zur kurzzeitigen Informationsspeicherung, z. B. zum Ausgleich unterschiedlicher Verarbeitungsgeschwindigkeiten miteinander arbei-



tender Einheiten.

P/V-Flag

(Parity-Overflow-Flag)

Paritäts-Überlauf-Flag,

Flag, das mit 1 belegt wird, wenn das Ergebnis bei logischen Operationen von gerader Parität ist, oder wenn bei arithmetischen Operationen ein Überlauf auftritt.

Q

---

Quellprogramm

Programmtext, der in einer höheren Programmiersprache oder in Assemblersprache vorliegt.

Das Q. muß durch einen Übersetzer oder Assembler in ein Maschinenprogramm umgesetzt werden.

Quittung

Nachricht zur Bestätigung des korrekten Eintreffens einer Nachricht beim Empfänger in einem Mikroprozessorsystem.

Quittungsbetriebs-  
logik

Technische Vorrichtung in Bausteinen von Rechenanlagen, die durch das Absetzen eines Signals die Beendigung einer Aufgabe mitteilen, bzw. die Bereitschaft zum Informationsaustausch bestätigen (Handshake).

R

---

RAM

(Random-Access-Memory)

Schreib-Lese-Speicher

Direktzugriffsspeicher für Lesen oder Schreiben, sie verlieren mit dem Ab-

schalten der Betriebsspannung ihre Informationen.

**Rechenwerk**

Bauteil der  $\mu$ CPU, in dem alle Datenmanipulationen ausgeführt werden. Zum R.  $\mu$ hlt die  $\mu$ ALU und der  $\mu$ Registersatz.

**Refresh-Register**

7-Bit-Speicher des  $\mu$ Registersatzes eines Mikroprozessors. Das R. dient zur zyklischen Auffrischung von dynamischen  $\mu$ RAM's. Es ist für die Programmierung ohne Bedeutung.

**Register**

Speicherstelle mit schnellem und direktem Zugriff, die zur vorübergehenden Aufnahme von Daten, Operanden oder Adressen dient. Diese werden bei der Verarbeitung von einem R. in ein anderes R. transportiert und verarbeitet.

**Registersatz**

Gesamtheit aller  $\mu$ Register eines Mikroprozessors.  
Registersatz des Schaltkreises U 880 D:

A	F
B	C
D	E
H	L

Hauptregister-  
satz  
(je 8 Bit)

A'	F'
B'	C'
D'	E'
H'	L'

Zweitregister-  
satz  
(je 8 Bit)

I	R	B,C,D,E,H,L	allg. 8-Bit-Register
IX		A	Akkumulator
IY		F	Flag-Register
SP		I	Interrupt-register
PC		R	Refresh-Register
Adres- senre- gister- satz		IX,IY	Indexregister
		SP	Stackpointer
		PC	Programmzähler

ROM

(Read-Only-Memory)

Nur-Lese-Speicher zur Speicherung von festen Programmen und Konstanten.

S

---

S-Flag

(Sign-Flag)

Vorzeichenflag

Flag, das mit 1 belegt wird, wenn auf Grund einer arithmetischen oder logischen Operation das höchstwertigste Bit des Akkumulators mit 1 belegt, der Inhalt des Akkumulator also negativ ist.

SIO

(Serial Input/Output)

Serieller Ein/Ausgabe-Baustein, Peripheriebaustein, der zum Anschluß von Geräten mit Bit-seriellem Datenaustausch dient (z. B. Fernschreiber).

Software

Gesamtheit der zu einem Mikrorechner gehörenden Programme, z. B. Assembler, Zugriffsroutinen zu externen Datenträgern, Anwenderprogramme.

peicher	<p>Bauelemente eines Mikrorechners zum Speichern von Programmen und Daten. Die Einheit des S., der Speicherplatz, heißt <math>\uparrow</math>Byte. Die Kapazität wird in KByte angegeben. Speichertypen: <math>\uparrow</math>RAM, <math>\uparrow</math>ROM, <math>\uparrow</math>PROM (interne Speicher)</p> <p>Es gibt auch externe Speicher, sie sind nicht direkt adressierbar (z. B.: Magnetband)</p>
peicheradresse	<p>Zahlenangabe zur Auffindung eines Speicherplatzes in einem <math>\uparrow</math>Speicher. Die Nummer eines Speicherplatzes ist die S. oder die absolute Adresse.</p>
tack	<p>Stapelspeicher, der zur Verwaltung ein eigenes Adreßregister (<math>\uparrow</math>Stackpointer) besitzt. Er dient zur Organisation von Unterprogramm-Verarbeitungen.</p> <p>Funktionsprinzip: Die zuletzt abgelegte Adresse wird als erste wieder entnommen, danach die vorletzte usw. Die Verwaltung erfolgt unabhängig von der Programmierung.</p>
ackpointer	<p>Kellerzeiger <math>\uparrow</math>Register oder Speicherplatz, dessen Inhalt die Adresse des Arbeitspunktes in einem <math>\uparrow</math>Stack ist.</p>
uerbus	<p>Leitungen zur Übertragung von Steuerungssignalen zwischen den einzelnen Schaltkreisen.</p>
ntax	<p>Menge von Regeln, durch die die Erzeugung gültiger Sätze einer Programmiersprache exakt definiert ist.</p>

**Systembus**                      Sammelleitung eines Mikrorechners, über die der Informationstransport zwischen der CPU und dem Arbeitsspeicher erfolgt.  
Wird die Übertragung auf unterschiedlichen Leitungen realisiert, unterscheidet man Datenbus, Adreßbus und Steuerbus.

## T

---

**Tri-State**                      Möglichkeit von 3 verschiedenen Zuständen: 0, 1 und ein hochohmiger Zustand

## U

---

**Unterprogramm**                      In sich abgeschlossenes Teilprogramm, das in einem Programm häufig auftretende Algorithmen beinhaltet. Man gelangt zum U. mit einem speziellen Sprungbefehl. Am Ende eines U. steht ein Rücksprungbefehl.

**User**                              Anwender

## W

---

**Wort**                              Anzahl von aufeinanderfolgenden Bits, die in einem Mikrorechner als Einheit betrachtet werden.

---

lag (Zero-Flag)  
Nullflag  
▲Flag, das mit 1 belegt wird, wenn das Ergebnis aus einer Operation 0 ist.

Zentrale Verarbeitungseinheit ↑ CPU

ierkomplement Wird benutzt, um in einem Byte eine negative Zahl darzustellen. Im höchstwertigsten Bit B7 steht das Vorzeichen:  
0 positiv  
1 negativ  
Das Z. wird gebildet, indem die Binärzahl bitweise negiert und anschließend eine 1 addiert wird.

itregistersatz Der Inhalt des Haupt-Registersatzes kann ganz oder teilweise in den Zweitregistersatz "kopiert" werden, um die Inhalte des Hauptregistersatzes zu retten bzw. diese für die Ausführung weiterer Operationen freizumachen.

e umfassende Erläuterung der hier aufgeführten und weiterer Begriffe der Mikrorechentechnik findet der Leser im d 206 der Reihe "Automatisierungstechnik", Gerhard lin: "Kleines Lexikon der Mikrorechentechnik".



Rs 440/85 WV/6/1-10 920







veb mikroelektronik · karl marx · erfurt  
stammbetrieb

DDR-5010 Erfurt, Rudolfstraße 47 · Telefon: 5 80, Telex: 061 306

**elektronik**  
**export·import**

Volkseigener Außenhandelsbetrieb der  
Deutschen Demokratischen Republik  
DDR - 1026 Berlin, Alexanderplatz 6  
Telex: BLN 114721 etel, Telefon: 2180