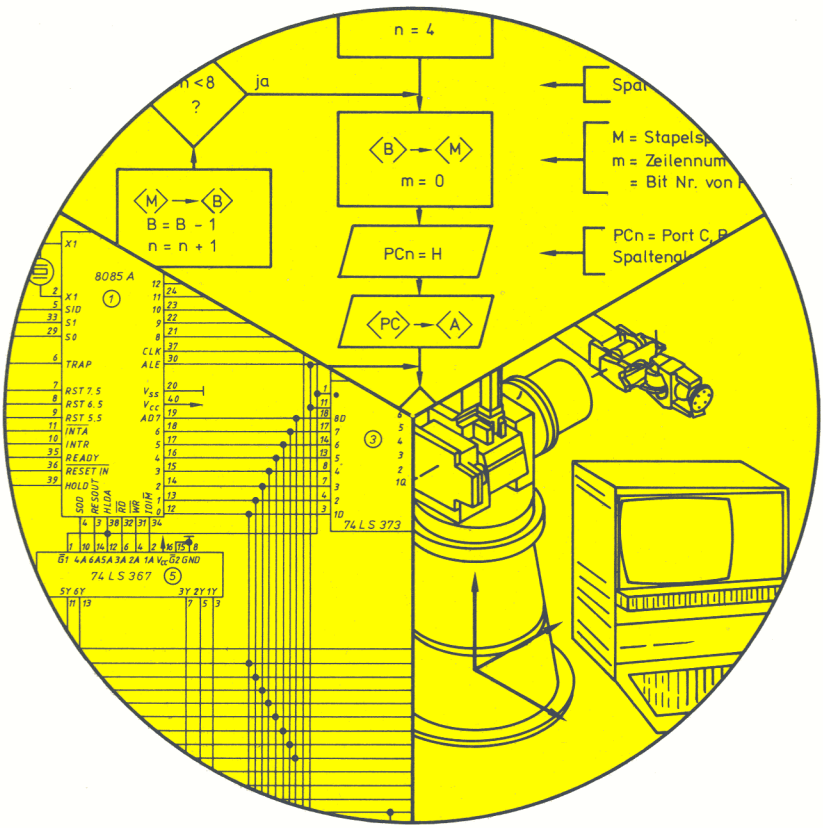
Dieter Hannemann

Einführung in die Mikrocomputer-Technik

Programmierung – Schaltungstechnik – Anwendung von Mikroprozessoren



Girardet

Dieter Hannemann

Einführung in die Mikrocomputer-Technik

Programmierung –
Schaltungstechnik –
Anwendung von
Mikroprozessoren

3., überarbeitete Auflage

Verlag W. Girardet · Essen

ISBN 3-7736-1024-6
Bestellnummer 10246

W. Girardet Buchverlag GmbH, Essen

Druck W. Girardet, Essen · Printed in Germany · 1984

Vertrieb: Cornelsen-Velhagen & Klasing Verlagsgesellschaft, Bielefeld

3. Auflage/2. Druck

Inhalt

Vorwort

Seit mehr als 10 Jahren gibt es diesen revolutionären Baustein, genannt Mikroprozessor. Die Auswirkungen dieser neuen Technologie haben weite Felder der industriellen Technik und in zunehmendem Maße auch private Bereiche erfaßt.

Es ist daher ein Anliegen Vieler, sich mit dem Gegenstand dieser Umwälzungen näher zu beschäftigen; einmal, um im Beruf weiter zu kommen, zum anderen aus allgemeinem Interesse oder im Verlauf eines Studiums. Ich hoffe, daß dieses Buch dem genannten Kreis der Interessierten eine gute Stütze zur Einarbeitung in die Mikrocomputertechnik bietet.

Langjährige Erfahrungen bei der Entwicklung von Mikrocomputersteuerungen sowie der Durchführung von Mikroprozessor-Seminaren und Vorlesungen, führten zu der hier angebotenen Stoffauswahl.

Die Beschränkung auf nur einen Mikroprozessor-Typ ist aus didaktischen Gründen zwingend, da sonst die Fülle der Unterschiede zwischen den einzelnen Mikroprozessor-Typen den Anfänger in unnötige Schwierigkeiten bringt.

Das Fachvokabular der Mikrocomputertechnik enthält englische Ausdrücke. In den meisten Fällen sind Übersetzungen ins Deutsche alternativ aufgeführt. Im allgemeinen Sprachgebrauch dieser Disziplin werden jedoch häufig die englischen Fachausdrücke benutzt.

Teil A des Buches beschreibt Grundlagen und wichtige Elemente für die Programmierung von Mikroprozessoren auf der Ebene der Maschinensprache. In diesem Abschnitt kann der Mikrocomputer noch als sog. 'black box' betrachtet werden. Zur Einübung der Programmiertechnik haben sich hier die verschiedenen Kleinstcomputer bewährt (siehe Bezugsquellennachweis).

Teil B soll den Leser befähigen, die elektronischen Vorgänge in einem Mikrocomputer zu verstehen und eigene Mikrocomputerschaltungen zu entwerfen oder vorhandene zu erweitern.

Teil C schließlich zeigt einige Anwendungsbeispiele aus der Praxis.

Der gesamte Stoff wurde gestrafft, und insbesondere auf lange grundsätzliche Erörterungen verzichtet. Dem praktischen Beispiel mit vielen nachbaufähigen Schaltungen ist der Vorzug gegeben. Darüber hinaus soll eine Fülle von Übungsaufgaben helfen, den Stoff besser zu verarbeiten, eine Erfolgskontrolle durchzuführen und Ergänzungen zu geben.

Innerhalb kurzer Zeit war die erste Auflage von 1981 vergriffen, so daß 1982 eine Neuauflage mit einigen kleinen Verbesserungen vorgenommen wurde.

Die anhaltend gute Nachfrage machte eine erneute, dritte Auflage erforderlich. Diese wurde zum Anlaß genommen, die Kapitel 4, 5 und 11 zu überarbeiten. Der Wunsch Vieler nach mehr Beispielen für den Umgang mit Übungscomputern und Anleitungen zum Auf- und Ausbau eines solchen Systems wurde in den Kapiteln 4 und 11 berücksichtigt.

Das Kapitel 5 enthält zusätzlich einige Ausführungen über Personal-Computer.

Ich danke dem Verlag W. Girardet für die gute Zusammenarbeit bei der Herstellung dieses Buches.

Dieter Hannemann

Professor Dr. rer. nat. Dipl.-Phys.et-Ing.

Einleitung

Der Mikroprozessor oder auch der Mikrocomputer - der Unterschied wird später erklärt - ist das momentane Endprodukt zweier Entwicklungen, die aufeinander aufgebaut haben.

Zum einen ist es die Entwicklung automatischer Rechen- oder Datenverarbeitungssysteme, d.h. der Computer und zum anderen sind es die Entwicklungen auf dem Gebiet der Elektrotechnik, vom einfachen Schalter (Relais) bis hin zu den sog. integrierten Schaltkreisen (Abb.1).

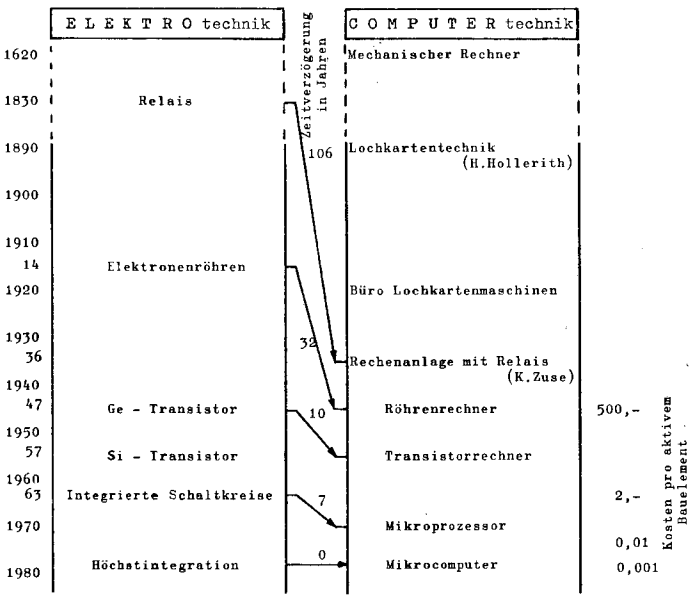


Abb.1: Historischer Überblick im Vergleich zwischen der Elektro- und der Computertechnik

Aufbauend auf ersten mechanischen Rechnern, haben Neuerungen in der Elektronik stets zu einer Verbesserung von Rechenanlagen beigetragen. Diese Verbesserungen betrafen die Größe, die Leistungsfähigkeit, den Stromverbrauch und den Preis eines Computers. Wobei deutlich zu sehen ist, daß die Geschwindigkeit mit der die neuen Bauteile der Elektrotechnik in der Computertechnik Anwendung fanden ständig zunahm und heute praktisch keine Verzögerung mehr auftritt. Die Ursache hierfür ist darin zu sehen, daß die Computertechnik zu einem Teilgebiet der Elektrotechnik geworden ist.

Dieser Schritt wurde gemacht, als es 1970 gelang, die komplette Prozessoreinheit eines Computers auf einem kleinen Silizium-Kristallplättchen unterzubringen. Dies war die Geburtsstunde des Mikroprozessors. Dieser Mikroprozessor-Baustein, entwickelt nach den logischen Prinzipien der Computertechnik, hielt Einzug in die Elektronik und begann dort die hergebrachten Techniken zu revolutionieren.

Heute tritt das Element der Programmierung oder der Programmierbarkeit immer stärker in den Vordergrund.

Ein Mikrocomputer kann ohne Änderung des Aufbaues, an vielen unterschiedlichen Stellen eingesetzt werden und dort völlig verschiedene Aufgaben übernehmen. Die Anpassung an eine andere Aufgabe geschieht allein durch die Erstellung eines anderen Programmes, welches in den Programmspeicher des Mikrocomputers eingeschrieben wird. Ein solcher Mikrocomputer hatte bis 1977 noch etwa die Größe einer Postkarte und bestand aus einer kleinen Anzahl hoch integrierter Bausteine. Heute sind diese einzelnen Bauteile bereits zu einem einzigen Baustein zusammengefaßt worden. Dieser Mikrocomputer besteht also nur noch aus einem kleinen Siliziumplättchen von weniger als 1qcm Fläche.

Der Schlüssel für die flexible und vielseitige Anwendung der Mikrocomputer ist ihre Programmierbarkeit.

Programmieren bedeutet: eine in einzelnen Schritten angegebene Lösungsvorschrift für ein Problem. Wobei man die "einzelnen Schritte" auch "Befehle" nennt.

Die Sprache des Computers, in der man ihm Befehle übermitteln kann, kennt nur zwei Zeichen (Buchstaben) und zwar das L und das H (aus dem Englischen kommend, als Abkürzung für Low und High).

Aus diesen beiden Zeichen müssen alle Worte (z.B. Befehlsworte) zusammengesetzt werden. Der Grund hierfür besteht darin, daß ein Computer nur in der Lage ist mittels einer Art Morsezeichen mit der Umwelt in Kontakt zu treten (siehe Abb.2). Dem Punkt des Morsealphabetes entspricht hierbei z.B. das L und bedeutet, daß auf einer Leitung keine oder nur eine sehr kleine Spannung herrscht, während dann der Strich in diesem Beispiel dem H des Computeralphabetes entspricht und bedeutet, daß auf dieser Leitung eine hohe elektrische Spannung (z.B. 3V) besteht.

Eine Leitung im Computer kann also immer nur den Zustand L oder H annehmen. Diese Vorgehensweise bezeichnet man auch als Digitaltechnik.

Auf dieser Basis lassen sich alle erdenklichen Informationen mit dem Computer austauschen.

Abb.2: Vergleich des Morse- mit einem Computer-Alphabet.

Information:	1	9	8
Morse-Alphabet:	.----	----.	---..
Comp.-Alphabet:	LLLH	HLLH	HLLL

Noch vor 30 Jahren wurden zum Aufbau elektronischer Schaltungen (z.B. Radios) ausschließlich Elektronenröhren und zusätzlich relativ große Widerstände, Kondensatoren und Spulen benutzt.

Der erste Röhrenrechner 1947 wog immerhin 30t.

Mit Beginn des Halbleiterzeitalters, durch die Entwicklung des Transistors (1947), wurden die Bauteile immer kleiner. Wobei ein Transistor die Funktion einer Verstärkerröhre übernahm. Auch die andern Bauteile lassen sich in gleicher Weise herstellen. Hierdurch wurde es möglich (1963), ganze Schaltungen durch Verbinden der einzelnen Bauteile, auf einem Halbleiterplättchen herzustellen; diese nennt man dann integrierte Schaltungen.

Das Ausgangsmaterial für eine Halbleiterschaltung ist ein Kristall, ähnlich einem Edelstein. -Eine moderne Herstellungsvariante benutzt z.B. künstliche Saphire als Basismaterial-. Im allgemeinen wird jedoch ein hochreiner künstlich gezüchteter Silizium-Kristall benutzt.

Zur Zeit ist es möglich ca. 100 000 einzelne Bauteile (Transistorfunktionen) auf einem Silizium-Plättchen (Chip) zu integrieren. Die einzelnen Strukturen auf solch einem Chip haben Abmessungen, die im Bereich von 1/1000 mm liegen.

Ständig eröffnen sich neue Anwendungsgebiete für den Mikrocomputer. Hier einige Beispiele dazu:

Datenverarbeitung:	Büromaschinen, Heim- und Tischcomputer, Terminals
Nachrichtentechnik:	Telefon, Fernschreiber, Bildschirmtext
Industrie:	Roboter, Prozeßsteuerung, Waagen, 'Intelligente' Meßgeräte, Maschinensteuerung
Verkehr:	Ampelsteuerung, Flugsicherungsanlagen, Kfz. Motorsteuerung, Bremsen, Abstandsradar, Leitsystem, Trip-Computer
Unterhaltung:	Fernsehgeräte, Telespiele, Filmkameras, Orgeln
Haushalt:	Waschmaschinen, Heizung, Raumüberwachung, Nähmaschinen
Medizin:	Diagnosehilfen, Computertomographie, Analysegeräte

Hinweise zur Benutzung des Buches

Es gilt auch hier die alte Weisheit: "probieren geht über studieren", das heißt, ohne einen gebührenden praktischen Umgang mit Mikrocomputern, wird der Leser keinen wirklich fundierten Einblick in dieses Fachgebiet bekommen.

Zum Sammeln von Programmiererfahrungen (Teil A) reicht -und ist didaktisch sogar besonders wertvoll - ein kleiner Übungscomputer wie in Abb.3.3. Wenn man ein solches Gerät dann noch selbst aufbaut - z.B. aus einem Bausatz - und verschiedene Peripherie anschließt und betreibt, findet man sich auch in der Hardware (Teil B) sehr schnell zurecht.

Für den Eiligen, der schnell zum eigenen Programmieren kommen möchte, hier nun noch einige Hinweise, welche Abschnitte er zunächst überspringen kann.

Wer die Kapitel 1 bis 3.2 durchgearbeitet hat, sollte in der Lage sein, kleine Programme - ohne Unterprogramme - selbst zu schreiben. Hierzu können dann auf einem Übungssystem, wie in Kapitel 4, kleine Programme erstellt und getestet werden.

Danach ist es unerläßlich zumindest den ersten Teil des Kapitels 3.4: "Unterprogramme" zu bearbeiten und das Erlernte durch weiteres Programmieren einzuüben.

Hiernach kann dann bereits Teil B des Buches in Angriff genommen werden.

Der Autor wünscht dem Leser einen guten Einstieg und viel Erfolg bei dem Bemühen einen Computer für sich arbeiten zu lassen.

A) Programmierung

Als Ergebnis einer Programmierung erhält man ein Programm (software), welches einen Mikroprozessor oder einen Mikrocomputer (hardware) zu einer speziellen Tätigkeit aktiviert. Diese Programmierung läuft grundsätzlich ähnlich ab, wie es allgemein in der Datenverarbeitung üblich ist. Jedoch gibt es auch eine ganze Reihe von Besonderheiten, welche nur in der Mikrocomputertechnik anzutreffen sind. Aus diesem Grunde sollen hier zunächst die wichtigsten Grundlagen der Programmierung und später die spezielle Anwendung bei Mikrocomputern dargestellt werden.

1. Allgemeine Grundlagen

1.1 Mensch und Rechner

Viele Arbeiten, welche früher ausschließlich vom Menschen ausgeführt wurden, werden heute von einem Rechner (Computer) übernommen.

Es wird dem Menschen jedoch nicht nur das Rechnen abgenommen, vielmehr werden durch den Computer insbesondere Steuer -, Regel - und Kontrollfunktionen wahrgenommen.

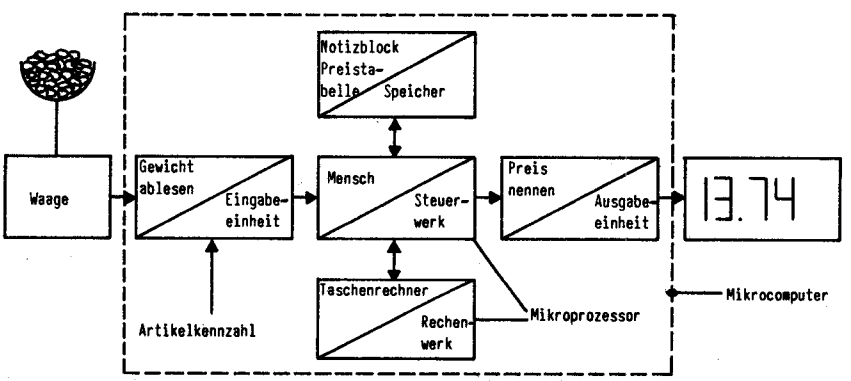


Abb. 1.1: Mensch und Rechner im Vergleich, bei einer Waage

Am Beispiel einer Waage läßt sich zeigen (Abb.1.1), wie die Teile eines Computers benannt werden, welche spezielle Aufgaben des Menschen übernehmen.
In der Mitte steht der steuernde Mensch. Er sorgt dafür, daß der gesamte Ablauf ordnungsgemäß vonstatten geht. Die gleiche Aufgabe übernimmt im Computer das Steuerwerk.

Preisberechnungen werden mit einem Rechengerät oder im Kopf des Menschen durchgeführt. Diese Aufgabe übernimmt im Computer das Rechenwerk.
Steuerwerk und Rechenwerk, zusammen mit einigen anderen speziellen Teilen, bilden die zentrale Prozessor-Einheit eines Computers und sind heute im Mikroprozessor vereinigt.

Ergänzt durch den Speicher, welcher neben einer 'Notizblock-Funktion' auch noch das Programm aufnimmt, und den Ein-/Ausgabe-Einheiten, läßt sich mit Hilfe des Mikroprozessors ein Mikrocomputer (Abb.1.1) aufbauen.
Außerhalb des Computers befindet sich die Umwelt, welche mit ihm über die Ein-/Ausgabe-Einheiten in Kontakt treten kann (Waage und Anzeige in Abb.1.1).
In welcher Form hat diese Kontaktaufnahme zu erfolgen? Bereits in der Einleitung wurde darauf hingewiesen, daß vom technischen Standpunkt aus nur elektrische Signale in Frage kommen. Hierüber mehr im Kapitel 1.3 Codierung.

Zunächst wollen wir uns der Frage vom grundsätzlichen, mathematischen Standpunkt aus nähern.

Da der Computer nur zwei verschiedene Buchstaben kennt und versteht (L, H) müssen wir uns fragen wie sich mittels dieses Alphabetes verschiedene Wörter bilden lassen. Diese verschiedenen Wörter können dann, repräsentiert durch elektrische Spannungen, dem Computer übermittelt werden. Bekommt der Computer die richtigen Wörter zugesandt faßt er sie als Befehle auf und 'gehorcht'.

Die Begriffe 'Buchstabe' und 'Alphabet' sollen hier in einem allgemeineren Sinne verstanden werden, sie lassen sich auch durch die Begriffe 'Zeichen' und 'Zeichenvorrat' ersetzen.

1.2 Zahlensysteme

Eine Zahl läßt sich in verschiedenen Systemen darstellen. Charakteristisch für das Zahlensystem sind die Art und Anzahl der Zeichen (Buchstaben). Wobei die Menge der verschiedenen Zeichen den Zeichenvorrat (numerisches Alphabet) bestimmt.

Gleichzeitig wird durch die Anzahl der Zeichen auch die Basis des Zahlensystems festgelegt.

Am vertrautesten ist uns das

Dezimalsystem.

Der Zeichenvorrat umfaßt die Ziffern 0 bis 9. Die Basis B dieses Zahlensystems ist damit gleich 10.

Der Wert einer Zahl wird nach Potenzen von 10 geordnet. Auf diese Weise erhält man die sog. Potenzschreibweise einer Zahl. Vertrauter, weil einfacher, ist uns jedoch die sog. Stellenschreibweise, welche sich unmittelbar aus der Potenzschreibweise herleiten läßt.

Bei der Stellenschreibweise werden nur noch die Faktoren vor den Zehnerpotenzen aneinandergereit. Beim Lesen einer solchen Zahl geht man dann davon aus, daß eine Ziffer, welche weiter links steht, dem Faktor zur nächst höheren Zehnerpotenz entspricht.

Beispiele:

Ganze Zahl

Stellenschreibweise	Potenzschreibweise
7349	$= 7 \cdot 1000 + 3 \cdot 100 + 4 \cdot 10 + 9 \cdot 1$
	$= 7 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 9 \cdot 10^0$

Dezimalbruch

2,37	$= 2 \cdot 10^0 + 3 \cdot 10^{-1} + 7 \cdot 10^{-2}$

Auf gleiche Weise lassen sich auch andere Zahlensysteme mit anderem Zeichenvorrat (Basis) aufbauen. Alle Zahlensysteme, bei denen die Zahl z in der Stellenschreibweise dargestellt wird, nennt man

Polyadische Zahlensysteme.

Für ein polyadisches Zahlensystem gelten die in Abb. 1.2 dargestellten Regeln.

$B \geq 2$ und $0 \leq b_i < B$

STELLENSCHREIBWEISE:

$$z = \underbrace{b_n\, b_{n-1} \cdots\cdots b_1\, b_0}_{\text{ganze Zahl}} \,,\; \underbrace{b_{-1}\, b_{-2} \cdots\cdots b_{-m+1}\, b_{-m}}_{\text{gebrochene Zahl}}$$

POTENZSCHREIBWEISE (Kurzform)

$$z = \sum_{i=m}^{n} b_i \, B^i \qquad \begin{matrix} n \geq 0 \\ m \leq -1 \end{matrix}$$

B = 2 Dualsystem
B = 5 Quintärsystem
B = 8 Oktalsystem
B = 16 Hexadezimalsystem

Abb. 1.2: Allgemeine Definition eines Polyadischen Zahlensystems

Hier sollen uns jedoch nur zwei weitere Zahlensysteme interessieren, welche für das Verständnis der folgenden Ausführungen von großer Bedeutung sind.

Als Grundlage für die Darstellung von Informationen in einem Computer haben wir zunächst einmal das

Dualsystem.

Das Dualsystem bildet einen Sonderfall im Bereich der Binärsysteme (binär = zweiwertig).

Unter einem Binärsystem versteht man irgendein Zahlensystem, welches mit nur 2 Elementen auskommt (z.B. Tetradendarstellung in Kap.1.3) bei dem aber die Zahl nicht unbedingt durchgängig nach Zweierpotenzen geordnet ist.

Wenn gemäß der Regeln über ein polyadisches Zahlensystem (Abb.1.2) eine Ordnung nach Potenzen von 2 vorgenommen wird, erhalten wir das duale Zahlensystem.

Die beiden Zeichen dieses Systems mit der Basis 2 sind das L und das H.

Beispiel: $HLLH,HLH = 1\cdot2^3+0\cdot2^2+0\cdot2^1+1\cdot2^0+1\cdot2^{-1}+0\cdot2^{-2}+1\cdot2^{-3}$
$= 2^3+2^0+2^{-1}+2^{-3} = 9{,}625$

Eine Dualzahl ist im Schnitt 3,3mal länger als die entsprechende Dezimalzahl.

Dezimal $10^2 10^1 10^0$	Dual $2^7 2^6 2^5 2^4$	Dual $2^3 2^2 2^1 2^0$	Hexadezimal $16^1 16^0$
0		L	0
1		H	1
2		H L	2
3		H H	3
4		H L L	4
5		H L H	5
6		H H L	6
7		H H H	7
8		H L L L	8
9		H L L H	9
1 0		H L H L	A
1 1		H L H H	B
1 2		H H L L	C
1 3		H H L H	D
1 4		H H H L	E
1 5		H H H H	F
1 6	H	L L L L	1 0
1 7	H	L L L H	1 1
:			
3 1	H	H H H H	1 F
3 2	H L	L L L L	2 0
:			
6 3	H H	H H H H	3 F
6 4	H L L	L L L L	4 0
:			
1 2 7	H H H	H H H H	7 F
1 2 8	H L L L	L L L L	8 0
:			
2 5 5	H H H H	H H H H	F F

Abb. 1.3: Vergleich der drei wichtigsten Zahlensysteme

Wegen der Länge der Dualzahlen gestaltet sich der Umgang mit ihnen besonders aufwendig. Aus diesem Grunde hat sich ein weiteres Zahlensystem bewährt, welches die abkürzende Darstellung einer vierstelligen Dualzahl (Tetrade) durch ein Zeichen gestattet und zwar im

Hexadezimalsystem.

Eine vierstellige Dualzahl kann insgesamt $2^4 = 16$ verschiedene Werte annehmen. Deshalb muß das zur Abkürzung einer Tetrade zu benutzende Zahlensystem die Basis 16 haben. Als Zeichenvorrat wird das numerische

Alphabet (0...9) und die ersten sechs Zeichen des Buchstabenalphabetes (A...F) verwandt. Im folgenden werden diese Zeichen häufig HEX-Zeichen genannt.

b_i: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
B=16 — Zeichenvorrat

Beispiel: $379H = 3 \cdot 16^2 + 7 \cdot 16^1 + 9 \cdot 16^0 = 889D$

$BAD_H = 11 \cdot 16^2 + 10 \cdot 16^1 + 13 \cdot 16^0 = 2989_D$

Zur Kennzeichnung, ob es sich um eine Dezimalzahl oder eine Haxadezimalzahl handelt, wird hinter die Zahl als Index, oder auch in gleicher Höhe, ein D oder H geschrieben.

Die Grundrechenarten im Dualsystem

Das Rechnen mit Dualzahlen ist besonders einfach, da es ja nur zwei Zeichen gibt. Zum besseren Verständnis einiger späterer Besonderheiten soll hier kurz auf das Wesentliche eingegangen werden. Analog der uns bekannten Regeln im Dezimalsystem erhält man für die Addition von Dualzahlen:

dezimal	dual
0 + 0 = 0	L + L = L
0 + 1 = 1	L + H = H
1 + 0 = 1	H + L = H
1 + 1 = 2	H + H = HL

Beispiel:

	70		HLLLHHL
+	34	+	HLLLHL
	1 Übertrag		HH
=	104	=	HHLHLLL

Das Addieren von Dualzahlen ist eine der Basisfähigkeiten eines jeden Computers. Alle anderen Grundrechenarten können auf die, evtl. mehrfach durchzuführende, Addition zurückgeführt werden.

Die Grundregeln der Multiplikation im Dualsystem sind:

L · L = L
L · H = L
H · L = L
H · H = H

Werden an eine Dualzahl s Stellen (L) angehängt, so bedeutet dies eine Multiplikation mit 2^s.

Beispiele: HLH = 5 --► HLH LL = 20 = $5 \cdot 2^2$

dezimal			dual		
Multi-plikand	• Multi-plikator		Multi-plikand	• Multi-plikator	
33	• 13		HLLLLH	• HHLH	
33			HLLLLH		
99		Teilprodukte	HLLLLH		
= 429			LLLLLL		
			HLLLLH		
			= HHLHLHHLH		

Im Dualsystem sind die Teilprodukte entweder Null oder gleich dem Multiplikanden.

Die Subtraktion läßt sich auf eine Addition zurückführen, wenn von der zu subtrahierenden Zahl vorher das Komplement gebildet wird.

Das 1-Komplement (Einerkomplement = unechtes Komplement) einer Dualzahl entsteht dadurch, daß jedes L durch ein H und jedes H durch ein L ersetzt wird (Negation).

Da das Ergebnis einer Subtraktion auch negativ sein kann, ist noch eine Vorzeichenregel erforderlich. Die Vorzeichen + und -, welche normalerweise benutzt werden, eignen sich für einen Computer nicht, da dieser ja nur die Zeichen L oder H kennt. Aus diesem und anderen Gründen wird für das Vorzeichen + ein L und für - ein H benutzt und dieses dann vor die Zahl geschrieben.

Beispiel: H LHHL = -6 L HLHL = +10

Zur Berechnung von 6-4 wird zunächst das 1-Komplement der Zahl 4 gebildet und dieses dann zur 6 addiert. Entsteht hierbei ein Übertrag, so ist er noch hinzu zu addieren. Wenn die entstehende Zahl negativ (H) ist, so muß das Ergebnis noch rückkomplementiert werden.

Beispiele:

-4 = H LHLL	--► 1-Komplement	--► H HLHH		
+6	L LHHL	- 6	H HLLH	
-4	H HLHH	- 4	H HLHH	
= +2	HL LLLH	= -10	HH LHLL	
	└►+ H		└►+ H	
	=L LLHL		=H LHLH	
	=+ 2	Rückkomplement:	=H HLHL = -10	

Wird anstelle des 1-Komplements das 2-Komplement (echtes Komplement) benutzt, braucht nach der Addition der evtl. Übertrag nicht mehr hinzuaddiert zu werden, dieses geschieht bereits bei der Komplementbildung. Das 2-Komplement wird durch die Addition eines H aus dem 1-Komplement gebildet.

Beispiel: -4 --► H HLHH 1-Komplement
+ ____H
H HHLL 2-Komplement

Ein, beim Rechnen mit dem 2-Komplement auftretender, Übertrag wird nicht mehr berücksichtigt.

Beispiele:

+6	L LHHL		- 6	H HLHL	
-4	H HHLL		- 4	H HHLL	
= +2	HL LLHL		= -10	HH LHHL	
	=L LLHL			=H LHHL	
	=+ 2		2-Rückkompl.:	=H HLLH+H	
				=- HLHL=-10	

Die einfachste Art der Division ist die fortgesetzte Subtraktion des Divisors vom Dividenden.

Beispiel: Dividend : Divisor = Quotient

672 : 32	= 21	
-32	1. Sub.	} 1. Quotientenstelle = 2
=+35		
-32	2. Sub.	
=+ 3		
-32	3. Sub.	geht nicht
=-29		
+32	Korrekturaddition	
=+ 32	nächste Dividendenstelle	
- 32	1. Sub.--►	2. Quotientenstelle = 1
= 0		

Soll eine Division durch eine Zweierpotenz (2, 4, 8, 16....) durchgeführt werden, so wird das Komma einfach um soviel nach links verschoben, wie der Zweierexponent angibt.

Beispiel: 33:4=8,25 $4=2^2$
HLLLLH:HLL = HLLL,LH

Umwandlung von Zahlen

Beim Umgang mit Mikroprozessoren ist es manchmal erforderlich eine Zahl, aus einem Zahlensystem in die entsprechende Zahl in einem anderen System, umzuwandeln.
Hierzu gibt es verschiedene Verfahren. Besonders häufig sind Umrechnungen zwischen dem dualen und dem hexadezimalen Zahlensystem durchzuführen.
Diese Umrechnungen sind jedoch auch besonders einfach.

Beispiel: dual --► hexadezimal
HHL LHHH = 67H (4+2 4+2+1)
hexadezimal --► dual
B8H = HLHH HLLL (B = 11 = 8+0+2+1, 8 = 8+0+0+0)

Bei der Umrechnung dual --► hexadezimal, teilt man die Dualzahl von rechts nach links in Tetraden auf und kann diese dann unmittelbar in das entsprechende HEX-Zeichen umrechnen, wenn man die Wertigkeit der einzelnen Stellen in einer Tetrade (8-4-2-1) berücksichtigt.

Die Umrechnung in der Gegenrichtung erfolgt dadurch, daß man feststellt, welche der Wertigkeitszahlen einer Tetrade (8-4-2-1) in der hexadezimalen Zahl enthalten sind. Man beginnt mit der höchsten Zahl und setzt immer dann ein H, wenn sie enthalten ist. Hierzu ist es allerdings sinnvoll das jeweilige HEX-Zeichen vorher gedanklich in eine Dezimalzahl umzuwandeln.

Mit etwas Übung lassen sich diese elementaren Umwandlungen im Kopf durchführen. Wenn die Übung fehlt, kann auch die Tabelle in Abb.1.3 zu Hilfe genommen werden.

Bei der Umwandlung von Zahlen zwischen dem dualen und dem dezimalen Zahlensystem sollte man unterscheiden, ob der Programmierer diese Umrechnung durchführen soll, oder der Mikrocomputer.

Umwandlungen, welche der Mikrocomputer zu machen hat, die also programmiert werden müssen, werden später beschrieben (Kap.1.3, 3.5, 5.).

Wenn der Programmierer während seiner Tätigkeit eine Umwandlung durchzuführen hat, ist folgendes Verfahren angebracht.

Zunächst einmal werden die Zweierpotenzen hingeschrieben, falls man sie nicht im Kopf hat. Bei einer Umwandlung vom dualen ins dezimale Zahlensystem werden dann einfach alle in der Dualzahl vorkommenden (H) Potenzwerte addiert. Eine Umwandlung in Gegenrichtung wird dadurch erreicht, daß man prüft, welche der Zweierpotenzen in der Dezimalzahl enthalten (H) sind. Hierbei beginnt man mit der größten Zweierpotenz und geht dann systematisch zu den kleineren über.

Beispiele:

128	64	32	16	8	4	2	1
dual --► dezimal							
H	L	H	L	L	L	H	L = 128+32+2 = 162D

dezimal --► dual

162		34				2	
-128		-32				-2	
34		2				0	
H	L	H	L	L	L	H	L

Läßt sich die Zweierpotenz von der Dezimalzahl oder ihrem Rest subtrahieren, ohne das eine negative Zahl übrig bleibt, so ist sie in der Dezimalzahl enthalten und es wird ein H gesetzt, andernfalls ein L.

1.3 Codierung

Das Wort Codierung bedeutet Verschlüsselung und unter einem Code versteht man eine Umwandlungsvorschrift für eine Information.

Zur Kontaktaufnahme mit einem Mikroprozessor bedient man sich des Binäralphabetes, d.h. mittels der beiden Binärzeichen L und H können Wörter gebildet werden die ein Mikroprozessor 'versteht'. In gleicher Weise gibt auch der Mikroprozessor Informationen an seine Umwelt hinaus.

Codierung der Binärzeichen

Die technische Realisierung der beiden Binärzeichen L und H kann auf verschiedene Weise erfolgen.

Beispiele: L entspricht: Aus, offen , kein Strom, keine Spannung
H entspricht: Ein, geschlossen, Strom , Spannung

Am häufigsten tritt heute die letzte Form auf, d.h. eine Spannungscodierung der Binärzeichen.
In Abb.1.4 ist ein Wort (Zahl) dargestellt, welches aus 4 Buchstaben (Binärzeichen) besteht. Dem L entspricht der Zustand "keine Spannung" und dem H der Zustand "positive Spannung". Diese unterschiedlichen Spannungszustände, welche in einem bestimmten zeitlichen Rhythmus aufeinanderfolgen, können z.B. an dem Anschlußstift eines Mikroprozessors auftreten und für ihn eine Information bedeuten.

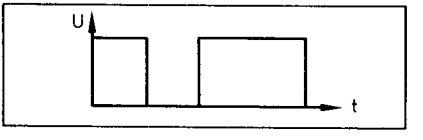


Abb. 1.4: Ideales Spannungs/Zeit-Diagramm der Zahl HLHH

Wenn die einzelnen Zeichen zeitlich hintereinander erscheinen, so wie in Abb.1.4, nennt man das einen "seriellen Informationstransport" (oder serielle Datenübertragung). Dies ist vergleichbar mit einem Menschen, welcher ein Wort buchstabiert. Wenn jedoch der Mensch ein Wort liest, so nimmt er es im allgemeinen als Gesamtheit in sich auf. Dieser Vorgang wird bei einem Computer "paralleler Informationstransport" genannt. Angewandt auf einen Mikroprozessor bedeutet dies, daß z.B. an 4 Anschlußstiften die 4 Binärzeichen der Abb.1.4 gleichzeitig für eine bestimmte Dauer anliegen.

Für jede einzelne Stelle in einer Dualzahl, d.h. für jede Ziffer wurde der Begriff bit (binary digit) eingeführt. Ein Bit steht also für eine Alternativentscheidung L oder H.

Beispiel: HLHH --► 4bit Datenwort

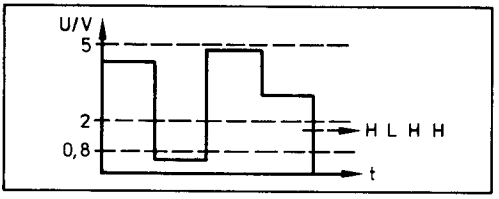


Abb. 1.5:
Reales Spannungs/Zeit-Diagramm (am Beispiel der TTL-Technik, Kap. 6)

Ein großer Vorteil der digitalen Informationsverarbeitung besteht darin, daß bei der Darstellung der beiden Binärzeichen durch elektrische Spannungen, keine genauen Werte eingehalten zu werden brauchen. In Abb.1.5 ist ein Beispiel hierfür dargestellt. Alle Signale die eine Spannung haben, welche zwischen OV und 0,8V liegt, werden dem Binärzeichen L (Low) zugeordnet. Eine Spannung zwischen 2V und 5V dagegen bedeutet H (High).

Befehlsdarstellung

Wenn einem Mikroprozessor eine Dualzahl übermittelt wird, so kann sie für ihn die Bedeutung eines Befehlswortes oder eines Datenwortes haben, jenachdem zu welchem Zeitpunkt das Wort eintrifft.

Ein Befehlswort aktiviert bestimmte Fähigkeiten in einem Mikroprozessor. Das erste Wort (Operationsteil des Befehles) gibt Instruktionen an das Steuerwerk und es kommt zur Ausführung einer Operation.

Die Menge der vom Mikroprozessor durchführbaren Operationen läßt sich in drei Gruppen einteilen.

Arithmetische Operationen (+, -, ...)
Logische Operationen (>, =, ...)
Organisatorische Operationen (Verschieben, Springen, ...)

In Abb.1.6 ist der typische Aufbau eines Mikroprozessor-Befehles dargestellt. Der Operationsteil dient zur Codierung der durchzuführenden Operation. Wenn der Operationsteil z.B. aus 8 bit besteht, so lassen sich hiermit $2^8 = 256$ verschiedene Operationen

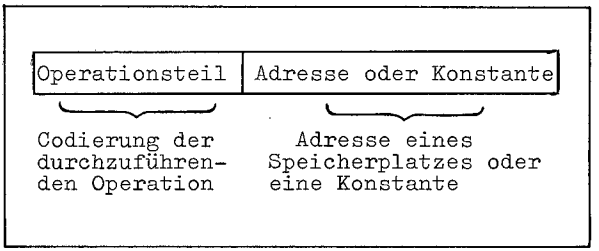


Abb.1.6: Aufbau eines Befehles

codieren (s.a. Kap.3.2), d.h. jede Bitkombination aktiviert eine andere Tätigkeit des Mikroprozessors. Es gibt Befehle, welche nur aus einem Wort bestehen und andere, an die noch eine Adresse oder eine Konstante angehängt werden muß.

Beispiele:

LLHHHHLL --► Erhöhe den Inhalt des Rechenregisters um 1.
HHLLLLHH Adr. --► Springe zur angegebenen Adresse.
LLHHHHHL Konst. --► Bringe die Konstante ins Rechenregister.

Eine ausführliche Behandlung dieser Thematik erfolgt in Kap.3.2 am Beispiel der Befehle des Mikroprozessors 8080.

Hier soll uns nun die Frage interessieren, welche Bedeutung eine Konstante haben kann. Die Konstante in einem Befehl ist eine Form eines Datenwortes. Sie kann eine beliebige Bedeutung erlangen und wird insbesondere auch benutzt um als Codewort für die verschiedensten Zeichen zu dienen.

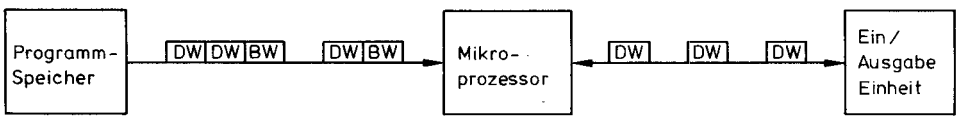


Abb.1.7: Transport von Daten- und Befehlsworten vom und zum Mikroprozessor.

In Abb.1.7 ist dargestellt, daß ein Datenwort (oder Datum) nicht nur als Teil eines Befehles auftreten kann. Insbesondere der Austausch mit der Umwelt über die Ein-/Ausgabe Einheiten erfolgt mittels solcher Datenworte (Datenübertragung).

Codierung alphanumerischer Zeichen

Der Informationsinhalt der Datenworte, welche mit dem Computer ausgetauscht werden, kann sehr unterschiedlich sein. Beim Beispiel der Waage in Abb.1.1 müssen von der Wägeeinrichtung die Gewichtsdaten übernommen und später der Preis in Dezimalziffern angezeigt werden. Da aber z.B. zur Anzeigeeinheit nur binäre Daten übertragen werden können, muß ein Code vereinbart werden, welcher eine Zuordnung zwischen den Dezimalziffern 0...9 und einem Binärwort herstellt.

Wie lang muß ein Binärwort sein, um die zehn Dezimalziffern darstellen zu können? Die Mathematik lehrt uns, daß mittels 2 verschiedener Zeichen (L u. H), welche zu einer Folge von s Zeichen aneinandergereiht werden, 2^s verschiedene Kombinationen gebildet werden können. Hieraus ergibt sich, daß 4 Bits nötig sind um die 10 Dezimalziffern zu codieren. Die Codierung erfolgt dann sinnvoller Weise so wie es bereits in Abb.1.3 dargestellt wurde.

Beispiele: 0 = LLLL, 1 = LLLH,...5 = LHLH,...9 = HLLH

Diese Form der Codierung wird auch BCD-Code (Binary Coded Dezimals), oder auch 8-4-2-1 -Code genannt. Die letzte Bezeichnung gibt einfach die Wertigkeit der 4 Bits wieder.

Da bei dieser Codierung immer 4 Bits zusammengefaßt auftreten ist auch noch die Bezeichnung Tetraden-Darstellung einer Dezimalzahl üblich (s.a. Kap.1.2).

Beispiel: Tetraden-Darstellung der Dezimalzahl 139

1	3	9
LLLH	LLHH	HLLH

Ein Teil der mit den 4 Bits möglichen Kombinationen (2^4 = 16) wird nicht benötigt, diese Tetraden werden auch Pseudotetraden genannt. Es sind dies all diejenigen Tetraden, welche beim Hexadezimalalphabet den Ziffern A...F entsprechen (s.a. Abb.1.3).

Wenn neben den numerischen Zeichen (0...9) auch noch die Buchstaben (Alphazeichen) und die Sonderzeichen (? + ; - / ...) übertragen werden sollen, benötigt man mehr Bits.

Die Summe dieser Zeichen nennt man alphanumerische Zeichen, oder alphanumerisches Alphabet.

Zur Codierung der alphanumerischen Zeichen werden zwei Tetraden benutzt. Diese beiden Tetraden ergeben zusammengefaßt ein 8 bit-Wort welches den Namen Byte erhält.

1 Byte = 8 bit = 2 Tetraden = 2 HEX-Zeichen

Beispiel: HHLLLHHH = HHLL LHHH = C7H

Die linken 4 Bits werden höherwertige Tetrade und die rechten 4 Bits niederwertige Tetrade genannt.

Mittels der höherwertigen Tetrade werden die gesamten alphanumerischen Zeichen in einzelne numerierte Zonen aufgeteilt, welche jeweils 16 Zeichen umfassen. Die niederwertige Tetrade dagegen wird zur Durchnumerierung der 16 Zeichen einer Zone benutzt.

Beispiel: Darstellung des Buchstaben B

L H L L	L L H L
Zonen-teil	Ziffern-teil

= Zone 4, drittes Zeichen

Durch den Zonenteil lassen sich 16 verschiedene Zonen adressieren, d.h. insgesamt wären 16 x 16 = 256 verschiedene Zeichen darstellbar. Dies ist jedoch im allgemeinen nicht erforderlich, deshalb wird beim amerikanischen Fernschreib-Code (ASCII) das höchste Bit weggelassen und gelegentlich als Paritätsbit zur Datensicherung benutzt.

Die ASCII-Norm (American Standard Code of Information Interchange) wird in der Datenverarbeitung überwiegend angewandt. Neben allen alphanumerischen Zeichen beinhaltet der Code auch noch zwei Zonen mit

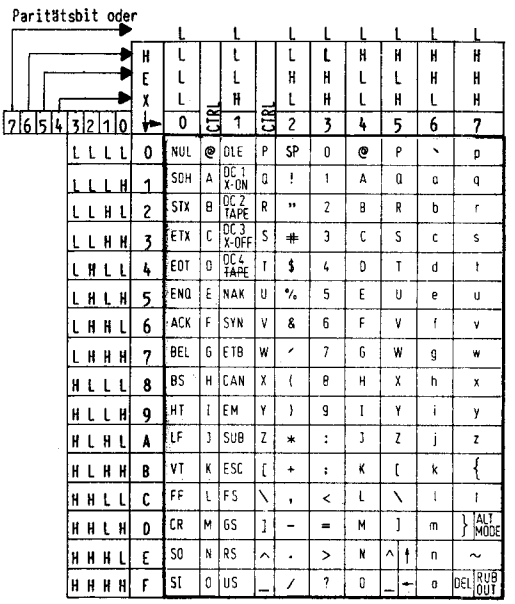


Abb. 1.8: ASCII-Werte-Tabelle. Die ersten beiden Spalten enthalten Control-Zeichen in der üblichen englischen Bezeichnungsweise, s.a. Anhang 3. (SP = space = blank = Leerschritt)

insgesamt 32 sog. Steuerzeichen. Diese Steuerzeichen sind im allgemeinen unsichtbar, d.h. es ist ihnen kein sichtbares Zeichen zugeordnet (s.a. Anhang 3.). Mit einer Standard-Tastatur (Kap.1o.1) lassen sie sich durch das gleichzeitige Drücken der sog. CTRL-Taste (control) mit einer Zeichen-Taste erzeugen.
In Abb.1.8 sind alle gültigen ASCII-Zeichen zusammengestellt. Jede Spalte entspricht einer Zone. Die einem Zeichen zugeordnete Dualzahl (Byte) und die entsprechende HEX-Zahl setzen sich jeweils aus den Teilen am oberen und am linken Tabellenrand zusammen.

Beispiel: A = LHLLLLLH = 41H, 5 = LLHHLHLH = 35H, t = LHHHLHLL = 74H

1.4 Programmierung

Indem man einem Mikroprozessor eine Folge von Befehlen erteilt, kann er zu einer sinnvollen Tätigkeit aktiviert werden. Eine solche Befehlsfolge wird Programm genannt, wenn sie eine lückenlose und in einzelnen Operationsschritten angegebene Lösungsvorschrift zur Bearbeitung eines Problemes darstellt.

Flußdiagramm

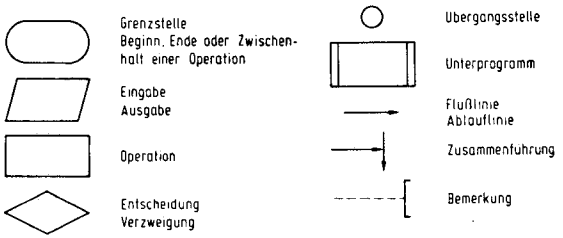


Abb.1.9: Flußdiagrammsymbole nach DIN 66001

Um sich über den Lösungsweg eines Problemes klar zu werden, hat es sich bewährt zunächst ein Flußdiagramm anzufertigen.

1. Rufe Klara an um 16^{00}
2. Wenn erreicht, weiter bei 7.
3. Wenn schon 17^{00}, weiter bei 9.
4. Warte 15 min
5. Rufe Klara erneut an
6. Weiter bei 2.
7. Versuche Treffen zu arrangieren
8. Wenn klar, dann weiter bei 11.
9. Rufe Inge an
10. Wenn nicht erreicht, oder nicht klar weiter bei 12.
11. Ausgehen
12. Schluß

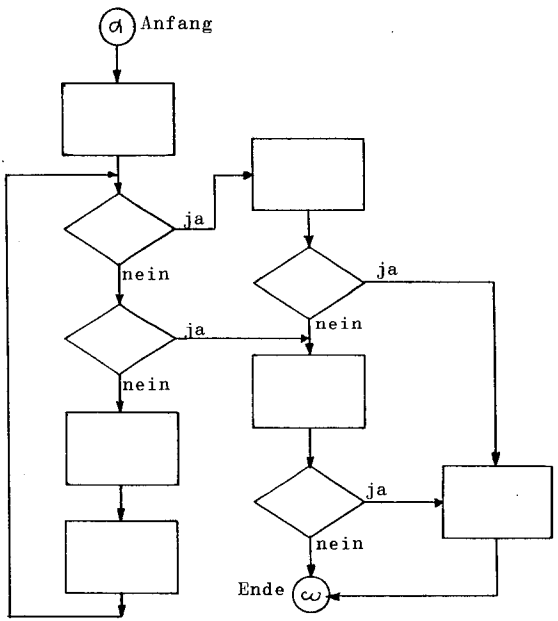


Abb.1.10: Flußdiagramm zur Lösung des Problems einer Rendevous-Vereinbarung

Zur Herstellung dieses Flußdiagrammes können die in Abb.1.9 dargestellten Symbole benutzt werden.

Ein Flußdiagramm kann zur groben Strukturierung eines größeren Problemes, oder aber auch zur detailierten Darstellung eines Lösungsweges benutzt werden. Im zweiten Fall kann dies so weit gehen, daß in den Kästchen des Flußdiagrammes die einzelnen Befehle eines Programmes stehen. Ein einfaches Beispiel zur Lösung eines 'alltäglichen' Problemes ist in Abb.1.10 dargestellt. Dem Leser wird empfohlen das in der Abb. durch die Punkte 1. bis 12. kurz beschriebene Problem auf das daneben stehende Flußdiagramm zu übertragen. Es reicht, wenn in die Kästchen des Flußdiagrammes die Nummern der zugehörigen Textzeilen eingetragen werden. (siehe auch Übungsaufgabe 1.21)

Maschinensprache

Wenn ein Computer zur Lösung eines Problemes herangezogen werden soll, muß hierzu ein Programm erstellt werden, welches aus einer Folge von Befehlen besteht. Diese Befehle wiederum werden als eine Folge von Binärzeichen dargestellt (Kap.1.3). Jede Mikroprozessor-Familie hat einen anderen Befehlsvorrat und selbst gleiche Befehle werden im allgemeinen durch eine andere Bitfolge codiert.

Hieraus erklärt sich, weshalb ein Programm, welches für einen Mikroprozessor geschrieben wurde, bei einem Mikroprozessor eines anderen Typs im allgemeinen nicht ablauffähig ist.

Die unterste Ebene, auf der ein Programm erstellt werden kann, ist das Programmieren im sog. Objektcode. Hierbei werden die Befehle direkt in der, eine Tetrade abkürzend darstellenden, Hexadezimalschreibweise eingegeben oder aber im sog. Mnemonic-Code (Kap.3.2). Wenn ein Mikrocomputer im Mnemonic-Code programmierbar ist, wird der Operationsteil eines Befehles durch eine Buchstabenkombination ersetzt, welche als Gedächtnisstütze für die Wirkung des Befehles dient. Der Mikrocomputer wandelt dann, durch ein spezielles Programm, den Mnemonic-Code in den entsprechenden Binärcode um.

Eine einfachere Programmierbarkeit wird durch die Einführung einer sog. Assembler-Programmiersprache erreicht (Kap. 5.3). Die Assembler-Programmiersprachen und einige verbesserte Formen hiervon sind jedoch noch immer sog. 'Maschinenorientierte Programmiersprachen'. Dies besagt, daß ein in dieser Sprache verfaßtes Programm nur auf dem zugehörigen Mikroprozessor-Typ ablauffähig ist.

Ein Assembler, welcher ein, in der zugehörigen Assembler-Sprache verfaßtes, Programm in den Objekt-Code übersetzt, arbeitet in ähnlicher Weise wie ein Compiler (siehe weiter unten).

Höhere Programmiersprachen

Eine Unabhängigkeit vom Mikroprozessor wird beim Programmieren erst durch die Verwendung sog. 'Problemorientierter Programmiersprachen' erreicht. Diese Programmiersprachen orientieren sich nicht mehr am Typ des Mikroprozessors sondern an Problemgruppen für die sie entworfen worden sind.

Beispiele problemorientierter Programmiersprachen:

Programmiersprachen zur Lösung naturwissenschaftlich, technischer Probleme.
ALGOL (= algoritmic language)
FORTRAN (= formula translation)

Programmiersprache zur Lösung wirtschaftwissenschaftlicher und kommerzieller Probleme.
COBOL (= common business oriented language)

Universelle Programmiersprachen.
PL/1 (= programming language 1)
PASCAL (benannt nach Blaise Pascal, 1623-1663, Erfinder der ersten Rechenmaschine)

Entwickelt wurden diese Programmiersprachen für den Gebrauch an Groß- und Minirechnern. Sie eignen sich jedoch auch für den Einsatz an Mikrocomputern. Die meisten Mikrocomputer besitzen Compiler oder Interpreter welche es gestatten Programme, geschrieben in einer höheren (problemorientierten) Programmiersprache, zu verarbeiten.

Ein Compiler ist ein spezielles Programm, welches das in einer höheren Programmiersprache geschriebene Quell-Programm in den Objektcode des zugehörigen Mikroprozessors übersetzt und abspeichert. Das auf diese Weise erzeugte Objektcode-Programm kann dann durch den Mikroprozessor abgearbeitet werden. In gleicher Weise arbeitet auch ein Assembler.

Demgegenüber werden bei einem Interpreter die einzelnen Befehle des Quellprogrammes nur interpretiert. Das Interpretierprogramm (=Interpreter) nimmt einen Befehl des Quellprogrammes und gibt dann dem Mikroprozessor eine Reihe von Befehlen, damit eine dem Quellbefehl entsprechende Wirkung erzeugt wird. Aufgrund dieser Wirkungsweise bleibt das Quellprogramm erhalten und wenn es gestartet werden soll muß immer erst der Interpreter in den Speicher gebracht und gestartet werden. Der Interpreter arbeitet dann das Quellprogramm ab.

Insbesondere bei Mikrocomputern werden häufig Interpreter an Stelle von Compilern benutzt, da sie im allgemeinen sehr viel billiger sind.

1.5 Übungsaufgaben

1.1 Berechnen Sie HLLHHLLL + LLHHHLHH

1.2 Berechnen Sie HLLH • HLL

1.3 Wieviel ergibt HHLL - HLLH?

1.4 Berechnen Sie HLLH : HLL

1.5 Wandeln Sie die Dezimalzahl 2378 in eine Dualzahl um.

1.6 Wandeln Sie HLLHLHHL in eine Dezimalzahl um.

1.7 Wandeln Sie LHHHHLHL in eine Hexadezimalzahl um.

1.8 Wandeln Sie 178_D in eine Hexadezimalzahl um.

1.9 Wandeln Sie CD in eine Dezimalzahl um.

1.10 Wie lautet der Dualbruch HLLH, HLHHLH als Dezimalbruch?

1.11 Was verstehen Sie unter arithmetischen, logischen und organisatorischen Operationen?

1.12 Wieviel verschiedene Befehle lassen sich durch einen 8 Bit Operationsteil darstellen?

1.13 Was versteht man unter einer Tetrade und Pseudotetraden?

1.14 a) Welchen Wert hat die Dezimalzahl, welche folgender Tetraden Darstellung entspricht LLLHHLLHLHHL?

b) Wandeln Sie diese Dezimalzahl in eine Dualzahl um und vergleichen Sie diese mit der Binärzahl unter a)

1.15 Wieviel Bit hat ein Byte?

1.16 Welche Bezeichnung wird noch für den BCD-Code verwandt?

1.17 Wieviel verschiedene Zeichen lassen sich mit 7 Bit codieren?

1.18 Aus wieviel Bit besteht der Zonenteil beim ASCII-Code?

1.19 Welchen HEX-Code hat das ASCII-Zeichen g?

1.20 Was verstehen Sie unter alphanumerischen Zeichen?

1.21 Beschriften Sie das Flußdiagramm in Abb.1.10.

1.22 Erklären Sie den Unterschied zwischen einer problemorientierten und einer maschinenorientierten Programmiersprache.

2. Grundeinheiten eines Mikrocomputers

Wenn ein Computer zur Bearbeitung von z.B. mathematischen oder organisatorischen Problemen herangezogen werden soll, ist es für den Benutzer nicht unbedingt erforderlich, daß er Kenntnisse über den prinzipiellen oder gar den elektronischen Aufbau des Computers besitzt. Die Programmierung eines solchen Computers erfolgt dann in einer höheren, oder einer speziellen Programmiersprache.

Der Schwerpunkt der Mikrocomputer Anwendungen liegt jedoch auf den Gebieten der Steuer-, Regel- und Kontrollaufgaben.

Dieser Problemkreis und die in diesen Fällen häufig angewandte Form der Programmierung in Maschinensprache, macht es jedoch erforderlich, an dieser Stelle einige strukturelle Eigenarten der Mikroprozessoren und der daraus aufgebauten Mikrocomputer, kurz darzulegen.

Eine detailierte Behandlung des elektronischen Aufbaues der Mikrocomputer erfolgt dann im Teil B.

2.1 Der Mikrocomputer

Das Kernstück eines Mikrocomputers (MC) ist der Mikroprozessor (MP). Die meisten Mikroprozessoren befinden sich in einem Gehäuse mit den Abmessungen 5x1,5x0,5cm^3. Der Chip im Inneren des Gehäuses ist über 40 Kontaktbeinchen mit der Außenwelt verbunden.

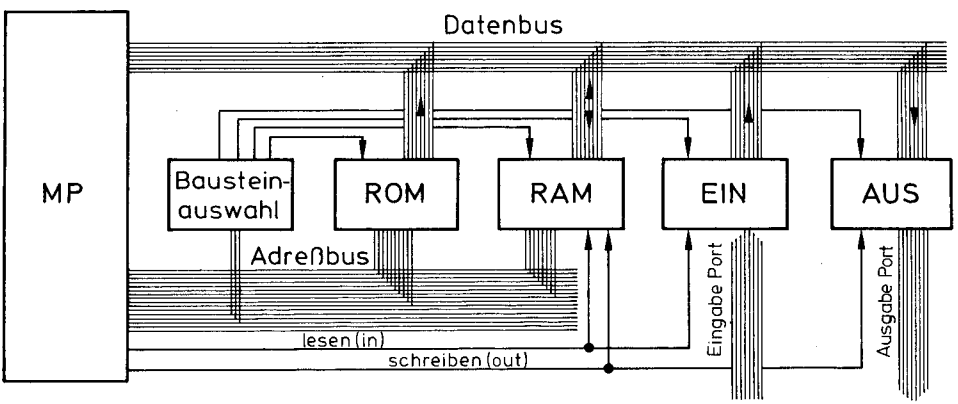


Abb. 2.1:
Aufbau eines Mikrocomputers
ROM (Read Only Memory) = Programmspeicher. RAM (Random Access Memory) = Arbeitsspeicher. EIN/AUS=Schnittstellen zum Datenaustausch mit der Umwelt. Bus = Mehradrige Sammelschienen für den Informationstransport.

Damit der Mikroprozessor arbeiten kann, braucht er einen Programmspeicher aus dem er sich seine Befehle holen kann. Hierzu wird im allgemeinen ein ROM (Read Only Memory) benutzt. Diese Lesespeicher sind integrierte Halbleiterbausteine, welche bei Spannungsausfall ihren Inhalt behalten. Dies ist wichtig, damit das Programm beim Ausschalten nicht verloren geht.
Damit der Mikroprozessor sich Zwischenergebnisse oder andere Daten merken kann, benötigt er noch einen zusätzlichen Speicher - einen sog. Arbeitsspeicher - in den er wahlweise etwas hineinschreiben oder herauslesen kann. Hierzu werden wiederum meistens spezielle Halbleiterspeicher, sog. RAMs, benutzt (Random Access Memory).

Zur Vervollständigung des Mikrocomputers müssen nun noch ein oder mehrere Ein-/Ausgabe Bausteine hinzu gefügt werden, denn nur dann kann der Mikroprozessor eine sinnvolle Funktion erfüllen, indem er über diese Schnittstellen (interface) mit seiner Umwelt in Kontakt tritt.
Wie in Abb.2.1 ersichtlich, sind die Bausteine, welche zusammen den Mikrocomputer bilden, über sog. Busse miteinander verbunden. Ein Bus ist eine mehradrige Sammelschiene für den Informationstransport.
Im Mikrocomputer gibt es den Datenbus - welcher heute meistens aus 8 Leitungen besteht - zur Ein- und Ausgabe von Datenworten, wie z.B. Meßergebnisse, Rechenergebnisse, externe Steuersignale und zur Einholung von Befehlswörtern.
Der Adreßbus hat meistens 16 Leitungen und dient dem Mikroprozessor dazu, den anderen Bausteinen mizuteilen, wer und welches Teil in dem angesprochenen Baustein, ihm über den Datenbus etwas mitteilen, oder eine Information entgegen nehmen soll.
Über den Adreßbus werden also nur vom Mikroprozessor Adressen herausgegeben, während auf dem Datenbus Informationen in beiden Richtungen fließen können.
Da alle Bausteine am Datenbus angeschlossen sind, wird eine Zusatzschaltung benötigt, welche die vom Mikroprozessor ausgesandten Adressen in sog. Bausteinauswahlsignale (Chip select, CS; Chip Enable, CE) umwandelt. Auf diese Weise wird erreicht, daß immer nur ein Baustein über den Datenbus mit dem Mikroprozessor verbunden ist, alle anderen befinden sich dann im sog. hochohmigen Zustand. Diese Bausteinauswahl kann z.B. durch einen einfachen 1 aus 8 Dekoder vorgenommen werden.

Die Bausteinauswahlleitungen und einige weitere Leitungen, wie z.B. die welche anzeigen ob der Mikroprozessor lesen oder schreiben (in/out) will, werden auch Steuerbus genannt.

Die Weiterentwicklung der Halbleitertechnologie führt zu immer höheren Integrationsdichten bei den Bausteinen. Hierdurch ist es gelungen, die soeben beschriebenen Einheiten eines Mikrocomputers in einem Baustein, zu vereinen (single chip computer)

Beispiel: Der Single Chip Mikrocomputer 8051 beinhaltet neben dem Mikroprozessor
4096 Byte Programmspeicher
128 Byte Arbeitsspeicher
32 Ein-/Ausgabe Kanäle
und zusätzlich noch einige spezielle Komponenten.

2.2 Die Speicher

Der Speicher eines Computers dient zur Aufnahme des Programmes und als Arbeitsspeicher.

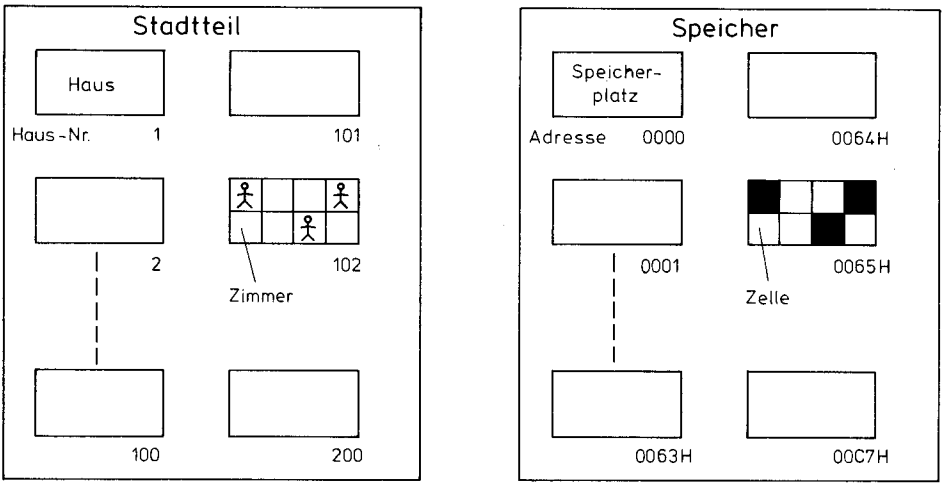


Abb. 2.2: Vergleich zwischen einem Stadtteil und einem Speicher

Der Programmspeicher setzt sich meistens aus einem sog. residenten Teil (Festwertspeicher, ROM) zur Aufnahme des Monitors (Kap. 4.2) und einem frei programmierbaren Teil (RAM) zur Aufnahme von Anwenderprogrammen zusammen.
Die Speicher im Mikrocomputer bestehen heute ausschließlich aus sog. Halbleiterspeichern. Darüberhinaus gibt es dann noch externe Massenspeicher zur Aufnahme sehr großer Programm- oder Datenmengen.

Die Struktur eines solchen Speichers geht aus Abb.2.2 hervor.

Ein Speicher, meist in der Form eines Halbleiterbausteines (IC), besteht aus einer großen Menge von Speicherplätzen, entsprechend den Häusern in einem Stadtteil.
So, wie die Häuser durchnummeriert sind, erhalten die Speicherplätze eine fortlaufende Adresse, welche allerdings mit 0 beginnt. Diese Adressierung wird im allgemeinen hexadezimal vorgenommen, da der Computer ja nur die den HEX-Zeichen zugeordneten binären Tetraden 'verstehen' kann.

Bei 8 bit-Mikroprozessoren enthält jeder Speicherplatz 8 Zellen, den 8 Zimmern eines Hauses vergleichbar. Jede Zelle kann 1 bit (L oder H) speichern, äquivalent dazu, ob ein Zimmer bewohnt (H) oder unbewohnt (L) ist.
Der Speicherplatz mit der Adresse 0065H in Abb.2.4 enthält dann die Information HLLHLLHL oder hexadezimal 92.

2.3 Der Mikroprozessor 8080

Zum Verständnis der Möglichkeiten, welche bei der Erstellung eines Mikroprozessorprogrammes (Kap.3) zur Verfügung stehen, ist es erforderlich einige innere Strukturen eines Mikroprozessors kennenzulernen.

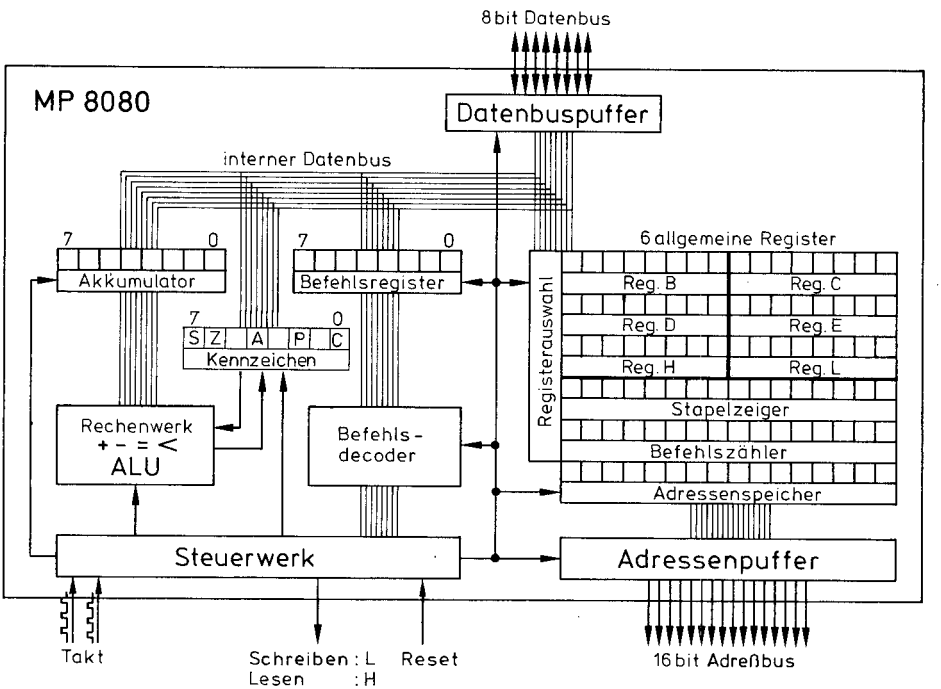


Abb. 2.3: Innerer Aufbau des Mikroprozessors 8080 in vereinfachter Darstellung s.a. Lit. (13).

Die Komponenten eines Mikroprozessors befinden sich auf einem einzelnen Silizium Chip und sind untereinander durch ein Bus-System verbunden, ähnlich wie beim Mikrocomputer. Insbesondere der externe Datenbus setzt sich ins Innere des Mikroprozessors fort. Die meisten zur Zeit verfügbaren Mikroprozessoren haben einen 8bit Datenbus, so wie der in Abb.2.3 dargestellte 8080. Sie werden deshalb auch 8bit-Mikroprozessoren genannt.

Der Adreßbus ist häufig 16bit breit, d.h. eine auf ihm parallel ausgesandte Adresse ist durch 4 HEX-Zeichen darstellbar.

Beispiel: Logischer Zustand der 16 Adreßbusleitungen

L	L	H	L	H	H	L	L	H	L	L	L	L	H	H	H

= LLHL HHLL HLLL LHHH = 2C87H

Wenn z.B. der 100. Speicherplatz in Abb.2.2 den Inhalt seiner 8 Zellen auf den Datenbus geben soll, muß der Mikroprozessor auf dem Adreßbus die Adresse 0063H aussenden.

Beinhaltet der angesprochene Speicherplatz z.B. ein Befehlswort (Byte) so wird es über den Datenbus in das Befehlsregister eingeschrieben. Der Befehlsdecoder interpretiert dann den Befehl und das Steuerwerk steuert die Befehlsabarbeitung.

Arithmetisch logische Operationen werden im Rechenwerk des Mikroprozessors durchgeführt (ALU = Arithmetic Logic Unit).

Weitere wichtige Bestandteile sind die in Abb.2.3 als eine Reihe von Kästchen dargestellten Register. Diese Kästchen symbolisieren einzelne Speicherzellen, d.h. die Register sind 8 oder 16bit Speicherplätze.

Neben den 6 allgemeinen 8bit-Registern gibt es noch das Rechenregister, den sog. Akkumulator (Akku). Wie man später - bei der Beschreibung der Befehle - sehen wird, nimmt der Akku eine Sonderstellung unter den Registern ein.

Jeweils zwei der allgemeinen Register lassen sich zu sog. Registerpaaren zusammenfassen, auf diese Weise erhält man die drei 16bit Register BC, DE und HL.

Zwei Sonderregister von doppelter Bitbreite sind das Befehlszählregister (program counter, PC) und der Stapelzeiger (stack pointer, SP). Die Bedeutung dieser Register wird später beschrieben.

Befehlsablauf

Das Steuerwerk im Mikroprozessor sorgt für den richtigen zeitlichen Ablauf aller Vorgänge im Mikrocomputer. Hierzu benötigt es einen von außen vorgegebenen Takt, welcher meistens über einen Schwingquarz erzeugt wird und deshalb sehr frequenzgenau ist.

Im Rhythmus dieses Taktes werden dann aus dem Programmspeicher Befehlsbytes in den Mikroprozessor geholt und abgearbeitet.

Die verschiedenen Befehle können aus einer unterschiedlichen Anzahl von Bytes bestehen und auch in ihrer Komplexität variieren. Deshalb erfordern die einzelnen Befehle zwischen 4 bis 18 Takte (8080) zum Einholen und Abarbeiten (Kap.3.2).

Wenn die Taktfrequenz z.B. f = 2MHz beträgt ergibt sich eine Befehlszykluszeit von 2 bis 9µs.

Beispiel: Eine 8bit Addition benötigt 4 Takte.
Taktzeit $T = 1/f = 5 \cdot 10^{-7}s$, $4T = 2\mu s$.
Das ergibt 500000 Additionen/s.

Wenn ein Mikrocomputer eingeschaltet wird, sorgt eine spezielle Schaltung (Einschaltreset) dafür, daß der Mikroprozessor immer in gleicher Weise seine Arbeit beginnt.

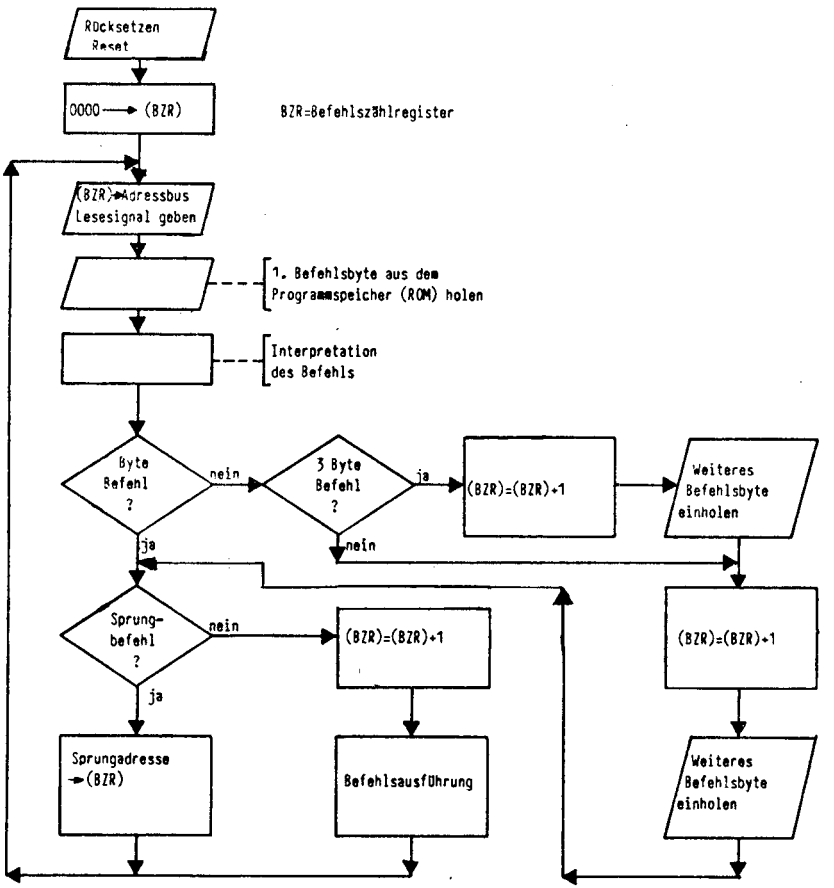


Abb. 2.4: Vereinfachte Darstellung des Befehlsablaufes im MP 8080.
(...) bedeutet, Inhalt von ...

Zunächst muß sich der Mikroprozessor seinen ersten Befehl holen, damit er mit einer sinnvollen Tätigkeit anfangen kann. Aus diesem Grunde sendet der Mikroprozessor zu Beginn die Adresse des 1. Befehlsbytes und ein Lesesignal aus. Diese Startadresse wird nach jedem Reset vom Mikroprozessor selbständig zunächst ins Befehlszählregister (BZR) geschrieben und dann der Inhalt des BZR auf den Adreßbus gegeben. Beim MP 8080 hat diese zuerst anstehende Adresse immer den Wert 0000, d.h. das erste Befehlsbyte muß stets im ersten Programmspeicherplatz stehen.

Das erste Byte eines jeden Befehles enthält den Operationsteil, durch den mit Hilfe des Befehlsdecoders dem Steuerwerk mitgeteilt wird, welche Operation durchzuführen ist und ob noch weitere Bytes zu diesem Befehl dazu gehören. Wenn es sich um einen 2 oder 3 Byte-Befehl handelt, erhöht der Mikroprozessor den Inhalt des Befehlszählregisters um 1 und sendet ihn erneut auf den Adreßbus. Hierdurch wird dann der zweite Speicherplatz angesprochen und das 2. Byte eingeholt. Falls erforderlich, wird auf die gleiche Weise dann noch das 3. Byte eingeholt.

Wenn der Mikroprozessor seinen ersten Befehl erhalten hat, führt er ihn aus. Dann holt er sich auf gleiche Weise den nächsten Befehl.

Bei einem sog. linearen Programm holt der Mikroprozessor Byte für Byte und damit auch Befehl für Befehl aus dem Speicher. Hierzu wird immer nur das Befehlszählregister (eigentlich Befehlsbyte-Zählregister) um 1 erhöht und auf den Adreßbus gesandt.

Beispiel:

Befehlszählreg. --► Adreßbus	Datenbus --► Befehlsdecoder	Funktion des Befehles
0000	31	Lade den
0001	00	Stapelzeiger
0002	11	mit 1100 H
0003	3E	Lade den
0004	81	Akku mit 81 H
⋮	⋮	
Speicher Adresse	Speicher Inhalt	

Liegt jedoch ein sog. verzweigtes Programm vor, d.h. sollen Sprünge ausgeführt werden, so wird die Sprungzieladresse ins Befehlszählregister geschrieben, da das ja die Adresse des nächsten zu bearbeitenden Befehles ist.

Beispiel:

Befehlszählreg. --► Adreßbus	Datenbus --► Befehlsdecoder	Funktion des Befehles
010E	C3	01D0 --► ⟨BZR⟩
010F	D0	Springe zur
0110	01	Adresse 01D0
⋮	⋮	
01D0	3C	⟨A⟩ = ⟨A⟩ + 1
01D1	2F	negiere ⟨A⟩
⋮	⋮	

⟨...⟩ bedeutet, Inhalt von ...; A = Akku;
BZR = Befehlszählregister.

2.4 Übungsaufgaben

2.1 Was verstehen Sie unter den Begriffen ROM, RAM, BUS, Chip Select, Register, Akku?

2.2 Welche verschiedene Busse kennen Sie und wozu dienen sie?

2.3 Aus wievielen Tetraden besteht der Adreßbus und wieviel verschiedene Adressen sind darstellbar?

2.4 Was ist ein Single Chip Computer?

2.5 Welche Adresse hat der einhundertste Speicherplatz in einem Speicher?

2.6 Wieviel Zellen hat ein Speicherplatz beim MP 8080?

2.7 Wieviel allgemeine Register hat der MP 8080?

2.8 Wieviel Bit hat das Registerpaar DE beim MP 8080?

2.9 Welche Zahl steht nach der Ausführung eines Befehles im Befehlszählregister?

2.10 Welche Operationen führt das Rechenwerk durch?

2.11 Was macht der MP 8080 nach einem Reset?

2.12 Auf welcher Adresse muß das erste Byte des ersten Befehles beim MP 8080 stehen?

2.13 Wieviel Verschiebebefehle (MOV, 5 Takte) können - bei 2MHz Takt - pro Sekunde ausgeführt werden?

3. Programmerstellung und Abarbeitung

Aufbauend auf den Kenntnissen über die Darstellung und den Ablauf von Befehlen in einem Mikroprozessor, sowie dessen innerer Struktur, soll nun dargelegt werden, wie mittels dieser Befehle Programme erstellt werden können und wie der Mikroprozessor sie abarbeitet. Über die allgemeine und grundsätzliche Erörterung hinaus wird hier immer der Mikroprozessor 8080, bzw. der weiterentwickelte Typ 8085, als Beispiel herangezogen.

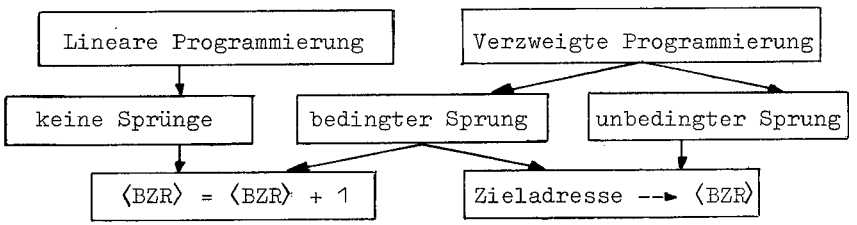


Abb.3.1: Programmierungsarten (BZR = Befehlszählregister)

Doch zuvor soll anhand von Abb.3.1 nocheinmal kurz auf die lineare und die verzweigte Programmierung eingegangen werden.

Eine Verzweigung in einem Programm wird immer durch Sprünge erreicht, d.h. anstelle der nächsten Befehlsadresse wird die des Sprungzieles in das Befehlszählregister geschrieben. Wobei zu diesen Sprüngen auch Unterprogrammaufrufe und Unterprogrammrücksprünge (Kap.3.4) gehören.

Neben den unbedingten Sprüngen, welche immer ausgeführt werden, gibt es jedoch auch noch die bedingten Sprünge. Bei einem bedingten Sprungbefehl wird dieser nur ausgeführt, wenn eine bestimmte Bedingung erfüllt ist, andernfalls wird das Programm linear weiter abgearbeitet.

Die bedingten Sprungbefehle stellen ein besonders wichtiges Hilfsmittel bei der Programmierung dar.

3.1 Die Kennzeichenbits

Bei der Bearbeitung von Befehlen können verschiedene Ereignisse auftreten, wie z.B., daß das Ergebnis einer arithmetischen Operation Null ist oder eine zu große Zahl entsteht. Ein Mikroprozessor hat deshalb eine Reihe von Speicherzellen durch die er sich das Eintreten solcher Ereignisse merken kann. Für jedes unterschiedliche Ereignis ist eine bestimmte Speicherzelle zuständig. Wenn das Ereignis eintritt, wird ein H in die Speicherzelle geschrieben, andernfalls ein L. Das Einschreiben von H in die Speicherzelle wird 'setzen' des Kennzeichenbits

(engl. flag) genannt. Wird dagegen ein L eingeschrieben, weil das Ereignis nicht eingetreten ist, so spricht man vom 'rücksetzen' oder 'nicht setzen' des betreffenden Kennzeichenbits. Diese Kennzeichenbits sind es nun, von deren Zustand die Ausführung eines bedingten Sprunges abhängig gemacht wird.

In Abb.2.3 ist zu sehen, daß der MP 8080 insgesamt 5 Zellen in seinem Kennzeichenregister hat, d.h. er kennt 5 verschiedene Kennzeichen. Diese Kennzeichen, oder auch Flags, sollen im folgenden kurz beschrieben werden.

Null (Zero) Kennzeichen: Z

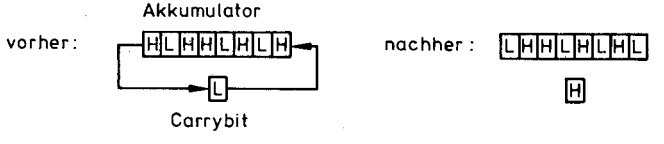
Das Ergebnis eines Befehles ist 0 : H --► ⟨Z⟩
Das Ergebnis ist ungleich 0 : L --► ⟨Z⟩

Übertrag (Carry) Kennzeichen: C

Beispiele: Addition zweier 8bit Registerinhalte.

```
        | HLHHLLHH
      + | LHHHLHLL
      H | HHH      Übertrag
⟨C⟩ ◄-- H | LLHLLHHH Ergebnis länger als 8bit.
```

Rotation des Akkumulators.



Vorzeichen (Sign) Kennzeichen: S

Das Bit, welches im Byte ganz links steht (2^7), kann als Vorzeichenbit benutzt werden (Kap.1.2). Deshalb wird dieses Bit in den Vorzeichenspeicher geschrieben.
Anstelle der Zahlen von 0 bis 255 lassen sich dann die Zahlen - 127 bis + 127 durch ein Byte darstellen.

Parität (Parity) Kennzeichen: P

Eine gerade Anzahl von Bits sind H:H --► ⟨P⟩
Eine ungerade Anzahl von Bits sind H:L --► ⟨P⟩
Beispiele: LHLLHLHH:H --► ⟨P⟩
HLLLLLLL:L --► ⟨P⟩

Hilfsübertrag (Auxiliary Carry) Kennzeichen: AC

Übertrag von der niederwertigen zur höherwertigen Tetrade.
Dieses Kennzeichen wird bei Dezimaloperationen benötigt (Kap.3.5)
Beispiel:

```
      LHLL HHLH
      LLHL HLLH
⟨AC⟩◄--H   H
      LHHH LHHL
```

Die Kennzeichen werden bei der Abarbeitung eines Befehles gesetzt oder nicht gesetzt. Dies bedeutet jedoch nicht, daß jeder Befehl die Kennzeichenbits beeinflußt. Es gibt vielmehr eine Reihe von Befehlsarten, welche nur einige der Kennzeichenbits oder auch gar keine beeinflussen. In diesen Fällen bleiben die alten Zustände der Kennzeichenspeicher erhalten.

Beispiel:

Befehlsbeschreibung	Register und Kennzeicheninhalte nach der Befehlsausführung						
	A	B	Z	C	S	P	AC
1. Eingabekanal --> ⟨A⟩	LLLLHLLL	X	X	X	X	X	X
2. 5 --> ⟨B⟩	LLLLHLLL	LLLLLHLH	X	X	X	X	X
3. ⟨A⟩ = ⟨A⟩ + ⟨B⟩	LLLLHHLH	LLLLLHLH	L	L	L	L	L
4. Springe auf den ersten Befehl, wenn ⟨C⟩ = H	LLLLHHLH	LLLLLHLH	L	L	L	L	L

X bedeutet: Inhalt unbekannt. Nur der 3. Befehl beeinflußt die Kennzeichenbits. Der Sprungbefehl (4.) wird nicht ausgeführt.

3.2 Die Befehle

Der Mikroprozessor 8080 bzw. 8085 verfügt über Befehle unterschiedlicher Länge.
Ein einfacher Verschiebebefehl - z.B. kopiere den Inhalt des Registers B in das Register E - besteht nur aus einem Byte. Wenn dagegen in ein Register eine Zahl eingeladen werden soll, muß dem ersten Byte noch das einzuladene Byte als zweites folgen. Der längste Befehl besteht aus drei Bytes. Dies ist z.B. bei einem Sprungbefehl der Fall, da dem ersten Byte eine 16bit Adresse (2 Byte) folgen muß.

Bitmuster	Hex-Code	Mnemonic-Code
	C3 34 12	JMP 1234_H

Abb. 3.2: Der Befehl "Springe zum Befehl mit der Adresse 1234 H" in seinen drei Darstellungsformen.

Der in Abb.3.2 dargestellte Sprungbefehl zeigt die Aufteilung der Befehlsinformation auf die drei Bytes. Nach dem Operationsbyte (Operationsteil) folgt zunächst das niederwertige Adreßbyte und dann das höherwertige Adreßbyte. Die beiden Adreßbytes stehen also vertauscht in Bezug auf die normale Stellenschreibweise einer Zahl. Dies wird besonders deutlich, wenn man das Befehls-Bitmuster in die entsprechenden

HEX-Zeichen umformt und diese dann mit der Adresse im sog. Mnemonic-Code vergleicht.

Hex-Code

Als Hex-Code eines Befehles wird die Umwandlung der, den Befehl repräsentierenden, Dualzahl in eine Hexadezimalzahl verstanden (Kap.1.2).

Dieser Code ist besser les- und handhabbar als der Binär-Code.

Zur Eingabe eines Programmes im Hex-Code reicht eine Rastatur mit 16 Tasten aus. Jeder Taste ist dann ein HEX-Zeichen zugeordnet (Kap.10.1) Einfache Mikrocomputer verfügen meistens über solch eine Tastatur und eine mehrstellige Anzeigeeinheit, auf der sich HEX-Zeichen darstellen lassen.

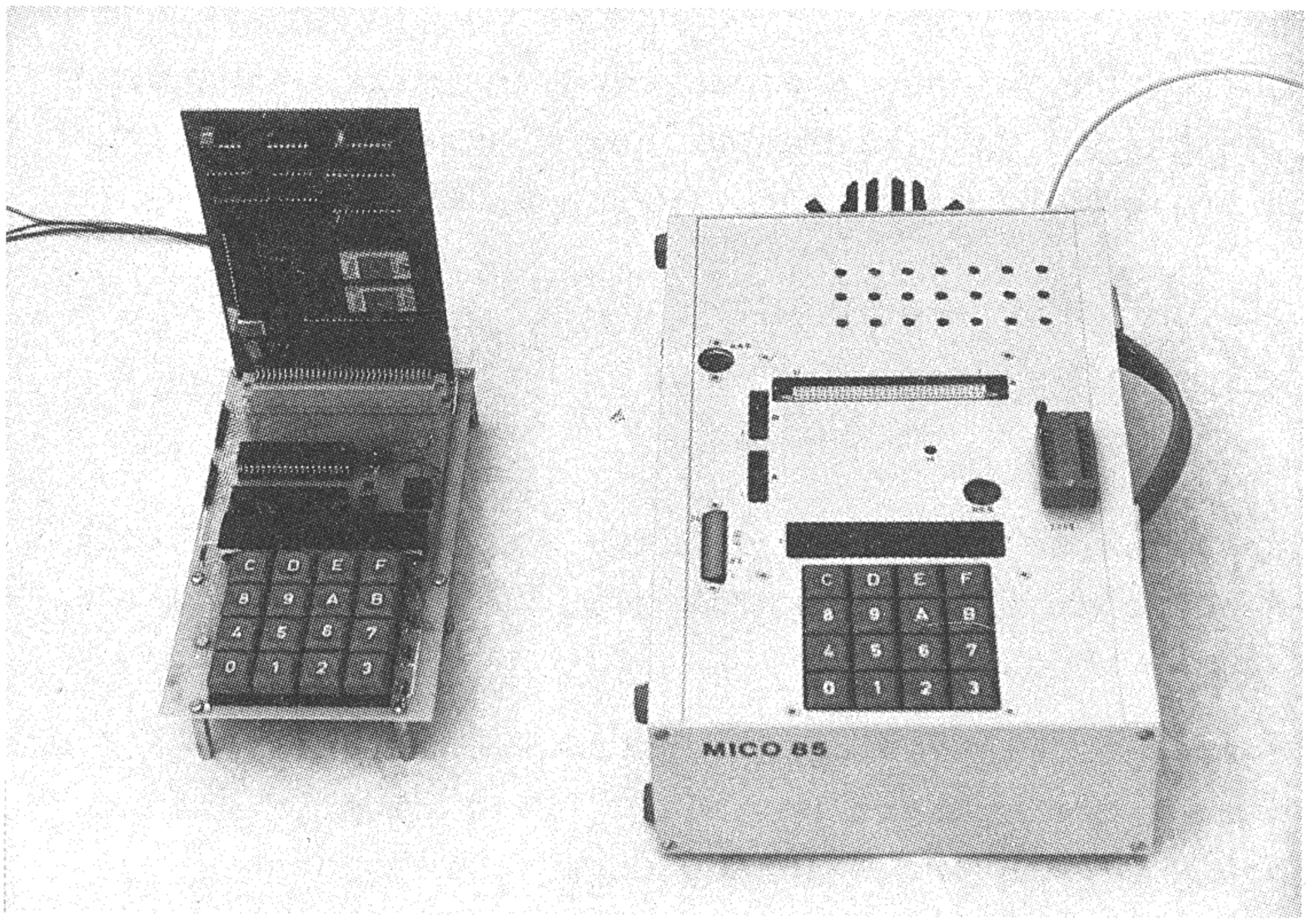


Abb. 3.3: Zwei Beispiele für Mikrocomputer, die im Hex-Code programmierbar sind.

Das Programmieren im Hex-Code wird jedoch nur zu Lehrzwecken, bei einfachen Problemen, oder bei kleinen Programmkorrekturen angewandt. Zur Erstellung längerer Programme sollte man sich des sog. Mnemonic-Codes bedienen.

Mnemonic-Code

Beim Mnemonic-Code (Mneme - Gedächtnis) wird der Operationsteil eines Befehles durch eine Buchstabenkombination ersetzt, die sich aus der Befehlsfunktion herleitet.

Bei 2- oder 3Byte-Befehlen werden die zusätzlichen Bytes im Hex-Code angehängt. Eine Vertauschung des niederwertigen und höherwertigen Bytes wird nicht vorgenommen.

```
Beispiele:   JMP 12D0   ; Springe (jump) zur Adresse 12D0
             MOV B,C    ; Kopiere den Inhalt des Registers C ins Reg. B
             ADI 3E     ; Addiere zum Akku 3E
```

Zur Eingabe dieser Befehle in den Speicher eines Mikrocomputers, ist eine Tastatur wie bei einer Schreibmaschine erforderlich, denn es müssen ja die mnemonischen Abkürzungen eingegeben werden. Die Buchstaben des Mnemonic-Teils werden deshalb durch die Eingabetastatur im ASCII-Code (Kap.1.3) an den Mikrocomputer weitergegeben.

Beispiel: JMP 12D0 ---► 4A 4D 50 20 31 32 44 30

Die Befehle, die auf diese Weise in den Speicher eines Mikrocomputers gelangen, müssen anschließend mit Hilfe eines speziellen, im Mikrocomputer bereits vorhandenen, Programmes in den Binär-Code umgewandelt werden. Häufig spricht man hierbei auch von einer Umwandlung in den Hex-Code, da dieser ja unmittelbar dem Binär-Code zugeordnet ist und eigentlich nur eine andere Schreibweise darstellt.

```
Beispiele:   JMP 12D0 --►  C3D012
             MOV B,C  --►  41
             ADI 3E   --►  C63E
```

Dieses Übersetzen in den Hex-Code kann entweder automatisch nach der Eingabe eines jeden Befehles geschehen, oder aber nachdem das gesamte Programm eingegeben wurde.

Die Übersetzung erfolgt meistens anhand einer Tabelle, die sich im Speicher des Mikrocomputers befindet. Diese Tabelle enthält den zu jeder mnemonischen Abkürzung zugehörigen Hex-Code.

Bei den meisten Befehlen kommen zu dem Mnemonic-Code noch weitere Angaben hinzu. Diese machen Aussagen über betroffene Register, über Daten oder Adressen.

Diesen zweiten Teil nennt man die Operanden des Befehles.

```
Beispiele:   Mnemonic-Code (Operationscode)
                 Operanden
                                      Kommentare
             MOV  A,B    ; Kopiere den Inhalt von B in den Akku
             OUT  18     ; Gebe den Inhalt des Akkus auf den Port 18
             CMP  E      ; Vergleiche den Inhalt von E mit A
             MVI  C,30   ; Bringe das Datum 30H ins Register C
```

Eine Erweiterung des Mnemonic-Codes stellt die Assembler-Programmiersprache dar. Hierauf wird im Kap. 5.3 näher eingegangen.

Befehlsliste des MP 8080/85

In der Abb.3.4 sind alle Befehle des MP 8080 und die zusätzlichen Befehle des MP 8085 enthalten.

Von den zusätzlichen Befehlen des MP 8085 (Abb.3.4d 9.) sind nur die ersten 2 offiziell bekannt. Die 10 weiteren Befehle sind zwar zur Zeit vorhanden, es ist jedoch nicht auszuschließen, daß sie im Laufe der Zeit vom Hersteller modifiziert werden. Da der MP 8085, genau wie der MP 8080, von vielen verschiedenen Herstellern angeboten wird, kann es sogar langfristig, bezüglich der zusätzlichen 10 Befehle, zu unterschiedlichen Veränderungen kommen.

In der ersten Spalte der Befehlsliste sind die sog. Mnemonics, d.h. der Mnemonic-Code der Befehle, in der international gebräuchlichen Form, dargestellt. Die zweite Spalte gibt jeweils an, aus wieviel Byte der Befehl besteht.

Der Hex-Code des Befehles, oder ein Verweis (s.T.) auf die rechts am Rand stehenden Tabellen, ist in der Spalte 3 angegeben. Diese zweistellige Hex-Zahl stellt jeweils den Operationsteil des Befehles dar. In den Fällen, wo im ersten Befehlsbyte bereits variable Adressierungen enthalten sind, geben die zugehörigen Tabellen den entsprechenden Hex-Code wieder.

Beispiele: MOV A,B --► 78 ; MVI H, konst. --► 26..

Die Spalte 4 enthält die Abkürzungsbuchstaben derjenigen Kennzeichenbits (Flags), die bei der Befehlsausführung beeinflußt werden.

In den Spalten 5 und 7 sind die Anzahl der Taktzyklen wiedergegeben, die zur Abarbeitung des Befehles beim MP 8080 bzw. beim MP 8085 erforderlich sind. Bei bedingten Sprüngen oder Unterprogrammaufrufen, variiert die erforderliche Anzahl der Taktzyklen in Abhängigkeit davon, ob die Bedingung erfüllt ist oder nicht. Wenn eine Bedingung nicht erfüllt ist, wird ein bedingter Sprung auch nicht ausgeführt und der Mikroprozessor braucht die Sprungadresse nicht einzulesen. Hierdurch wird die Befehlsausführungszeit kürzer.

Die Spalte 6 enthält eine Kurzbeschreibung des Befehles. Einige der Befehle werden im folgenden noch näher erklärt werden. Eine detaillierte Beschreibung aller Befehle findet man in den Datenbüchern der Hersteller.

Demjenigen Leser, der selbst Mikroprozessorprogramme erstellen möchte, wird dringend empfohlen sich die Kurzbeschreibung aller Befehle in Abb.3.4 anzusehen. Nur wenn man einen Überblick über die Möglichkeiten gewinnt, die sich bei der Anwendung der verschiedensten Befehle eröffnen, ist ein effektives Programmieren möglich.

BEFEHLSLISTE DES 8080/85

Mnemonic	Bytes	Hex. Code	Flagbeeinflussung *	Taktzyklen	Funktion des Befehls	Taktzyklen 8085
1. TRANSFERBEFEHLE						
1.1 Register→Register						
MOV r_1,r_2	1	s.T.	--	5	r_1,r_2 = A,B,C,D,E,H,L . Lade Reg. r_1 mit dem Inh. von Reg. r_2	4
XCHG	1	EB	--	4	Vertausche Inhalte der Registerpaare (D,E) und (H,L)	4
XTHL	1	E3	--	18	Vertausche Inh. d. Reg.-Paares (H,L) u. d. Inh. d. Wortes, das durch den Stapelzeiger adressiert ist	16
SPHL	1	F9	--	5	Lade Stapelzeiger mit dem Inh. d. Reg.-Paares (H,L)	6
1.2 Speicher, Peripherie→Register						
MOV r_1,M	1	s.T.	--	7	r_1 = A,B,C,D,E,H,L. Lade Reg. r_1 mit d. Inh. d. Speicherplatzes,das durch d. Inh. d. Reg.-Paares (H,L) adressiert ist	7
LDA adr	3	3A	--	13	Accu laden mit Inh. der Adresse adr	13
LDAX rp	1	s.T.	--	7	rp = B,D. Accu laden mit Inh. d. Speicherplatzes, der durch den Inh. d. Reg.-Paares rp adressiert ist	7
LHLD adr	3	2A	--	16	Lade Reg.-Paar (H,L) mit d. Inh. der Adressen (adr+1) und adr	16
POP rp PSW	1	s.T. F1	Z,S,P,C,A	10	rp = B,D,H,PSW. Reg.-Paar rp wird mit d. Wort geladen, das durch d. Stapelzeiger adressiert ist (PSW = Accu + Flags)	10
IN nr	2	DB	--	10	Accu wird mit d. Inh. d. Eingabekanals Nummer nr (nr≤255) geladen	10
1.3 Konstante→Registerpaar						
LXI rp,adr	3	s.T.	--	10	rp = B,D,H,SP. Lade Reg.-Paar rp mit wert adr	10
1.4 Register→Speicher, Peripherie						
MOV M,r_1	1	s.T.	--	7	r_1 = A,B,C,D,E,H,L. Inh. v. Reg. r_1 auf d. Speicherplatz abspeichern, d. durch d. Inh. d. Reg.-Paares (H,L) adressiert ist	7
STA adr	3	32	--	13	Accu-Inh. unter Adresse adr abspeichern	13
STAX rp	1	s.T.	--	7	rp = B,D. Accu in dem Byte abspeichern, das durch den Inh. d. Reg.-Paares rp_1 adressiert ist	7
SHLD adr	3	22	--	16	Reg.-Paar (H,L) unter (adr+1) und adr abspeichern	16
PUSH rp	1	s.T.	--	11	rp = B,D,H,PSW. Inh. d. Reg.-Paares rp wird in das Wort übertragen, das durch den Stapelzeiger adressiert ist	12
OUT nr	2	D3	--	10	Accu-Inh. wird auf Ausgabekanal (Nummer nr ≤ 255) ausgegeben	10
1.5 Konstante→Register						
MVI M ,konst	2	36	--	10	Bringe d. Wert d. Konstanten (konst ≤ 255) in d. Speicherplatz, der durch d. Inh. d. Reg.-Paares (H,L) adressiert ist	10
MVI r_1, konst	2	s.T.	--	7	r_1 = A,B,C,D,E,H,L. Lade Reg. r_1 mit Konstante (konst≤255)	7

MOV	A	B	C	D	E	H	L	M
A	7F	78	79	7A	7B	7C	7D	7E
B	47	40	41	42	43	44	45	46
C	4F	48	49	4A	4B	4C	4D	4E
D	57	50	51	52	53	54	55	56
E	5F	58	59	5A	5B	5C	5D	5E
H	67	60	61	62	63	64	65	66
L	6F	68	69	6A	6B	6C	6D	6E
M	77	70	71	72	73	74	75	

z.B. → MOV M,D (D)→(M)

LDAX	B	D
	0A	1A

POP	B	D	H	PSW
	C1	D1	E1	F1

LXI	D16 ← 16Bit Datenwort
B	01
D	11
H	21
SP	31

STAX	B	D
	02	12

PUSH	B	D	H	PSW
	C5	D5	E5	F5

MVI	D8
A	3E
B	06
C	0E
D	16
E	1E
H	26
L	2E
M	36

* A = Auxiliary Carry (HilfsÜbertrag) , Z = Zero (Null) , S = Sign (Vorzeichen) , P = Parity (Parität) , C = Carry (Übertrag)

Abb. 3.4 a:

2. ARITHMETISCHE OPERATIONEN

INR r_1	1	s.T.	Z,S,P,A	5	r_1 = A,B,C,D,E,H,L. Zum Inh. d. Reg. r_1 wird 1 addiert	4
INR M	1	s.T.	Z,S,P,A	10	Zum Inh. d. durch Reg.-Paar (H,L) adressierten Bytes wird 1 addiert	10
DCR r_1	1	s.T.	Z,S,P,A	5	r_1 = A,B,C,D,E,H,L. Vom Inh. d. Reg. r_1 wird 1 subtrahiert	4
DCR M	1	35	Z,S,P,A	10	Vom Inh. d. durch Reg.-Paar (H,L) adressierten Bytes wird 1 subtrahiert	10
INX rp	1	s.T.	--	5	rp = B,D,H,SP. Der Inh. d. Reg.-Paares rp_1 wird um 1 erhöht	6
DCX rp	1	s.T.	--	5	rp = B,D,H,SP. Der Inh. d. Reg.-Paares rp_1 wird um 1 erniedrigt	6
ADD r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Inh. v. Reg. r_1 wird zum Inh. des Accus addiert	4
ADD M	1	86	Z,S,P,C,A	7	Der Inh. d. Speicherbytes, das durch d. Inh. d. Reg.-Paares (H,L) adressiert ist, wird zum Inh. d. Accus addiert	7
ADC r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Inh. d. Reg. r_1 u. Inh. d. Carry-Bits werden zum Inh. des Accus addiert	4
ADC M	1	8E	Z,S,P,C,A	7	Inh. d. Speicherbytes,das durch d. Inh. d. Reg.-Paares (H,L) adressiert ist u. d. Inh. d. Carry-Bits werden zum Inh. d. Accus addiert	7
DAD rp	1	s.T.	C	10	rp = B,D,H,SP. Inh. d. Reg.-Paares rp_1 u. d. Inh. d. Reg.-Paares (H,L) werden addiert. Ergebnis in (H,L)	10
SUB r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Inh. d. Reg. r_1 wird v. Accu-Inh. subtrahiert	4
SUB M	1	96	Z,S,P,C,A	7	Inh. d. Speicherbytes, das durch d. Reg.-Paar (H,L) adressiert ist, wird vom Accu subtrahiert	7
SBB r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Inh. v. Reg. r_1 u. Inh. des Carry-Bits werden v. Accu-Inh. subtrah.	4
SBB M	1	9E	Z,S,P,C,A	7	Inh. d. Speicherbytes, das durch d. Reg.-Paar (H,L) adressiert ist, u. Inh. des Carry-Bits werden vom Accu subtrahiert	7
ADI konst	2	C6	Z,S,P,C,A	7	Konstante (konst≤255) wird zum Inh. d. Accu addiert	7
ACI konst	2	CE	Z,S,P,C,A	7	Zum Accu-Inh. werden konst≤255 u. Carry-Bit addiert	7
SUI konst	2	D6	Z,S,P,C,A	7	konst≤255 wird vom Accu-Inh. subtrahiert	7
SBI konst	2	DE	Z,S,P,C,A	7	konst≤255 und das Carry-Bit werden v. Accu-Inh. subtrahiert	7
DAA	1	27	Z,S,P,C,A	4	Accu-Inh. wird in eine 2stellige Zahl umgewandelt	4

INR	A	B	C	D	E	H	L	M
	3C	04	0C	14	1C	24	2C	34

DCR	A	B	C	D	E	H	L	M
	3D	05	0D	15	1D	25	2D	35

INX	B	D	H	SP
	03	13	23	33

DCX	B	D	H	SP
	0B	1B	2B	3B

ADD	A	B	C	D	E	H	L	M
	87	80	81	82	83	84	85	86

ADC	A	B	C	D	E	H	L	M
	8F	88	89	8A	8B	8C	8D	8E

DAD	B	D	H	SP
	09	19	29	39

SUB	A	B	C	D	E	H	L	M
	97	90	91	92	93	94	95	96

SBB	A	B	C	D	E	H	L	M
	9F	98	99	9A	9B	9C	9D	9E

3. LOGISCHE OPERATIONEN

CMA	1	2F	--	4	Accu-Inh. wird negiert	4
ANA r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Accu-Inh. u. d. Inh. d. Reg. r_1 werden UND-verknüpft	4
ANA M	1	A6	Z,S,P,C,A	7	Der Inh. d. durch Reg.-Paar (H,L) adressierten Bytes wird mit d. Accu-Inh. UND-verkn.	7
ANI konst	2	E6	Z,S,P,C,A	7	Accu-Inh. wird mit der konst≤255 UND-verknüpft	7
ORA r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Accu-Inh. wird mit d. Inh. d. Reg. r_1 ODER-verknüpft	4
ORA M	1	B6	Z,S,P,C,A	7	Inh. d. Über Reg.-Paar (H,L) adressierten Bytes wird mit d. Accu-Inh. ODER-verknüpft	7
ORI konst	2	F6	Z,S,P,C,A	7	Accu-Inh. wird mit konst≤255 ODER-verknüpft	7
XRA r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Accu-Inh. wird mit d. Inh. d. Reg. r_1 Exclusiv-ODER-verknüpft	4
XRA M	1	AE	Z,S,P,C,A	7	Das Über Reg.-Paar (H,L) adressierte Byte wird durch Exclusive-ODER mit d. Accu verkn.	7
XRI konst	2	EE	Z,S,P,C,A	7	Der Accu wird mit d. Wert konst≤255 durch Exclusive-ODER verknüpft	7
CMP r_1	1	s.T.	Z,S,P,C,A	4	r_1 = A,B,C,D,E,H,L. Accu-Inh. wird mit d. Inh. d. Reg. r_1 verglichen	4
CMP M	1	BE	Z,S,P,C,A	7	Accu-Inh. wird mit d. Inh. d. durch Reg.-Paar (H,L) adressierten Bytes verglichen	7
CPI konst	2	FE	Z,S,P,C,A	7	Accu-Inh. wird mit konst≤255 verglichen	7

ANA	A	B	C	D	E	H	L	M
	A7	A0	A1	A2	A3	A4	A5	A6

ORA	A	B	C	D	E	H	L	M
	B7	B0	B1	B2	B3	B4	B5	B6

XRA	A	B	C	D	E	H	L	M
	AF	A8	A9	AA	AB	AC	AD	AE

CMP	A	B	C	D	E	H	L	M
	BF	B8	B9	BA	BB	BC	BD	BE

Abb. 3.4 b:

Mnemonic	Bytes	Hex. Code	Flagbeeinflussung *	Taktzyklen	Funktion des Befehls	Taktzyklen 8085
4. REGISTERANWEISUNGEN						
4.1 Akkumulator rotieren						
RLC	1	07	C	4	Accu-Inh. wird zyklisch um 1 Bit nach links verschoben. Bit 2^7 wird in das Carry-Bit geschrieben	4
RRC	1	0F	C	4	Accu-Inh. wird zyklisch um 1 Bit nach rechts verschoben. Bit 2^0 wird in das Carry-Bit geschrieben	4
RAL	1	17	C	4	Accu-Inh. wird um 1 Bit nach links geschoben. Bit 2^7 wird in das Carry-Bit und das Carry-Bit in das Bit 2^0 geschrieb.	4
RAR	1	1F	C	4	Accu-Inh. wird um 1 Bit nach rechts geschoben. Bit 2^0 wird in das Carry-Bit und das Carry-Bit in das Bit 2^7 geschrieb.	4
4.2 Übertragsbit-Anweisungen						
CMC	1	3F	C	4	Carry-Bit wird negiert	4
STC	1	37	C	4	Carry-Bit wird gesetzt	4
5. SPRUNGBEFEHLE						
5.1 Unbedingte Sprünge						
PCHL	1	E9	--	5	Programm wird an der Adresse fortgesetzt, die im Reg.-Paar (H,L) steht	6
JMP adr	3	C3	--	10	Programm wird an der Adresse fortgesetzt	10
5.2 Bedingte Sprünge						
JC adr	3	DA	--	10	Bei Carry-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	7/10
JNC adr	3	D2	--	10	Bei Carry-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	7/10
JZ adr	3	CA	--	10	Bei Zero-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	7/10
JNZ adr	3	C2	--	10	Bei Zero-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	7/10
JM adr	3	FA	--	10	Bei Sign-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	7/10
JP adr	3	F2	--	10	Bei Sign-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	7/10
JPE adr	3	EA	--	10	Bei Paritäts-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	7/10
JPO adr	3	E2	--	10	Bei Paritäts-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	7/10

→ Bedingung erfüllt
→ Bedingung nicht erfüllt

6. UNTERPROGRAMMBEHANDLUNG

6.1 Programmaufrufe

Bei allen Aufrufbefehlen wird die RÜckkehradresse im Stapel abgelegt

CALL adr	3	CD	--	17	Programm wird bei der Adresse adr fortgesetzt	18
CC adr	3	DC	--	11/17	Bei Carry-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	9/18
CNC adr	3	D4	--	11/17	Bei Carry-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	9/18
CZ adr	3	CC	--	11/17	Bei Zero-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	9/18
CNZ adr	3	C4	--	11/17	Bei Zero-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	9/18
CM adr	3	FC	--	11/17	Bei Sign-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	9/18
CP adr	3	F4	--	11/17	Bei Sign-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	9/18
CPE adr	3	EC	--	11/17	Bei Paritäts-Bit = 1 wird das Programm bei der Adresse adr fortgesetzt	9/18
CPO adr	3	E4	--	11/17	Bei Paritäts-Bit = 0 wird das Programm bei der Adresse adr fortgesetzt	9/18
RST konst	1	s.T.	--	11	$0 \leq konst \leq 7$ Programm wird auf der Adresse 8xkonst fortgesetzt	12

RST	0	1	2	3	4	5	6	7
	C7	CF	D7	DF	E7	EF	F7	FF

Abb. 3.4 c:

6.2 Rücksprungbefehle

RET	1	C9	--	10	Programm wird an der Adresse fortgesetzt, die in dem Wort steht, das über den Stapelzeiger adressiert ist	10
RC	1	D8	--	5/11	Bei Carry-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12
RNC	1	D0	--	5/11	Bei Carry-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12
RZ	1	C8	--	5/11	Bei Zero-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12
RNZ	1	C0	--	5/11	Bei Zero-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12
RM	1	F8	--	5/11	Bei Sign-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12
RP	1	F0	--	5/11	Bei Sign-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12
RPE	1	E8	--	5/11	Bei Paritäts-Bit = 1 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12
RPO	1	E0	--	5/11	Bei Paritäts-Bit = 0 wird das Programm an der Adresse fortgesetzt, die in dem über dem Stapelzeiger adressierten Wort steht	6/12

7. PROGRAMMUNTERBRECHUNG

EI	1	FB	--	4	INTE-Flipflop wird gesetzt, der Mikroprozessor kann Unterbrechung annehmen	4
DI	1	F3	--	4	INTE-Flipflop wird rückgesetzt, der Mikroprozessor ignoriert Unterbrechungsanforderungen	4

8. SONSTIGE BEFEHLE

HLT	1	76	--	7	Programm hält an, bis eine Unterbrechungsanforderung eintrifft	5
NOP	1	00	--	4	Leerbefehl	4

9. ZUSÄTZLICHE 8085 BEFEHLE

RIM	1	20	--	4	Lesen der Unterbrechungsmaske incl. SID	SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5	
SIM	1	30	--	4	Setzen der Unterbrechungsmaske incl. SOD	SOD	SOE	X	R7.5	MSE	M7.5	M6.5	M5.5	
SHLX	1	D9	--	10	Inh. (L)→Inh. $(M)_{DE}$, Inh. (H)→Inh. $(M)_{DE+1}$									
LHLX	1	ED	--	10	Inh. $(M)_{DE}$→Inh. (L) , Inh. $(M)_{DE+1}$→Inh. (H)									
LDHI	2	28	--	10	Inh. (D,E) = Inh. (H,L) + Inh. (Byte 2)									
LDSI	2	38	--	10	Inh. (D,E) = Inh. (SP) + Inh. (Byte 2)									
DSUB	1	08	Z,S,P,C,A,X5,V	10	Inh. (H,L) = Inh. (H,L) - Inh. (B,C)									
ARHL	1	10	C	7	Verschiebe Inh. (H,L) nach rechts, L_0→Carry, H_7 H_6									
RDEL	1	18	C,V	10	Verschiebe Inh. (D,E) nach links, D_7→Carry, Carry E_0									
JNX5	3	DD	--	7/10	Sprung, wenn das Flag X5 nicht gesetzt ist									
JX5	3	FD	--	7/10	Sprung, wenn das Flag X5 gesetzt ist									
RSTV	1	CB	--	5/12	Restart 40_H, wenn Überlauf (Flag V)									

ZUSÄTZLICHE FLAGS: V wird gesetzt beim Überlauf des Zweierkomplements
X5 wird bei einem Überlauf des INX-Befehles und bei einem Unterlauf des DCX-Befehles gesetzt

Abb. 3.4 d: Vollständige Befehlsliste der Mikroprozessoren 8080 und 8085. s. T. bedeutet: siehe nach, in den Tabellen am rechten Rand.

Der Anhang enthält noch eine zusammenfassende Darstellung der Funktionen aller Befehle und eine Liste der numerisch geordneten Befehle.

Programmliste

Mit Hilfe der Befehle aus Abb.3.4 können nun Mikroprozessor-Programme geschrieben werden.
Nachdem man sich über die Problemstellung klar geworden ist, kann man - evtl. unter Zuhilfenahme eines Flußdiagrammes (Kap.1.4) - an die Erarbeitung eines Lösungsweges gehen. Wie schon erwähnt, ist es hierbei wichtig, eine genaue Vorstellung von den Möglichkeiten zu haben, welche die Befehle bieten.

Gleichgültig, ob die Befehlseingabe im Hex- oder Mnemonic-Code erfolgen soll, die Programmliste wird immer zunächst im Mnemonic-Code geschrieben. Zur Erstellung des Programmes kann man eine Liste wie in Abb.3.5 benutzen.

Adr.	Hex-Code	Mnemonic-Code	Kommentare
1A00	3E1C	MVI A,1C	1C --► ⟨A⟩
1A02	0605	MVI B,05	5 --► ⟨B⟩
1A04	4F	MOV C,A	⟨A⟩ --► ⟨C⟩
1A05	05	DCR B	⟨B⟩ = ⟨B⟩ -1 (Decrement-Register)
1A06	81	ADD C	⟨A⟩ = ⟨A⟩ + ⟨C⟩
1A07	05	DCR B	⟨B⟩ = ⟨B⟩ -1
1A08	C2061A	JNZ 1A06	(Jump on no zero)Springe solange ⟨B⟩ nicht Null
1A0B	D318	OUT 18	⟨A⟩ --► Ausgabeport 18

Abb. 3.5: Beispiel für eine Programmliste (s. a. Übungsaufg. 3.10)

Zunächst schreibt man in die dritte Spalte den Mnemonic-Code eines Befehles und in die vierte Spalte eine kurze Erklärung über die Wirkung des Befehles. Die erste Spalte enthält immer die Adresse des ersten Bytes eines jeden Befehles.

Als Kommentare werden in Abb.3.5 die Befehlsfunktionen erklärt, um dem Anfänger das Lesen des Programmes zu erleichtern. Ein geübter Programmierer dagegen wird im Kommentarfeld einer Befehlszeile Erläuterungen über die Funktion des Befehles innerhalb dieses speziellen Programmes geben (s.a. Kap. 4.).

Nachdem auf diese Weise das gesamte Programm erstellt worden ist, können mit Hilfe der Befehlsliste aus Abb.3.4 die Hex-Codes der einzelnen Befehle ermittelt und in Spalte 2 eingetragen werden. Dieser Schritt ist jedoch nur erforderlich, wenn der Mikrocomputer im Hex-Code programmiert werden soll. Andernfalls übersetzt der Mikrocomputer selbst den Mnemonic-Code in den Hex-Code.
Der Hex-Code - er wird auch häufig Objektcode genannt - belegt dann eine entsprechende Anzahl von hintereinanderliegenden Speicherplätzen.

Beispiel: Belegung eines Speichers durch den Hex-Code aus Abb.3.5

Adr.	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	höherwertig
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	niederwertig
Hex-Code	3E	10	06	05	4F	05	81	05	C2	06	1A	D3	18	

Weitere Programmierbeispiele enthält Kap. 4..

3.3 Adressierungsarten

Mit Ausnahme der Steuerbefehle (NOP, HLT, EI, DI) erfordern alle anderen Befehle eine Angabe oder Festlegung darüber, wohin Daten transportiert werden sollen, oder auf welches Register oder Speicherelement er wirken soll.
Diese Festlegung von Quell- und/oder Zieladressen nennt man Adressierung.

Implizite Adressierung

Bei einer Reihe von 1Byte-Befehlen ist die Adressierung durch die Befehlsfunktion implizit gegeben. Diese Befehle brauchen deshalb auch keinen Operanden. Der Operationscode legt eindeutig das Ziel der Befehlswirkung fest.

Beispiele: STC (Set carry) --→ Bringe ein H ins Carry-Flag
DAA (Dezimal adjust accu) --→ Dezimalkorrektur des Akkus (Kap.3.5)

Registeradressierung

Die Registeradressierung tritt bei einem Großteil der Befehle auf. Hierbei handelt es sich um 1Byte-Befehle, die im Operationsteil eine Angabe über das Ziel- und/oder das Quellregister enthalten.

Register	Code
B	LLL
C	LLH
D	LHL
E	LHH
H	HLL
L	HLH
M	HHL
A	HHH

Abb.3.6: Binär-Code der Register und der Speicher (M=Memory)

Für die Adressierung dieser Register wird ein 3bit-Binär-Code verwandt, der in Abb.3.6 dargestellt ist.
Zusätzlich zu den Registern tritt hier noch der externe Speicher (Memory) des Mikroprozessors auf, er wird mit M abgekürzt. Da der externe Speicherraum des Mikroprozessors bis zu 65536 verschiedene Plätze haben kann, muß vor einer Befehlsausführung, bei der z.B. ein Datum von einem Register in einen Speicherplatz gebracht werden soll, die Adresse des Speicherplatzes ins Registerpaar HL geschrieben werden (siehe auch unter "Indirekte Adressierung").

Beispiele: Der Datentransfer Befehl MOV

MOV r_1,r_2: Ein Datenbyte wird von Reg.2 zu Reg.1 gebracht
Reg.2 bleibt unverändert.

r_1 = Zielregister

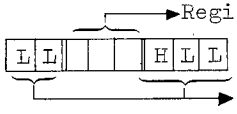
MOV r_1 r_2 r_2 = Quellregister

└Operationscode

MOV D,A: LH LHL HHH = 57H
Register-Code laut Abb.3.6

Die arithmetische Operation INR

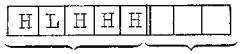
INR r: Inkrementiere den Inhalt des Reg. r; ⟨r⟩ = ⟨r⟩ + 1



Der Akkumulator - als spezielles Rechenregister - ist häufig implizit als 1. Operand vorgesehen.

Beispiel: Die logische Operation CMP

CMP r: Vergleiche (compare) den Inhalt
des Registers r mit dem Akku-Inhalt.



Operationscode r laut Abb.3.6

Unmittelbare Adressierung

Bei Befehlen, die eine unmittelbare Adressierung verwenden, werden die Daten als Teil des Befehles selbst, angegeben.

Nach dem ersten Befehlsbyte folgen deshalb ein oder zwei Bytes, die als Datum (Konstante) aufgefaßt werden.

Der Mnemonic-Code dieser Befehle endet mit einem I (für immediate).

Bis auf die Befehle MVI und LXI wird der Akkumulator implizit als 1. Operand benutzt.

Beispiel: Die arithmetische Operation ADI

ADI konst: Addiere zum Akku eine 1Byte-Konstante

ADI 5D: HHLLLHHL LHLHHHLH; ⟨A⟩ = ⟨A⟩ + 5D

Bei den MVI-Befehlen (Move immediate) wird mit dem 1. Befehlsbyte bereits festgelegt, welches Register (Abb.3.6) der 1. Operand ist.

In dieses Register läd der Mikroprozessor dann das 2. Befehlsbyte als Konstante ein.

rp	Code
BC	L L
DE	L H
HL	H L
SP	H H

Abb.3.7: Registerpaar-Codeworte, SP=Stapelzeiger (Stack Pointer)

Die LXL-Befehle gestatten das Laden eines Registerpaares (rp) mit einer 2Byte-Konstanten. Als 1. Operand muß deshalb ein Registerpaar-Codewort gemäß Abb.3.7 im 1. Befehlsbyte angegeben werden.

Beispiel:

LXI B,5D6F: LLLLLLLH LHHLHHHH LHLHHHLH = 016F5D
5D --► ⟨B⟩, 6F --► ⟨C⟩

Direkte Adressierung

Mit Ausnahme der Befehle IN und OUT - hier wird die 1Byte-Adresse intern als 2. und 3. Byte interpretiert - handelt es sich hierbei um 3Byte-Befehle, wobei das 2. und 3. Byte direkt die Adresse angeben.

Hierzu gehören z.B. alle Sprungbefehle.

Bei einer Reihe anderer Befehle ist der Akkumulator oder das Registerpaar HL implizit als 1. Operand vorgegeben.

Beispiele: Die Datentransferbefehle LDA und SHLD

LDA adr: Lade den Akku mit dem Inhalt des Speichers unter der angegebenen Adresse.

LDA 1B00: 3A 00 1B ⟨M⟩$_{1B00}$ --► ⟨A⟩

⟨M⟩$_{1B00}$ bedeutet: Inhalt des Speicherplatzes (Memory) mit der Adresse 1B00.

SHLD adr: (Store HL direct) Kopiere den Inhalt des Registerpaares HL in den Speicher mit der angegebenen Adresse.

SHLD 4C01: 22 01 4C
⟨L⟩ --► ⟨M⟩$_{4C01}$, ⟨H⟩ --► ⟨M⟩$_{4C02}$

Indirekte Adressierung

Bei der indirekten Adressierung wird die Adresse, vor der Ausführung des Befehles, durch andere Befehle in ein Registerpaar oder in zwei Speicherplätze gebracht. Der indirekt adressierte Befehl benutzt dann die Adresse, die im Registerpaar oder im Speicher steht.

Alle Befehle, bei denen als Operand ein M genannt wird, enthalten als implizite Festlegung des Adreßortes das Registerpaar HL.

Beispiel: Die logische Operation ANA

ANA M: (And memory with accu) Mache eine UND-Verknüpfung des durch HL adressierten Speicherplatzinhaltes mit dem Akku. Das Ergebnis steht im Akku.

MNE-Code	HEX-Code	⟨A⟩	⟨H⟩	⟨L⟩	$\langle M \rangle_{10B0}$	Bemerkungen
		5A	X	X	C4	
LXI H,10B0	21B010	5A	10	B0	C4	10B0 --► ⟨HL⟩
ANA M	A6	40	10	B0	C4	$\langle A \rangle = \langle A \rangle \wedge \langle M \rangle_{10B0}$

Bei anderen Befehlen wird das Registerpaar, in dem die Adresse steht, im ersten Byte - unter Benutzung der rp-Codes der Abb.3.7 - angegeben. Diese 1Byte-Befehle enthalten im Operandenteil die Abkürzung rp.

Beispiel: LDAX B: Bringe den Inhalt des Speicherplatzes, der durch das Registerpaar BC adressiert ist, in den Akku.

In beiden Beispielen ist der Akkumulator implizit als Zielregister festgelegt.

Eine Besonderheit bilden die Unterprogramm-Rücksprungbefehle (RET, s.a. Kap.3.4). In diesem Fall ist implizit festgelegt, daß in dem 16bit-Register des Stapelzeigers (SP) die Adresse steht, unter der in zwei aufeinanderfolgenden Speicherplätzen die Rücksprungadresse zu finden ist.

Indizierte Adressierung

Diese Adressierungsart kommt beim MP 8080/85 nicht vor.

Um zu einer effektiven Adresse zu kommen, wird eine im Befehl enthaltene Zahl (Displacement) zu einer Basisadresse addiert, die sich in einem sog. Indexregister befindet.

Kombinierte Adressierungsarten

Wie oben bereits häufig erwähnt, treten auch gemischte Adressierungsarten auf. Insbesondere die Befehle zum Aufruf von Unterprogramme (CALL; Kap.3.4) verwenden eine kombinierte Adressierungsart.

Die Adresse des Unterprogrammes wird in direkter Adressierung hinter das erste Byte geschrieben. Gleichzeitig wird jedoch die indirekte Adressierung benutzt, um über den Stapelzeiger die Adresse des nächsten Befehles im Speicher abzulegen. Ein Beispiel hierzu enthält das nächste Kapitel.

3.4 Unterprogramme

Beim Programmieren kommt es häufig vor, daß eine bestimmte Befehlsfolge an mehreren Stellen des Programmes benötigt wird. In solchen Fällen ist es nicht erforderlich diese Befehlsfolge an den verschiedenen Stellen des Programmes immer wieder hinzuschreiben. Diese Befehlsfolge wird in einen gesonderten Speicherbereich - z.B. ans Ende des Programmes - geschrieben und dann, an den Stellen des Programmes wo sie benötigt wird, durch einen sog. Unterprogrammaufruf ersetzt. Die ausgesonderte Befehlsfolge nennt man Unterprogramm und die rufende Hauptprogramm (Abb.3.8).

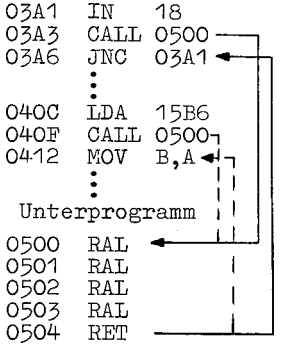


Abb.3.8: Zweimaliger Aufruf eines Unterprogrammes

An die Stelle im Hauptprogramm, wo die Befehlsfolge des Unterprogrammes abgearbeitet werden soll, wird ein CALL-Befehl geschrieben. Dieser CALL-Befehl enthält im Adreßteil die Adresse des ersten Bytes des Unterprogrammes. Wenn nun der Mikroprozessor bei der Abarbeitung des Programmes auf einen CALL-Befehl kommt, bewirkt dies einen Sprung zu der angegebenen Unterprogrammadresse. Bevor der Mikroprozessor diesen Sprung ausführt, merkt er sich jedoch noch die Adresse des nächsten Befehles hinter dem CALL-Befehl.

Jedes Unterprogramm endet mit einem RET-Befehl (return). Wenn der Mikroprozessor diesen Befehl erreicht, springt er zurück ins Hauptprogramm; und zwar zu derjenigen Adresse, die er sich beim Unterprogrammaufruf gemerkt hat, d.h. auf den unmittelbar hinter dem CALL stehenden Befehl.

Die Wirkung dieses Verfahrens ist demnach so, als würde an der Stelle des CALL-Befehles das gesamte Unterprogramm - ohne den RET-Befehl - stehen.

Der Stapelzeiger

Beim Beispiel in Abb.3.8 merkt sich der Mikroprozessor, bei der Ausführung des ersten CALL-Befehles, die Rücksprungadresse 03A6. Wo wird nun diese Adresse abgespeichert, damit der RET-Befehl auf sie zurückgreifen kann?

Eine Möglichkeit besteht darin, spezielle Register hierfür vorzusehen.

Der MP 8080/85 benutzt jedoch ein anderes Verfahren, welches eine größere Flexibilität mit sich bringt.

Im Mikroprozessor 8080/85 gibt es ein spezielles 16bit-Register, den sog. Stapelzeiger oder Stackpointer (SP, Abb.2.3). Mit Hilfe dieses Stapelzeigers kann sich der Mikroprozessor Rücksprungadressen und auch Registerinhalte merken (s.a. "Besonderheiten").

Der Stapelzeiger selbst nimmt keine Rücksprungadresse auf, denn dann könnte er ja nur eine davon speichern. Im Stapelzeiger steht vielmehr eine RAM-Adresse (Arbeitsspeicher), von der ausgehend Rücksprungadressen abgespeichert werden können. Mehrere Rücksprungadressen sind immer dann erforderlich, wenn in einem Unterprogramm erneut ein CALL-Befehl steht, der ein weiteres Unterprogramm aufruft. Eine solche Unterprogrammverschachtelung kann durchaus über viele Stufen gehen. Doch hierzu später mehr.

Der Stapelzeiger kann in vielem so behandelt werden, wie ein Registerpaar. Es ist z.B. möglich ihn mittels des LXI-Befehles mit einer 2Byte-Konstanten zu laden. Diese Konstante hat dann die Bedeutung einer Adresse.

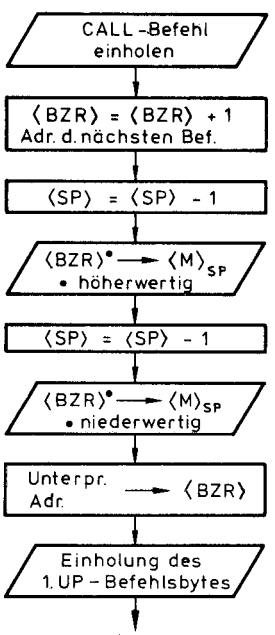


Abb.3.9: Abarbeitung eines CALL-Befehles (BZR=Befehlszählregister, SP=Stapelzeiger, M=Stapelspeicher)

Die Adresse im Stapelzeiger bezeichnet das Ende eines Arbeitsspeicherbereiches, der als Stapelspeicher (Stack) bezeichnet wird.

Da eine Rücksprungadresse immer aus 2Byte besteht, werden jeweils 2 Speicherplätze im Stapelspeicher für das Abspeichern einer Adresse benötigt. Hierzu erniedrigt der Mikroprozessor den Inhalt des Stapelzeigers um 1 und speichert das höherwertige Byte ab, dann wird der Stapelzeiger erneut um 1 erniedrigt und das niederwertige Byte unter dieser Adresse abgespeichert (Abb.3.9).

Im Stapelspeicher (-Bereich) werden also die zu speichernden Daten zu niedrigeren Speicheradressen hin 'aufgestapelt'. Aus diesem Grunde wird in den Stapelzeiger zu Beginn eines Programmes meistens eine hohe - häufig auch die höchste - Adresse des Arbeitsspeichers eingeschrieben, damit zu niedrigeren Adressen hin Platz für den Stapel besteht.

Unterprogrammverschachtelung

Von einer Unterprogrammverschachtelung spricht man, wenn in einem Unterprogramm ein weiteres Unterprogramm aufgerufen wird.

Adr.	Befehl	(SP)*	RAM Adr.	RAM Inh.	Bemerkungen
0000	LXI SP,1100	1100			SP laden
	⋮				
003C	CALL 024B	10FE	10FE	3F	Rücksprungadresse ins RAM
			10FF	00	laden und Sprung ins UP I
003F	XXXX	1100			
	⋮				
	U P I				
024B	XXXX	10FE			
	⋮		10FC	56	
0253	CALL 03A2	10FC	10FD	02	Sprung ins UP II
0256	XXXX	10FE	10FE	3F	
	⋮		10FF	00	
	RET	1100			Rücksprung ins Hauptprogramm
	U P II				
03A2	XXXX	10FC			
	⋮				
	RET	10FE			Rücksprung ins UP I

*Inhalt nach Ausführung des Befehles.

Abb.3.10: Beispiel für eine zweifache Unterprogrammverschachtelung. Höchste RAM-Adresse 10FF.

In einem solchen Fall stapelt der Mikroprozessor, beim Aufruf des zweiten Unterprogrammes, die weitere Rücksprungadresse auf die erste.

In Abb.3.10 wird sichtbar, daß der Stapelzeiger mit einer Adresse geladen werden kann, die um 1 höher ist als die größte RAM-Adresse (Arbeitsspeicher), denn bevor über den Stapelzeiger etwas abgespeichert wird, erniedrigt der Mikroprozessor den Inhalt um 1 (Abb.3.9).

Aufgrund dieser Verfahrensweise zeigt der Stapelzeiger immer auf den Speicherplatz, in den zuletzt etwas geschrieben wurde. Erst bei einem erneuten CALL-Befehl wird der Stapelzeiger dekrementiert und zeigt damit auf den nächsten freien Speicherplatz.

Bei der Ausführung eines RET-Befehles werden das Speicherbyte, auf welches der Stapelzeiger gerade zeigt und dasjenige, welches im nächsten Speicherplatz steht, zusammen als Rücksprungadresse gewertet. Bevor der Rücksprung ausgeführt wird, erhöht der Mikroprozessor den Stapelzeiger nochmals um1, so daß er nun auf die evtl. vorhandene nächste Rücksprungadresse zeigt.

Besonderheiten

Der MP 8080/85 verfügt auch über bedingte Unterprogrammaufrufe. Genau wie bei bedingten Sprüngen, werden sie nur ausgeführt, wenn bestimmte Kennzeichenbits gesetzt oder nicht gesetzt sind.

Beispiel: CC (Call on carry): Rufe das Unterprogramm auf, wenn das Übertragsbit gesetzt ist.

In gleicher Weise können auch bedingte Rücksprungbefehle benutzt werden.

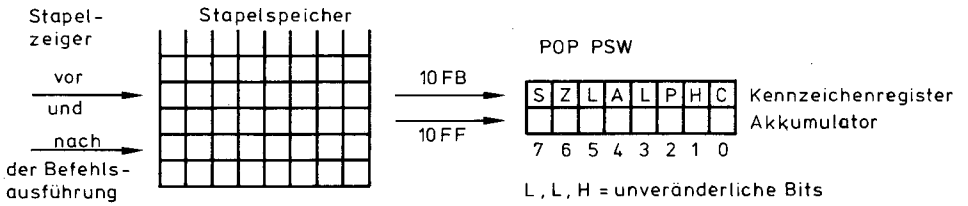
Beispiel: RNZ (Return on no zero): Springe zurück, wenn das Nullbit nicht gesetzt ist.

Wenn die Unterprogrammtechnik angewandt werden soll, gibt es zwei weitere Befehle, die von besonderer Bedeutung sind. Es sind dies der PUSH- und POP-Befehl. Diese beiden Befehle haben die Aufgabe, die Inhalte der Register und die Kennzeichenbits zu 'retten' und zurück zu holen. Mit einem Befehl können immer 16bit verlagert werden.

Beispiele: PUSH B: Kopiere den Inhalt der Register B und C in den Stapelspeicher.

PUSH D: $\langle D \rangle \dashrightarrow \langle M \rangle_{SP-1}$; $\langle E \rangle \dashrightarrow \langle M \rangle_{SP-2}$

POP PSW: Kopiere den Inhalt des Stapelspeichers ins Kennzeichenregister und in den Akku.



Die Registerinhalte werden also durch den PUSH-Befehl in den Stapelspeicher kopiert und durch den POP-Befehl wieder zurückgeladen. Hierbei wird der Stapelzeiger in gleicher Weise benutzt, wie bei einem CALL- und RET-Befehl.

Hieraus leitet sich eine wichtige Schlußfolgerung ab: In einem Unterprogramm müssen immer gleich viele POP- wie PUSH-Befehle stehen, da sonst der Stapelzeiger bei der Ausführung des RET-Befehles nicht auf die richtige Rücksprungadresse zeigt.

Die PUSH- und POP-Befehle werden in der Unterprogrammtechnik vor allem dazu benutzt, um zu Beginn des Unterprogrammes die Inhalte der zu benutzenden Register zu retten und vor dem Rücksprung ins Hauptprogramm, wieder zurück zu laden. Auf diese Weise wird sichergestellt, daß bei der Abarbeitung eines Unterprogrammes keine, für das Hauptprogramm wichtigen, Registerinhalte verändert werden.

Beispiel:

```
1. LXI  SP, 1100  ; Stapelzeiger laden
2. IN   18        ; Akku mit Port 18 laden
3. CALL UP1       ; Unterprogramm 1 aufrufen
8. MOV  B,A       ; Akkuinhalt ins Reg. B kopieren
   :
   :
   Unterprogramm UP1
4. PUSH PSW       ; Akkuinhalt und Kennzeichen retten
5. MVI  A,FF      ; Akku mit FF laden
   :
   :
6. POP  PSW       ; Akkuinhalt und Kennzeichen rückladen
7. RET            ; Zurück ins Hauptprogramm
```

Stapelspeicher	10FC	10FD	10FE	1CFF	1100
Stapelzeiger nach Befehlsausführung	↑4.		6.↑3.	7.↑1.	

Konkrete Beispiele hierzu enthält Kap. 4..

Die Übergabe von Daten und Parametern an ein Unterprogramm, oder die Übergabe von Ergebnissen ans Hauptprogramm kann über Register oder Speicherplätze erfolgen.

Im ersten Programm des Kap.4.4 (Blinkprogramm) wird z.B. der Zeitschleifenparameter durch das Register B ans Unterprogramm weitergereicht und dadurch die Blinkfrequenz festgelegt. Insbesondere, wenn größere Datenmengen ans Unterprogramm geleitet werden sollen, wird hierzu ein Speicherbereich festgelegt, auf den sowohl das Haupt- als auch das Unterprogramm zugreift.

Beispiel: Speicherbereich für den Datenaustausch HP --→ UP.
Zeilenpuffer für einen Drucker.

```
0000  LXI  SP,2000
0003  LXI  H,1F80 ; Beginn Zeilenpuffer
0006  MVI  B,50   ; Zeilenpuffer Länge = 80 Zeichen
0008  MVI  M,2A   ┐Bringe 80 mal 2A
000A  INX  H      │(ASCII-Wert für einen
000B  DCR  B      │Stern) in den
000C  JNZ  0008   ┘Zeilenpuffer
000F  CALL 0A00   ; Drucke eine Zeile voller Sterne
      :
      : UP Druckprogramm

0A00  PUSE B
0A01  PUSH H
0A02  MVI  B,50
0A04  LXI  H,1F80
      :           ┐Sende 80 Zeichen aus dem Zeilenpuffer
      :           ┘an den Drucker
0A19  POP  H
0A1A  POP  B
0A1B  RET
```

Wie aus dem Beispiel ersichtlich, müssen bei der Verwendung mehrerer PUSH-Befehle die POP-Befehle in umgekehrter Reihenfolge auftreten, da sonst die Registerinhalte vertauscht werden. Was zuletzt in den Stapelspeicher kopiert wurde, muß zuerst wieder rückgeladen werden.

Insbesondere beim Programmieren in einer Maschinensprache, werden Unterprogramme nicht nur dann benutzt, wenn Programmteile häufiger

gebraucht werden. Unterprogramme werden oft auch zur Struktuierung eines Programmes benutzt.
Hierunter wird verstanden, daß Programmteile, die eine weitgehend selbständige Funktion erfüllen, aus dem Hauptprogramm ausgegliedert werden, damit es übersichtlicher bleibt. Dieses Verfahren läßt sich soweit treiben, daß im Hauptprogramm fast nur noch CALL-Befehle und einige organisatorische Programmteile verbleiben. Die CALL-Befehle übernehmen dabei die Funktion komplexerer Kommandos und der Programmablauf wird recht übersichtlich.

```
Beispiel:   LXI  SP,2000 ;
            CALL UP1     ; Ports programmieren
            CALL UP2     ; Teste die Speicher
            IN   5A      ; Wahlschalter lesen
            RRC          ; Akku nach rechts rotieren und Bit 0 ins
                           Carry-Flag
            CC   UP3     ; Unterpr. 3, wenn Carry = H
            RRC
            CC   UP4
            RRC
            CC   UP5
             :
             :
```

In dem Beispiel wird gezeigt, wie ein Mikrocomputer dazu veranlaßt werden kann, verschiedene Programme abzuarbeiten. Je nachdem welches Bit, in dem über das Eingabe-Port 5A eingelesene Byte, ein H ist, werden ein oder mehrere Unterprogramme aufgerufen. Hierbei ist es wichtig, daß die Unterprogramme ab UP3 den Inhalt des Akkus unverändert lassen (PUSH-POP).

Interrupts

Unter einem Interrupt versteht man eine von außen angeforderte Unterbrechung eines laufenden Programmes. Ein Mikroprozessor besitzt ein oder mehrere Anschlußstifte, über die diese Unterbrechung erfolgen kann.
Wenn durch ein äußeres Signal eine solche Unterbrechung gewünscht wird, bearbeitet der Mikroprozessor den momentanen Befehl noch zuende und führt dann einen Unterprogrammsprung aus. Das Ziel dieses Sprunges kann entweder durch elektronische Maßnahmen in einem kleiner Bereich variiert werden, oder ist von vorn herein festgelegt. Am Sprungziel muß dann ein Unterprogramm stehen, welches das Gerät, das die Unterbrechung ausgelöst hat, bedient. Beim Erreichen des RET-Befehles wird ein Rücksprung zu dem Befehl ausgeführt, der bei der Unterbrechung als nächster abgearbeitet worden wäre. Da es Programmteile geben kann, die durch eine solche Unterbrechung zu sehr gestört würden - z.B. eine Zeitschleife - gibt es zwei Befehle mit denen die Annahme einer solchen Unterbrechung verhindert (DI = Disabel interrupts) und wieder erlaubt (EI = Enable interrupts) werden kann.
Eine ausführliche Behandlung der Interrupts enthält Kap.7.5

3.5 Spezielle Probleme und ihre Lösung

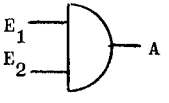
Vergleich von Hard- und Software-Lösungen

Der Mikroprozessor kann, wie jeder Computer (Rechner), zur Lösung mathematischer Probleme eingesetzt werden.

Darüberhinaus können mit einem Computer aber auch Regel-, Steuerungs- und Überwachungsprobleme gelöst werden. Die konventionellen Computer für diesen Aufgabenbereich heißen Prozeßrechner.

Bevor der Mikroprozessor verfügbar war, wurden kleinere und mittlere Aufgaben der Steuerungs- und Regeltechnik durch speziell auf einen Anwendungsfall zugeschnittene elektronische Schaltungen gelöst. Hauptbestandteile solcher Schaltungen waren sog. Gatter, die verschiedene logische Verknüpfungen zwischen mehreren elektrischen Eingangsvariablen ermöglichten.

Beispiel: UND-Verknüpfungen zweier Eingangsvariablen.



E_1	E_2	A
L	L	L
H	L	L
L	H	L
H	H	H

Das in dem Beispiel dargestellte UND-Gatter macht eine fast verzögerungsfreie Verknüpfung gemäß der rechts stehenden Tabelle. Die Signalverzögerungszeiten liegen in der Größenordnung von einigen 10^{-9}s (ns).

Die gleiche Funktion kann auch von einem Mikroprozessor übernommen werden (Abb.3.11).

```
                    Befehlsadresse (Adresse des 1.Bytes)
                        Hex-Code bzw. Objektcode
                            Mnemonic-Code bzw. Operationscode
                                Operanden
                                        Kommentare
0000 DB19       IN      19H     ;Eingang 19 (E1) lesen und in Akku bringen
0002 47         MOV     B,A     ;Daten vom Akku in Register B kopieren
0003 DB1A       IN      1AH     ;Eingang 1A (E2) lesen und in Akku bringen
0005 A0         ANA     B       ;UND Verknuepfung zwischen Reg.B und Akku
                                ;Das Ergebnis steht im Akkumulator
0006 D318       OUT     18H     ;Der Inhalt des Akkus wird auf den Ausgang
                                ;(A) gegeben
0008 C30000     JMP     0000H   ;Sprung zum Programmanfang
```

Abb. 3.11: MP-Programm zur UND-Verknüpfung zweier 8bit Eingangsvariablen.

Da bei einem Mikroprozessor zwischen dem Anliegen der Signale auf den Eingängen und der Ausgabe des Ergebnisses einige Befehle abgearbeitet werden müssen, ist die Signalverzögerungszeit entsprechend länger (siehe Übungsaufgabe 3.18). Hierdurch ist im allgemeinen die Grenze für den Einsatz von Mikroprozessoren gegeben.

Dezimalkorrektur des Akkumulators

Hierzu wird der Befehl DAA verwandt, er verwandelt die 8 Bit im Akkumulator in eine zweistellige Dezimalzahl (BCD-Code).

Dieser Befehl wird z.B. zur Addition von Dezimalzahlen verwendet. Es ist dies der einzige Befehl der durch das Hilfs-Carry-Flag beeinflußt wird.

Jeder 4-Bit-Wert (Tetrade) kann als Dezimalziffer behandelt werden, solange keine Pseudotetraden (A.....F) auftreten. Um bei einer Addition - welche im MP immer binär erfolgt - die dezimale Darstellung aufrechtzuerhalten, muß zu einer Tetrade 6 hinzu addiert werden, wenn Werte von 10 bis 15 (A.....F) auftreten.

Beispiel:

```
   2985
+  4936
-------
=  7921
```

1) Addition der Einer und Zehner (niederwertige Bytes):

```
 85 = HLLLLHLH
 36 = LLHHLHHL
-------------
⟨C⟩◄--L HLHHHLHH = BB ,  Hilfsübertrag = L
```

2) Durchführung der DAA-Operation:
Da die rechte Tetrade größer als 10 ist, wird 6 hinzu addiert.

```
⟨A⟩ = HLHHHLHH
  6 =     LHHL
        --------
        HHLLLLLH
```

Da die linke Tetrade ebenfalls größer als 10 ist wird dort auch 6 addiert.

```
⟨A⟩ = HHLLLLLH
      LHHL
    ---------
⟨C⟩◄--H LLHLLLLH = 21H
```

Der Akkuinhalt wird abgespeichert.

3) Die beiden höherwertigen Ziffern und der Inhalt des Übertragsspeichers (C) werden addiert.

```
 29 = LLHLHLLH
 49 = LHLLHLLH
⟨C⟩ =        H
    ----------
⟨C⟩◄--L LHHHLLHH = 73H ,  Hilfsübertrag = H
```

4) Durchführung der DAA-Operation:
Da das Hilfsübertragsbit gesetzt ist, wird 6 zum Akku addiert.

```
⟨A⟩ = LHHHLLHH
  6 =     LHHL
        -------
⟨C⟩◄--L LHHHHLLH = 79
```

Da die linke Tetrade kleiner als 10 ist und das Übertragsbit nicht gesetzt, ist die Operation beendet. Das Endergebnis lautet: 7921

Zur weiteren Verdeutlichung wurde in Abb.3.12 ein Programm dargestellt, das die Aufgabe aus obigem Beispiel löst.

Die beiden zu addierenden Zahlen werden in die Speicherplätze 20+21H und 22+23H eingeschrieben, während das Ergebnis dann in den Speicherplätzen mit den Adressen 24+25H erscheint.

```
                    Mne.  Operand     Flags     /     Register
                                      Z C S P A / A  B  C  D  E  H  L
0000 2A2000         LHLD  0020H     ;X X X X X   X  X  X  X  X  85 29
0003 EB             XCHG            ;X X X X X   X  X  X  85 29 X  X
0004 2A2200         LHLD  0022H     ;X X X X X   X  X  X  85 29 36 49
0007 7A             MOV   A,D       ;X X X X X   85 X  X  85 29 36 49
0008 84             ADD   H         ;L L H H L   BB X  X  85 29 36 49
0009 27             DAA             ;L H L H H   21 X  X  85 29 36 49
000A 67             MOV   H,A       ;L H L H H   21 X  X  85 29 21 49
000B 7B             MOV   A,E       ;L H L H H   29 X  X  85 29 21 49
000C 8D             ADC   L         ;L L L L H   73 X  X  85 29 21 49
000D 27             DAA             ;L L L L L   79 X  X  85 29 21 49
000E 6F             MOV   L,A       ;L L L L L   79 X  X  85 29 21 79
000F 222400         SHLD  0024H     ;L L L L L   79 X  X  85 29 21 79
0012 C30000         JMP   0000H     ;L L L L L   79 X  X  85 29 21 79
```

Abb. 3.12: Dezimaladdition der Zahlen 2985 und 4936; mit Angabe aller Flag- und Registerinhaltes. X bedeutet unbestimmter Inhalt.

Die drei ersten Befehle sind dazu da, die Registerpaare DE und HL mit den beiden Summanden zu laden und der vorletzte Befehl bringt die Summe in den Speicher. Im Kommentarteil eines jeden Befehles (hinter dem Semikolon) ist wiedergegeben, welche Inhalte die Flags und die Register nach Ausführung dieses Befehles haben.

Datentransfer bei Registerpaaren

Der Befehl XCHG bewirkt einen Austausch der Registerpaarinhalte von DE und HL. Wenn die Inhalte anderer Registerpaare ausgetauscht oder kopiert werden sollen, können hierzu die Befehle PUSH und POP benutzt werden.

Beispiele: ⟨BC⟩ --▸ ⟨DE⟩ :
```
PUSH B
POP  D
```
⟨BC⟩ ◂-▸ ⟨HL⟩ :
```
PUSH B
PUSH H
POP  B
POP  H
```

Inhalt des Stapelzeigers

Der Inhalt des Stapelzeigers ist nicht abfragbar. Wenn man in einem Programm wissen möchte, welchen Inhalt der Stapelzeiger momentan hat, kann man dazu die beiden folgenden Befehle benutzen:

```
LXI  H,0000
DAD  SP     ; Der Inhalt von SP ist in HL
```

Unterprogramme ohne Stapelzeiger

Wenn ein Mikrocomputer keinen Arbeitsspeicher (RAM) hat - z.B. in einem Minimalsystem - kann nicht mit dem Stapelzeiger gearbeitet werden.

Die Anwendung der normalen Unterprogrammtechnik ist dann nicht möglich.

Mit Hilfe des PCHL-Befehles läßt sich jedoch eine alternative Technik realisieren. Der PCHL-Befehl bringt den Inhalt des Registerpaares HL in den Befehlszähler (PC = Program counter). Dies bedeutet, daß das Programm einen Sprung zu der in HL stehenden Adresse ausführt. Darüberhinaus kann mittels dieser Technik dafür gesorgt werden, daß nach Beendigung des Unterprogramms zu einem beliebigen Befehl gesprungen wird.

Beispiel:

```
Hauptprogramm
    .
    .
013A LXI H,0140
013D JMP   0300
0140 XXX
    .
    .
02A3 LXI H,02AB
02A5 JNZ   0300
02A8 XXX
02AB XXX
```

```
Unterprogramm
0300 XXX
      .
      .
     PCHL
```

Vergleich von Registerpaar-Inhalten

Für das Arbeiten mit Registerpaaren, d.h. mit 16bit-Worten, stehen einige Befehle zur Verfügung (XCHG, LXI, INX, DAD...). Jedoch keiner dieser Befehle beeinflußt die Kennzeichenbits. Lediglich der DAD-Befehl (⟨HL⟩ = ⟨HL⟩ + ⟨rp⟩) beeinflußt ein Kennzeichen (C). Beim MP 8085 steht zur Zeit ein weiterer Befehl zur Verfügung, der alle Kennzeichenbits beeinflußt. Dieser Befehl führt eine 16bit Subtraktion aus und läßt sich z.B. zum vergleichen von Registerpaarinhalten verwenden.

Beispiel: Vergleiche BC mit DE

```
PUSH H
PUSH D
XCHG
DSUB  ⟨HL⟩ = ⟨HL⟩ - ⟨BC⟩ , alle Flags werden beeinflußt
XCHG
POP  D
POP  H
```

3.6 Übungsaufgaben

3.1 Was ist ein verzweigtes Programm, welche Verzweigungsarten kennen Sie und welche Bedeutung hat hier das Befehlszählregister?

3.2 Welche Kennzeichenbits kennen Sie und wann werden sie gesetzt?

3.3 Welche Bedeutungen können das 2. und 3. Byte eines Befehles haben?

3.4 Welchen Hex-Code hat der Befehl JMP BADEH?

3.5 Was verstehen Sie unter dem Mnemonic-Code?

3.6 Welchen Hex-Code haben folgende Befehle und was bewirken sie? DCR B, ADD M, ADI BA, ANI 00, STA 1210

3.7 Was kann die folgende Zeichenfolge bedeuten HHLLLLHHLLHHHHLLL ?

3.8 Was müssen Sie bedenken, wenn Sie den Befehl MOV B,M geben?

3.9 Welche Registerpaare hat der 8080?

3.10 In Abb.3.13 ist ein Beispiel aufgeführt für das Setzen der Kennzeichenbits (Flags). Das Programm ist nicht bis zum Ende ausgeführt worden. Den unteren Teil sollte der Leser vervollständigen.

	Bef. zähl Hex	Register A	B	C	Flags C	P	Z	S	Nebenrechnungen
MVI A, 28_D	0	1C	X	X	X	X	X	X	28:16=1R12 = 1C
MVI B, 5_D	2	1C	5	X	X	X	X	X	
MOV C,A	4	1C	5	1C	X	X	X	X	
DCR B	5	1C	4	1C	X	L	L	L	LLLHHHLL | 1C
ADD C	6	38	4	1C	L	L	L	L	+ LLLHHHLL | 1C
DCR B	7	38	3	1C	L	H	L	L	LLHHHLLL | 38
JNZ 0006	8	38	3	1C	L	H	L	L	
OUT 18	B								
	6	54	3	1C	L	L	L	L	+ LLLHHHLL | 1C
	7								LHLHLHLL | 54
	8								
	6								
	7								
	8								
	6								
	7								
	8								

Abb. 3.13: Programmbeispiel zur Flagbeeinflussung; X bedeutet unbestimmter Inhalt.

Die Programmschleife von der Befehlsadresse 6 bis A wird mehrfach durchlaufen. Im unteren Teil der Abb.3.13 können die sich daraus ergebenden Änderungen der Register und Flags eingetragen werden. Die Inhalte der Register und Flags bleiben immer solange erhalten, bis sie durch einen Befehl verändert werden.
Welche Aufgabe hat dieses Programm?

3.11 Welche Adressierungsart wird bei den folgenden Befehlen verwandt?
a) JC 3A00; b) MOV A,B; c) RRC; d) ANI 3F

3.12 Wie heißen die einzelnen Teile der folgenden Befehlszeile?

```
 1.   2.   3.  4.                5.
03A5  78  MOV A,B  ; Reg. B in Akku kopieren
```

3.13 Was steht im Stapelzeiger?

3.14
```
0000 LXI  SP,1029
     MVI   A,FF
     OUT     18
     CALL    0020
     :
0020 PUSH B
     MVI  B,FF
     DCR  B
```

a) Wie lautet an dieser Stelle des Programms der Inhalt des Stapelzeigers?

```
     :
     :
     RET
```

b) Wie lautet der Inhalt des Programmzählers nach Ausführung dieses Befehles?

3.15 Was geschieht, wenn man in einem UP den RET-Befehl durch die beiden Befehle POP H und PCHL ersetzt?

3.16 Wie tief können Unterprogramme ineinander verschachtelt werden?

3.17 Was ist ein Interrupt?

3.18 Welche Zeit vergeht zwischen dem Einlesen des 1. Befehles in Abb.3.11 und der Ausgabe des Akku-Inhaltes bei 2MHz Taktfrequenz (Signalverzögerungszeit)?

4. Programmerstellung mit einem MP-8085-Minimalsystem

Wie schon in der Einleitung ausgeführt wurde, soll dem Leser an dieser Stelle Gelegenheit geboten werden, das bisher Erlernte anzuwenden. Hierzu eignet sich in besonderer Weise ein einfacher Mikrocomputer mit Hexadezimaltastatur, wie er häufig als Lern- und Trainingsgerät eingesetzt wird.

Nur die Entwicklung eigener Programme bringt ein fundiertes Wissen über die Zusammenhänge und Abläufe in einem Mikrocomputer mit sich.

In diesem Kapitel wird der, bereits in Abb.3.3 dargestellte, Mikrocomputer MICO 85 als Übungssystem benutzt. Die grundsätzlichen Erörterungen sind jedoch auch auf andere Mikrocomputer dieser Art übertragbar. Eine Beschreibung der elektronischen Schaltung des MICO 85 ist in Kapitel 11 zu finden.

Die Programme dieses Kapitels werden teilweise in der üblichen Assemblerschreibweise wiedergegeben, d.h. zu dem Mnemonic-Code kommen im wesentlichen noch symbolische Adressen und einige Assembleranweisungen hinzu. Wem dies nicht geläufig ist, dem wird empfohlen zunächst das Kapitel 5.3: "Die Assemblersprache" durchzuarbeiten.

4.1 Beschreibung des Übungssystemes

Wie bereits im Kapitel 2 dargelegt wurde, verfügt der MP 8080/85 über einen 8bit-Datenbus -zum Austausch von Daten und Befehlsbytes- und über einen 16bit-Adreßbus, um Speicherplätze und Ein-/Ausgabe-Schnittstellen adressieren, d.h. auswählen zu können.

Mit 16bit (2 Byte Adresse) können 65536 verschiedene Speicherplätze adressiert werden. Zur Kennzeichnung der Größe eines Speicherraumes wird häufig die Einheit Kilobyte (KB) benutzt. Hierunter versteht man 1024 Speicherplätze mit je 8 Zellen (1 Byte); diese sind gerade durch 10bit adressierbar (2^{10} = 1024).

Der MP 8080/85 kann also einen Speicherraum von 64KB adressieren (64x1024 Bytes).

Die Adressen, bei den IN- und OUT-Befehlen, bestehen aus nur einem Byte -z.B. IN 23. Mit diesen 8bit können dann 256 verschiedene Ein-/Ausgabe-Schnittstellen (Ports) ausgewählt werden.

Speicherbelegung

Da der MP 8080/85 nach einem Reset immer zunächst die Adresse 0000 aussendet, wird auf diesem Adreßbereich ein Programmspeicher (ROM, Kap.8.4) angeordnet, der seinen Inhalt auch nach dem Abschalten der Versorgungsspannung nicht verliert (Abb.4.1).

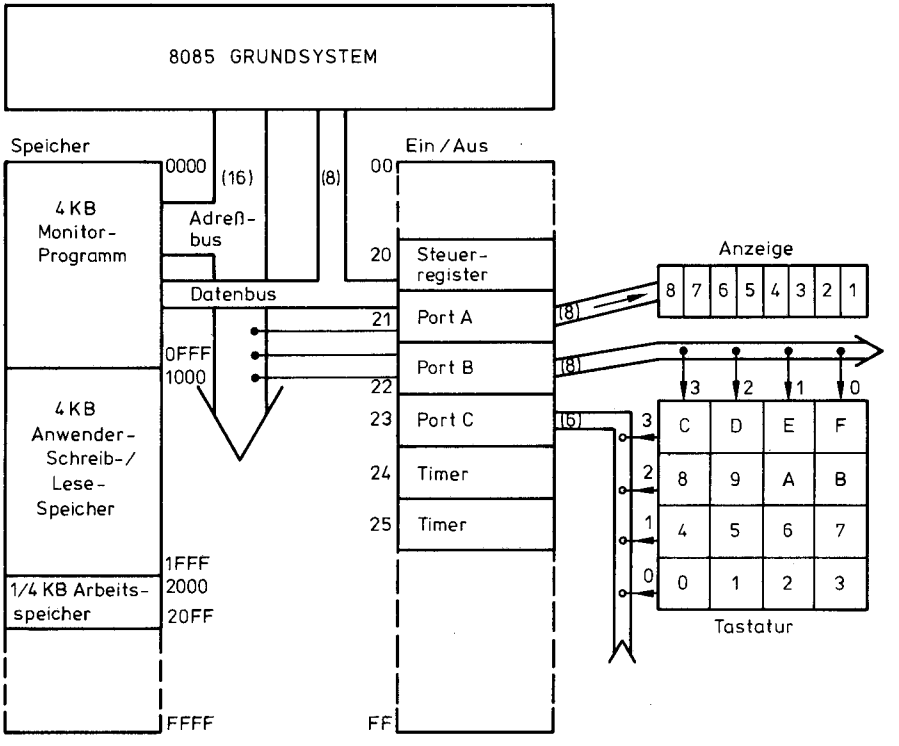


Abb.4.1: Aufbau des Übungssystemes MICO 85

Dieser Programmspeicher enthält die Grundsoftware, genannt Monitor-Programm, oder einfach Monitor (Kap.4.2).

Mit der Adresse 1000H schließt sich dann ein 4KB Schreib-/Lesespeicher (RAM, Kap.8.3) für allgemeine Anwendungen des Nutzers des Systemes an. Ein solcher Speicher wird häufig Anwenderspeicher genannt. Hier kann der Benutzer seine Programme und Daten abspeichern. Der Inhalt geht jedoch beim Abschalten verloren. Im Adreßbereich von 2000H bis 20FF schließlich, ist der Arbeitsspeicher - der "Notizblock" des Mikroprozessors - angeordnet (Beisp.4.1). Dieser Schreib-/Lesespeicher sollte vom Anwender nicht benutzt werden, da er sonst Gefahr läuft den Monitor in seiner Tätigkeit nachhaltig zu stören.

Beispiel 4.1: Arbeitsspeicheraufteilung

```
Adresse
2000-2007 : Anzeigenpuffer für die Anzeigen 1 bis 8.
2008-2028 : Vermittlung von Unterbrechungssprüngen (Interrupts
            RST1...RST7.5).
2029-203F : "Notizen" des Monitors
20XX-20FF : Stapel für CALL- und PUSH-Befehle, adressiert über den
            Stapelzeiger (stackpointer).
```

Ein-/Ausgabeschnittstellen

Es sind 3 programmierbare Schnittstellen (Port A, B, C) vorhanden. Näheres zur Programmierbarkeit und den verschiedenen Möglichkeiten, welche dieser Interface-Baustein bietet, enthalten die Kapitel 9 und 11. Doch soviel sei schon hier erwähnt: an diesen Ports sind die Anzeigeeinheiten und die Tastatur angeschlossen. Die nicht benötigten Ein-/Ausgänge und eine Reihe anderer wichtiger Leitungen stehen dem Anwender über 2 Stecker zur Verfügung. Hiermit lassen sich dann Steuerungsaufgaben und Peripherieanschlüsse realisieren. Weitere Einzelheiten über den MICO 85 enthält Kapitel 11.

4.2 Monitor und Service-Routinen

Als Beispiel für den Aufbau und die Leistungen eines Monitor-Programmes (Betriebsprogramm) und zum besseren Verständnis des folgenden, soll hier der Monitor des MICO 85 kurz erklärt werden.

Wie im vorangegangenen Kapitel bereits ausgeführt wurde, belegt der Monitor den Speicherbereich von 0000-0FFF und verfügt über einen Arbeitsspeicherraum von 2000-20FF. Wird das System eingeschaltet, oder der Reset-Taster betätigt, so beginnt der Mikroprozessor, von der Adresse 0000 an, das Monitorprogramm abzuarbeiten.

Welche Aufgaben hat solch ein Monitor? Insbesondere soll der Monitor dem Computer ein bestimmtes Mindestmaß an Fähigkeiten verleihen.

Beispiel 4.2: Aufgaben eines Monitors

- Tastatur abfragen zur Entgegennahme von Kommandos etc.,
- Kommandokennzahlen, sowie Adressen und Daten auf der Anzeige darstellen,
- Datenaustausch zwischen den internen Speichern (RAM) und den externen Massenspeichern (z.B. Magnetspeicher),
- Speicherinhalte anzeigen und verändern,
- Anwenderprogramme starten, usw..

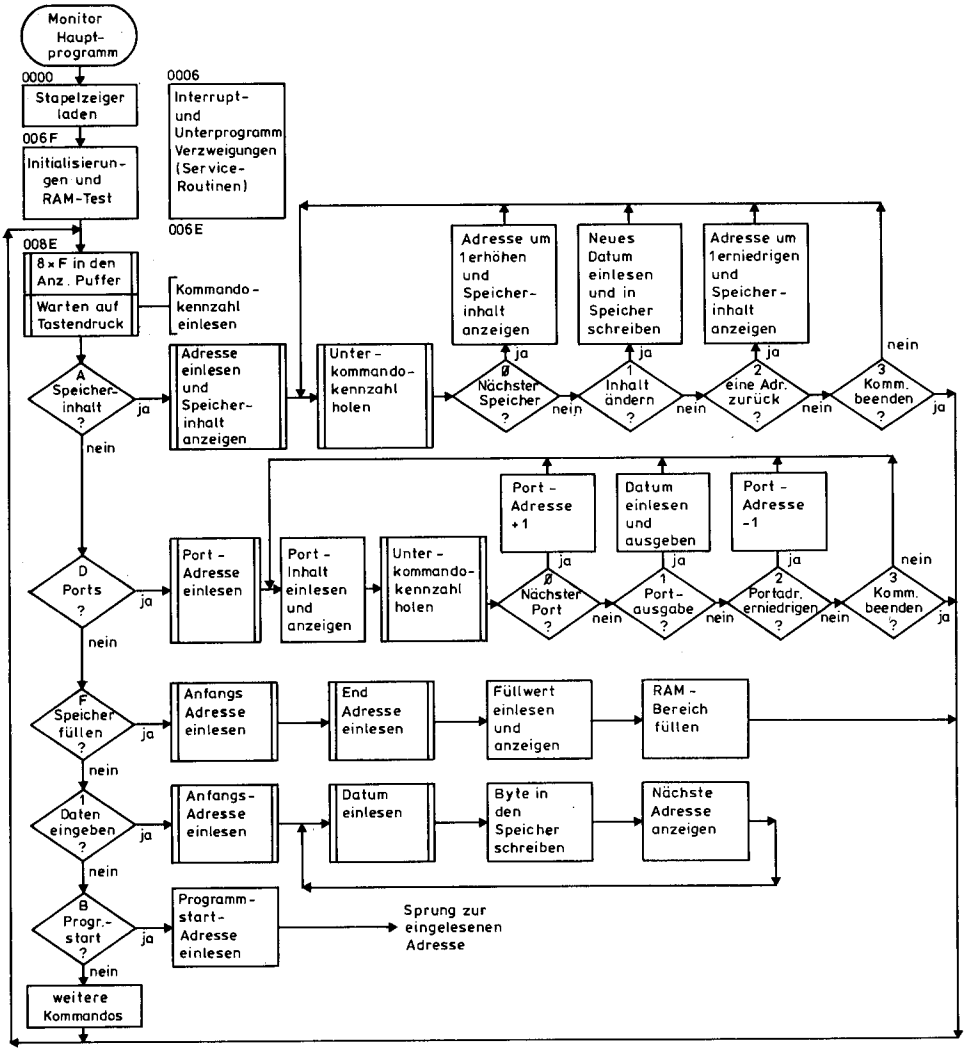


Abb.4.2: Vereinfachte Darstellung des Hauptprogramms eines Monitors

Kommandos

Die Benutzung und Aktivierung dieser Fähigkeiten geschieht durch die Eingabe von Kommandos. Hierzu fragt der Monitor -solange er zu keiner anderen Tätigkeit veranlaßt wurde- ständig die Tastatur ab und erwartet die Eingabe eines Kommandokennzeichens. Wird z.B. die Taste 1 gedrückt und danach eine 4stellige Adresse eingegeben, so bedeutet dies: die Zeichen aller danach gedrückten Tasten werden, von der eingegebenen Adresse an, in den Speicher geschrieben. Die

auf diese Weise eingeschriebenen Bytes können ein Programm darstellen, das dann durch die Eingabe eines anderen Kommandos gestartet wird (z.B. B1000: Beginn das Programm von der Adresse 1000H an abzuarbeiten).

Einige wichtige Kommandos sind in Abb.4.2 dargestellt.

Die konkrete Ausführung dieser Monitor-Funktionen, für die linke Spalte des Flußdiagrammes (Abb.4.2) und die Interrupt- und Unterprogramm-Verzweigungen, enthält Beisp.4.3.

Beispiel 4.3a: Ausschnitt aus dem Hauptprogramm des Monitors.
Bedeutung der Spalten von links nach rechts:
Adresse Objektcode (HEX-Code) Marke: Mnemonic-Code
Operanden ;Kommentar

```
;******  Monitorprogramm für den MICO 85  ******
0000 310021  LXI  SP,STACK ;Stackpointer laden
0003 C36F00  JMP  INITI    ;Interr.- und UP-Verzweigung überspringen
;Interruptvektoren auf den Arbeitsspeicher richten und
;Einsprünge in die Service-Routinen (SR) festlegen.
0006         DS   2        ;Adresse um 2 erhöhen
0008 C30820  JMP  INT      ;Sprung in den Arbeitsspeicher bei RST1
000B C34103  JMP  ZA1MS    ;Service-Routine: Zeitschleife (A)*1ms
000E         DS   2
0010 C30B20  JMP  INT+3    ;Sprung beim Interrupt RST2.
0013 C38503  JMP  TASDR    ;SR: Warten auf Tastendruck.
0016         DS   2        ;Der Tastenwert wird in A übergeben.
0018 C30E20  JMP  INT+6    ;RST3
001B C39F03  JMP  TASAB    ;SR: Tastaturabfrage, Tastenwert in A.
001E         DS   2        ;(A) = FF -→ keine Taste gedrückt.
0020 C31120  JMP  INT+9    ;RST4
0023 00      NOP
0024 C31420  JMP  INT+12   ;TRAP,nicht maskierbarer Interr.des 8085
0027 00      NOP
0028 C31720  JMP  INT+15   ;RST5
002B 00      NOP
002C C31A20  JMP  INT+18   ;RST5.5 des 8085
002F 00      NOP
0030 C31D20  JMP  INT+21   ;RST6
0033 00      NOP
0034 C32020  JMP  INT+24   ;RST6.5 des 8085
0037 00      NOP
0038 C32320  JMP  INT+27   ;RST7
002B 00      NOP
003C C32620  JMP  INT+30   ;RST7.5 des 8085
;Weitere Service-Routinen (Einsprünge in UPs)
003F C35702  JMP ANZ12 ;(B) -→ Anzeigestelle 1.+2.
0042 C35E02  JMP ANZ34 ;(B) -→       "       3.+4.
0045 C36502  JMP ANZ56 ;(B) -→       "       5.+6.
0048 C36C02  JMP ANZ78 ;(B) -→ Anzeigestelle 7.+8.
004B C38802  JMP ADRAN ;(DE) -→ Anzeigestelle 3.-6.
004E C31B03  JMP NULL  ;Anzeigepuffer FFFFFFFF setzen
0051 C3D903  JMP ANZEI ;Anzeigepuffer zur Anzeige bringen
0054 C3B502  JMP DALES ;1Byte von Tastatur holen (B) und anzeigen
0057 C3D002  JMP ADLE  ;Adresse von Tasta.einlesen (DE) und anzeigen
005A C35003  JMP KOPIE ;Copiere Speicherbereich (HL)-(BC) -→ (DE)
005D C3FE02  JMP ANLES ;Anzeige lesen:(B)=Nr.d.Anz.,-→ (B)=1Anz.Byte
0060 C3F600  JMP ANFA  ;Warmstart-Adresse, FFFFFFFF -→ Anzeige
0063 C38E00  JMP ANFA1 ;Einsprung in den Kommandobereich
0066         DS  9     ;Reserviert
```

Beispiel 4.3b: Initialisierungen und Kommandoverzweigungen.

```
006F 3E03     INITI: MVI   A,3      ;Steuerwort laden (PA+B=Aus,PC=Ein)
0071 D320            OUT   SR       ;Portbaustein 8155 programmieren
0073 210A20          LXI   H,INT    ;Arsp.Adr. zur Interruptverzweigung
0076 3E21            MVI   A,33     ; Den Speicherbereich, auf den die
0078 36C9     BERTA: MVI   M,0C9H   ; Interrupt-Vektoren gerichtet
007A 23              INX   H        ; wurden, mit C9 (RET) laden.
007B 3D              DCR   A
007C C27800          JNZ   BERTA

;Welcher RAM-Bereich steht zur Verfügung?
007F 7F              MOV   A,A      ;Flag-Befehl (Kap.4.3)
0080 210007          LXI   H,700H   ;Anfangsadresse RAM-Test
0083 222A20          SHLD  ARSP+1   ;Teststartadresse sichern
0086 CDEB01          CALL  RAMTS    ;RAM-Test (Kap.4.5)
0089 3E00            MVI   A,0      ;  Kennung "kein Einsprung vom
008B 323B20          STA   ARSP+18  ;  Testprogramm".

;Kommandoannahme und Verzweigung
008E CD1B03   ANFA1: CALL  NULL     ;Anzeigepuffer FFFFFFFF setzen
0091 CD8503          CALL  TASDR    ;Warte auf Tastendruck --> A
0094 F5              PUSH  PSW      ;Hauptkommando-Kennzahl sichern
0095 07              RLC            ;  Kennzahl in höherwertige
0096 07              RLC            ;  Tetrade bringen.
0097 07              RLC
0098 07              RLC
0099 F60F            ORI   0FH      ;Unterkomm.Kennzahl = F setzen.
009B 47              MOV   B,A      ;   Hauptkommando-Kennzahl
009C CD6C02          CALL  ANZ78    ;   auf die Anzeige 8 bringen.
009F 3E40            MVI   A,40H    ; SOD auf L setzen und damit die
00A1 30              DB    30H      ; Dezimalpunkte (Anzeige) Ein..
00A2 F1              POP   PSW      ;Hauptkomm.Kennz. zurück holen.
00A3 FE0A            CPI   0AH      ;--> Speicher-Inhalt anzeigen/
00A5 CA4E01          JZ    RAMAN    ;   ändern.
00A8 FE0D            CPI   0DH      ;--> Direktes Lesen und Ausgabe
00AA CA8601          JZ    PORTA    ;   auf Ports.
00AD FE0F            CPI   0FH      ;--> RAM-Bereich füllen.
00AF CA3201          JZ    RAMSE
00B2 FE01            CPI   1        ;--> Einschreiben von Programmen
00B4 CADB01          JZ    EIN      ;   oder Daten.
00B7 FE0B            CPI   0BH      ;--> Starten eines Programmes.
00B9 CA2901          JZ    START
00BC FE00            CPI   0        ;--> Weiteren RAM-Bereich suchen.
00BE CCEB01          CZ    RAMTS
00C1 FE0C            CPI   0CH      ;--> RAM-Inhalt copieren.
00C3 CA1701          JZ    RAMVE
00C6 FE0E            CPI   0EH      ;--> Einzelschritt Testsystem.
00C8 CAFA03          JZ    SINGL
00CB FE02            CPI   2        ;--> Programmstart auf 4000H.
00CD CA0040          JZ    4000H
00D0 FE03            CPI   3        ;--> Programmstart auf 1000H.
00D2 CA0010          JZ    1000H
00D5 FE04            CPI   4        ;--> Audiokassette lesen oder
00D7 CA0008          JZ    800H     ;   beschreiben.
00DA FE05            CPI   5        ;--> Relativierbares Programm
00DC CA0308          JZ    803H     ;   verschieben.
00DF FE06            CPI   6        ;--> EPROMs (2716/32) lesen
00E1 CA0608          JZ    806H     ;   oder programmieren.
00E4 FE09            CPI   9        ;--> Speicherinhalte vergleichen.
00E60CA0908          JZ    809H
00E9 FE07            CPI   7        ;--> Programmstart auf 1800H
00EB CA0018          JZ    1800H
00EE FE08            CPI   8        ;--> Speicherinhalt ausdrucken,
00F0 CA0310          JZ    1003H    ;   anzeigen, disassemblieren.
00F3 C38E00          JMP   ANFA1
```

Die linke Spalte in Abb.4.2 zeigt die Verzweigungen durch die Hauptkommando-Kennzahlen. Innerhalb eines Kommandos sind dann noch Unterkommandos möglich; diese Verzweigung ist durch die waagerechten Rauten dargestellt.

Auf der achtstelligen Anzeigeeinheit wird die Hauptkommando-Kennzahl ganz links und die Unterkommando-Kennzahl rechts daneben angezeigt.

Beispiel 4.4: Kommando A: Ändern eines Speicherinhaltes von 4E in C3

	Eingabe Tastatur	Anzeige 8	7	6	5	4	3	2	1
Kommando :	A	A	F	F	F	F	F	F	F
Adresse :	1000	A	F	1	0	0	0	4	E
Unterkommando :	1	A	1	1	0	0	0	4	E
Datum :	C3	A	1	1	0	0	0	C	3
Weiter :	0	A	0	1	0	0	1	X	X
Zurück :	2	A	2	1	0	0	0	C	3

Die Bedeutung weiterer Hauptkommandos enthält Beisp.4.3b.

Hauptprogramm

Das Monitor-Programm beginnt mit der Ladeoperation für den Stapelzeiger und einem Sprung zur Initialisierung (Beisp.4.3b). Der Speicherbereich von 0006 bis 003E (Beisp.4.3a) enthält, neben 3 Sprüngen zu Service-Routinen, die später erklärt werden, Sprünge in den Arbeitsspeicher.

Diese Sprungbefehle stehen auf den Adressen, die bei einer von außen -durch ein Hardware-Signal- erzwungenen Programmunterbrechung (Interrupt) von dem Mikroprozessor angesprungen werden; mehr dazu enthält Kapitel 7.4 und 11. An diesen Bereich schließen sich weitere Sprünge zu Unterprogrammen an (003F-006E). Diese häufig benötigten Unterprogramme stehen damit auch dem Benutzer des Mikrocomputers zur Verfügung und werden deshalb Service-Routinen (Dienstprogramme) genannt.

Die Anordnung der Sprünge am Anfang des Adreßbereiches sorgt dafür, daß sie immer auf der selben Adresse stehen bleiben. Bei Änderungen am Programm können sich zwar die Adressen der Unterprogramme ändern (verschieben), die auf diese Programme gerichteten Sprünge bleiben jedoch unverändert, denn die Befehlsfolge in Beisp.4.3a bleibt immer bestehen und wird höchstens nach unten hin (ab Adresse 006F) durch weitere Unterprogramm-Sprünge ergänzt, wenn neue UPs hinzu kommen.

Dem Anwender des Mikrocomputers kann deshalb eine Liste der Unterprogramm-Einsprünge zur Verfügung gestellt werden; was seine Programmierarbeit wesentlich erleichtert (Beisp.4.5).

Beispiel 4.5: Service-Routinen = Dienstprogramme

a) UP Warten auf Tastendruck: TASDR

Aufrufparameter	: keine
Veränderte Register	: Akkumulator (A)
Rückgabe	: Tastenwert in A (Hex).
UP-Adresse	: 0013H (siehe Beispiel 4.3a).

b) UP 1 Byte auf den Stellen 1.+ 2. anzeigen: ANZ12

Aufrufparameter	: Anzeige-Byte in Reg.B
Veränderte Register	: keine
Rückgabe	: keine
UP-Adresse	: 003F (laut Beispiel 4.3a)

c) UP Anzeige lesen : ANLES

Aufrufparameter	: niedrigste Nummer der beiden zu lesenden Anzeigestellen in Reg.B, Nr.→(B).
Veränderte Register	: B
Rückgabe	: Inhalt der Anzeigestellen Nr.+1 und Nr. im Register B, (Nr.+1) →(BH), (Nr.) →(BL). (BH) = Inhalt der höherwertigen Tetrade von B
UP-Adresse	: 005DH (gemäß Beispiel 4.3a)

Das Beispiel 4.6 zeigt die Benutzung zweier Service-Routinen aus Beisp. 4.5.

Beispiel 4.6: Programmieren mit Service-Routinen

Aufgabe: Der Wert jeder gedrückten Taste soll auf der Anzeigeeinheit Nr.1 erscheinen.

Lösung:

```
1000  CD1300  CALL  TASDR  ; Warten auf Tastendruck
1003  47      MOV   B,A    ; Anzeigeregister laden
1004  CD3F00  CALL  ANZ12  ; Tastenwert zur Anzeige 1
1007  C30010  JMP   1000   ; Beginn von vorn
```

Aus Beisp.4.6 geht hervor, wie sehr eine solche Programmieraufgabe, durch die Verwendung einiger Monitor-Unterprogramme, vereinfacht wird. Könnten diese UPs nicht verwendet werden, so würde das Programm um 116 Bytes länger und der Programmierer müßte detailierte Kenntnisse über den elektronischen Aufbau (hardware) des Computers haben.

Bei der Ausführung der ersten Service-Routine (TASDR) geschieht folgendes: Der MP macht zunächst -nachdem er sich die Rücksprungadresse gemerkt hat- einen Sprung zur Adresse 0013H des Monitors. Auf dieser Monitoradresse steht jedoch erneut ein Sprungbefehl und dieser führt dann zum ersten Befehl des Unterprogrammes auf der Adresse 0385H, laut Beisp.4.3a. Am Ende des Unterprogrammes, wenn eine Taste gedrückt wurde, macht der MP einen Rücksprung (RET) auf die Adresse 1003H, gemäß Beispiel 4.6.

Übungsaufgabe 4.1

a) Schreiben Sie ein Unterprogramm, das einem Hexadezimalzeichen (0...9.A...F) den entsprechenden ASCII-Code (Abb.1.8) zuordnet.
UP-Definition:
Aufrufparameter: Hex-Zeichen in niederwertiger Tetrade des Akkus,
Veränderte Register: A,
Rückgabe: ASCII-Code im Akku.

b) Schreiben Sie ein Hauptprogramm, -für den MICO 85- das ein Hex-Zeichen von der Tastatur annimmt, auf der Anzeige 1 darstellt und dann, unter Verwendung des obigen Unterprogrammes, den zugehörigen ASCII-Code ermittelt und als zweistellige Hexadezimalzahl auf den Anzeigen 7 und 8 erscheinen läßt (s.a. Beisp.4.5).

Lösung:
Aus Abb.1.8 geht hervor, daß der ASCII-Code für die Zahlen 0...9 30...39H und für die Buchstaben A...F 41...46H lautet. Hieraus folgt, daß man zu den Zahlen 30H und zu den Buchstaben 37H addieren muß, um den entsprechenden ASCII-Code zu erhalten.

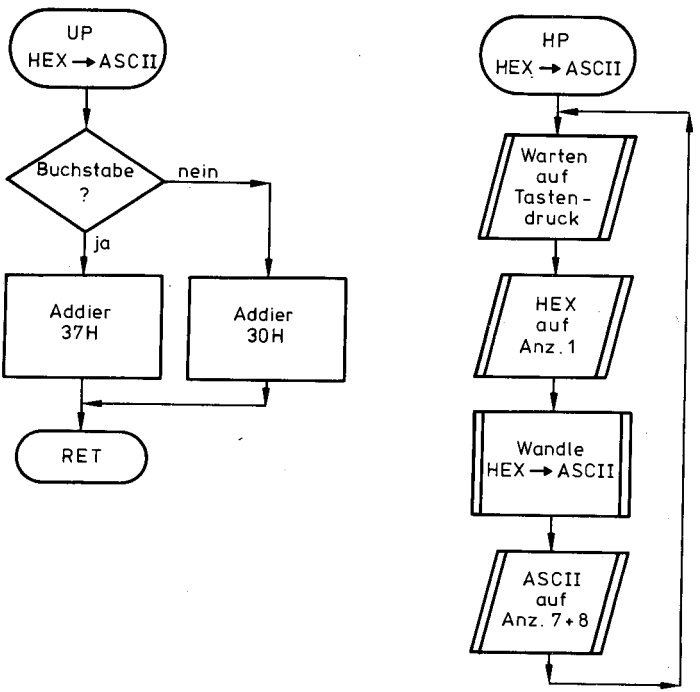


Abb.4.3: Flußdiagramm zur Lösung der Aufg.4.1.

Das Beispiel 4.7 zeigt eine Möglichkeit zum Aufbau des Unterprogrammes. Durch den ersten Befehl wird sichergestellt, daß die höherwertige Tetrade des Akkumulators immer Null ist, denn die Hex-Zahl soll ja in der niederwertigen Tetrade übergeben werden.

Beispiel 4.7: Lösung der Aufg.4.1a

```
1011  E60F     ANI  0FH    ; Höherwertige Tetrade von A=0
1013  FE0A     CPI  0AH    ; Buchstabe?
1015  DA1B10   JC   101B   ; Sprung, wenn Zahl
1018  C637     ADI  37H    ; ASCII-Code eines Buchstaben
101A  C9       RET
101B  C630     ADI  30H    ; ASCII-Code einer Zahl
101D  C9       RET
```

Der zweite Befehl (CPI 0AH) dient zur Feststellung, ob das Hex-Zeichen eine Zahl oder ein Buchstabe ist.
Jeder Vergleichsbefehl (compare) wird vom Mikroprozessor wie eine Subtraktion behandelt, nur mit dem Unterschied, daß der Akku-Inhalt erhalten bleibt.
Hieraus folgt, daß bei dem Befehl CPI 0AH (Vergleich den Akku-Inhalt mit der Zahl 0A) 0A vom Inhalt des Akkumulators abgezogen wird. Ist der Inhalt des Akkus kleiner als 0A (z.B. 05-0A), wird das Carry-Flag gesetzt, der Sprung ausgeführt und die Zahl 30H addiert (z.B. 05+30=35H --→ ASCII-Code der Zahl 5). Wenn die Zahl im Akku größer ist, wird das Carry-Flag zurück gesetzt, kein Sprung ausgeführt und die Zahl 37H addiert (z.B. 0B+37=42H --→ ASCII-Code des Buchstaben B). Im Beispiel 4.8 werden noch weitere Möglichkeiten zur Lösung dieses Problemes aufgezeigt.

Beispiel 4.8: Weitere Lösungen zur Aufgabe 4.1a

a)
```
HEXAS:  ANI  0FH
        ADI  30H
        CPI  3AH
        RC
        ADI  7
        RET
```

b)
```
HEXAS:  ANI  0FH
        ADI  90H
        DAA
        ACI  40H
        DAA
        RET
```

c)
```
HEXAS:  ANI  0FH
        CPI  0AH
        JM   ZAHL
        ADI  7
ZAHL:   ADI  30H
        RET
```

d) eine universelle Lösung zeigt Beispiel 4.12

Das Hauptprogramm (Aufg.4.1b) kann gem.Abb.4.3b leicht realisiert werden.

Beispiel 4.9: Hauptprogramm in Assembler-Schreibweise

```
Adresse  Hex-Code    Marke   Mnemonic-Code  Operanden   Kommentare
 0000                         ORG            1000H     ;Startadresse festlegen
 1000    CD1300      ANFA:    CALL           TASDR     ;Warten auf Tastendruck
 1003    47                   MOV            B,A       ;Anzeigeregister laden
 1004    CD3F00               CALL           ANZ12     ;Hex-Zeichen-→Anzeige 1
 1007    CD1110               CALL           HEXAS     ;HEX -→ ASCII wandeln
 100A    47                   MOV            B,A       ;ASCII-→Anzeigeregister
 100B    CD4800               CALL           ANZ78     ;ASCII-→ Anzeige 7 + 8
 100E    C30010               JMP            ANFA      ;Neues Zeichen einlesen
```

```
Adresse  Hex-Code      Namen   Mnemonic-Code  Operanden   Kommentare

                       ;Unbekannte Adressen festlegen
  0013                 TASDR   EQU            13H    ;UP: Tastendruck abwarten
  003F                 ANZ12   EQU            3FH    ;UP: (B) --> Anzeige 1+2
  0048                 ANZ78   EQU            48H    ;UP: (B) --> Anzeige 7+8
  1011                 HEXAS   EQU          1011H    ;UP: HEX --> ASCII
  0000                         END
```

4.3 Verschiebbarkeit und Test von Programmen

Ein Programm, das in den Objektcode (Hex-Code) übersetzt wurde, kann im allgemeinen von einem Computer nur sinnvoll abgearbeitet werden, wenn es in dem Speicherbereich steht, für den es geschrieben, bzw. übersetzt wurde.
Dies wird z.B. an dem Programm in Beispiel 4.9 deutlich: Nach der Übersetzung (Umwandlung des mnemonischen Codes und der Operanden in den Hex-Code) steht am Ende des Programmes C30010, entsprechend JMP 1000. Würde der Hex-Code dieses Programmes z.B. in einen Speicherbereich, beginnend mit der Adresse 1100H, geschrieben und dann gestartet, so könnte das Programm nur einmal sinnvoll durchlaufen werden, denn am Ende erfolgt ein Sprung auf die Adresse 1000, wo z.B. die vom Einschalten herrührenden Zufallszahlen im Speicher stehen.

Verschiebbare Programme

Durch kleine Ergänzungen an den Programmen ist der geschilderte Mangel behebbar: die Programme lassen sich durch ein spezielles Hilfsprogramm (Monitor-Kommando 5 s.u.) verschieben.
Welche Ergänzungen sind erforderlich? Zur Beantwortung dieser Frage muß man sich klar machen, welche Befehle eines Programmes geändert werden müssen und welche nicht, wenn das Programm in einem anderen Adressbereich ablauffähig sein soll. Weiterhin muß es eine Möglichkeit geben, dem Verschiebeprogramm kenntlich zu machen, welche Adressen umzurechnen sind und welche nicht. Zunächst einmal ist festzustellen, daß im wesentlichen für eine Umrechnung nur die Operanden von 3-Byte-Befehlen in Frage kommen, denn dies sind Adressen oder sie werden als solche benutzt.

Beispiel 4.10: Bedeutung der Operanden in drei-Byte-Befehlen

```
a) JMP    1000     ;Diese Operanden sind
   CALL   1011     ;immer Adressen.

b) LXI    H,10B5   ;Der erste Operand ist ein Registerpaar und der
                   ;zweite Operand kann die Bedeutung einer Adresse
          ;haben, wenn z.B. danach der Befehl MOV A,M steht ((M),
          ;adressiert durch (HL) in den Akku schreiben).
```

```
c) LXI   B,001E  ;Hier hat der zweite Operand primär die Bedeutung
   DAD   B       ;einer Konstanten, welche im folgenden Befehl zum
                 ;Inhalt des Registerpaares HL addiert wird.
```

Das Verschiebeprogramm muß also vor allem die Operanden der 3-Byte-Befehle verändern (2. und 3. Byte). Für den Fall, daß ein 3-Byte-Befehl nicht verändert werden darf (Beispiel 4.10c), wird vor diesem Befehl zusätzlich ein sog. Kennzeichenbyte in das Programm eingefügt. Welche Bytes stehen uns für solche Aufgaben zur Verfügung? Es müssen dies Bytes sein, die den normalen Ablauf des Programmes nicht stören, d.h. sie dürfen keinen, vom Mikroprozessor auszuführenden, Befehl darstellen.

Der Befehlssatz des 8080/85 enhält eine Reihe solcher Bytes und zwar sind sie den ansich sinnlosen Befehlen MOV r,r zugeordnet (z.B. MOV A,A usw.). Wenn der Mikroprozessor ein solches Byte als Befehlsbyte übermittelt bekommt, wird es wie der Befehl NOP behandelt, d.h. es geschieht nichts. Diese Bytes können zur Kennzeichnung bestimmter Befehle oder Programmbereiche herangezogen werden.

Beispiel 4.11: Kennzeichenbytes (Flag-Befehle)

```
HEX MNEMONIK
7F  MOV A,A : Der Operand des folgenden 3-Byte-Befehles darf nicht
              verändert werden.
49  MOV C,C : Es folgt eine Konstanten-Tabelle, deren Werte nicht
7F  MOV A,A   verändert werden sollen.
49  MOV C,C : Es folgt eine Adreßtabelle, alle Doppelbytes werden
40  MOV B,B   verändert.
5B  MOV E,E : Das Programm ist zuende, es folgt die Startadresse.
```

Einer Tabelle im Programm gehen zwei Kennzeichenbytes voran: Das erste kennzeichnet die Tabelle (49 MOV C,C) und das zweite gibt die Art der Tabelle an (Beispiel 4.11). Nach diesen beiden Bytes folgt noch ein Doppelbyte, welches angibt, aus wieviel Bytes die Tabelle besteht.

Das Programmende wird durch 5B (MOV E,E), gekennzeichnet. Zur eindeutigen Festlegung, von welcher Adresse an das Programm in dieser Form ablauffähig ist, wird hinter dem Endzeichen (MOV E,E) noch die Startadresse angegeben.

Beispiel 4.12: Verschiebbares Programm (als weitere Lösung zur Aufg. 4.1; s.a. Beispiel 4.9).

```
1 ;Codeumwandlung HEX --> ASCII
2 1000  CD1300  CALL  13      ;Tastatur --> (A)
3 1003  47      MOV   B,A     ;Anzeigeregister laden
4 1004  CD3F00  CALL  3F      ;HEX --> Anzeige 1
5 1007  CD1110  CALL  1011    ;HEX --> ASCII
6 100A  47      MOV   B,A
7 100B  CD4800  CALL  48      ;ASCII --> Anzeige 7+8
```

```
 8 100E  C30010  JMP   1000     ;Wiederholung
 9 ;UP Codeumwandlung mit Tabelle
10 1011  211E10  LXI   H,101E   ;Tabellenanfang
11 1014  5F      MOV   E,A      ;Hex-Zahl -> E
12 1015  1600    MVI   D,0
13 1017  19      DAD   D        ;Aufangsadresse Tabelle + Hex-Zahl
14 1018  7E      MOV   A,M      ;ASCII-Code -> (A)
15 1019  C9      RET
16 ;Umwandlungstabelle
17 101A  49      MOV   C,C      ;Kennzeichen "Tabelle"
18 101B  7F      MOV   A,A      ;Kennzeichen "Konstanten"
19 101C  1000    DW    0010     ;Anzahl der Bytes (16)
20 101E  3031323334353637       ;  Tabelle der ASCII-
21 1026  3839414243444546       ;  Zeichen 0...9 u.A...F
22 102E  5B      MOV   E,E      ;Programmende
23 102F  0010    DW    1000     ;Startadresse 1000H.
```

Mit Hilfe des Monitor-Kommandos 5 kann das Programm in Beispiel 4.12 zu einer anderen Adresse hin verschoben (kopiert) werden (Beisp. 4.13 u.4.14).
Nach der Ausführung dieses Kommandos (z.B. Verschieb zur Adresse 1100H), steht das Programm im neuen Adreßbereich (1100-1130H); die Befehle CALL 1011 und JMP 1000 wurden geändert (CALL 1111, JMP 1100), ein neuer Tabellenanfang definiert (LXI H,111E) und die Startadresse (letztes Doppelbyte = 0011) geändert.

Beispiel 4.13: Verschiebe-Kommandos

5: Programm oder Teile verschieben und Adressen umrechnen.
Unterkommandos:

1: Programm verschieben
2: Erzeug eine Lücke im Programm (s.u.)

(5)(1)(alte Anfangsadr.)(neue Anfangsadr.)
(5)(2)(Anfang Lücke)(Ende der Lücke)

(5) bedeutet, drück die Taste 5
(Adr.) bedeutet, geb eine 4stellige hexadezimale Adresse ein

Adressen, die außerhalb des Programmbereiches (1000-102D) liegen - wie z.B. die Aufrufe der Monitor-Service-Routinen - werden nicht geändert, d.h. es erübrigt sich, vor diese Befehle das Kennzeichenbyte 7F (MOV A,A) zu setzen. Über das Kommando (B)(1)(1)(0)(0) kann das Programm gestartet werden und ist ablauffähig.
Wie das Beisp.4.13 zeigt, bietet das Kommando 5 eine weitere Möglichkeit der Programmmanipulation; das Erzeugen einer Programmlücke. Dieses Hilfsprogramm dient der Lösung eines Problemes, welches beim Programmieren im Hex-Code auftritt, d.h. wenn kein Assembler (Kap. 5.4), mit symbolischer Adreßverarbeitung, zur Verfügung steht.
Wenn das Hauptprogramm aus Beisp.4.12 z.B. erweitert werden soll, verschiebt sich die Anfangsadresse des Unterprogramms und der Tabelle; diese Adressen müssen deshalb vom Programmierer neu ermittelt und geändert werden. Bei langen Programmen kann das sehr mühsam sein und birgt außerdem Fehlerquellen in sich.

Das Kommando 52 erzeugt deshalb in einem Programm, an beliebiger Stelle, eine Lücke, füllt diese mit den Bytes 00 (NOP-Befehl) und verändert alle, durch diese Programmerweiterung betroffenen, Adressen.

Beispiel 4.14: Ausschnitt aus dem Programm in Beisp.4.12, nach der Ausführung des Kommandos 52 100E 1010 (Lücke erzeugen).

```
 5 1007  CD1410  CALL  1014     ;HEX-Code in ASCXII-Code wandeln
 6 100A  47      MOV   B,A
 7 100B  CD4800  CALL  48       ;ASCII-Code auf Anzeige 7+8
 8 100E  00      NOP            ;Anfang Lücke
 9 100F  00      NOP
10 1010  00      NOP            ;Ende Lücke
11 1011  C30010  JMP   1000
12 1014  212110  LXI   H,1021   ;Tabellenanfang
```

In die erzeugte Lücke (Beisp.4.14) können dann neue Befehle geschrieben werden.
Auch die allgemeinen Teile des MICO 85-Monitors wurden verschiebbar programmiert. Hierdurch ist es möglich, diese Programmteile, unter Verwendung des Verschiebe-Kommandos, an einer anderen Stelle des Speichers (RAM-Bereich, Anwenderspeicher) neu zusammenzustellen und zu ergänzen; z.B. mit dem Ziel, ein anderes Betriebsprogramm aufzubauen.

Rückübersetzer

Ein Rückübersetzer (oder Disassembler, Kap.5.4) hat die Aufgabe, aus einem Objektcode-Programm (lauffähiger Hex-Code im Speicher) eine Liste im Mnemonic-Code zu erstellen.

Beispiel 4.15: Rückübersetzung des Programms aus Beisp.4.12 und 4.14.

a) Das Programm besteht aus folgendem Speicherinhalt:

```
1000  CD 13 00 47 CD 3F 00 CD 14 10 47 CD 48 00 00 00
1010  00 C3 00 10 21 21 10 5F 16 00 19 46 C9 49 7F 10
1020  00 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45
1030  46 5B 00 10
```

b) Hieraus erzeugt der Rückübersetzer die folgende Liste:

```
1000  CD1300   CALL  0013        --> 1018  1600     MVI   D,00
1003  47       MOV   B,A             101A  19       DAD
1004  CD3F00   CALL  003F            101B  46       MOV   A,M
1007  CD1410   CALL  1014            101C  C9       RET
100A  47       MOV   B,A             101D  49       MOV   C,C
100B  CD4800   CALL  0048            101F  7F       MOV   A,A
100E  00       NOP                   101F  1000     DW    0010
100F  00       NOP                   1021  3031323334353637
1010  00       NOP                   1029  3839414243444546
1011  C30010   JMP   1000            1031  5B       MOV   E,E
1014  212110   LXI   H,1021          1032  0010     DW    1000
1017  5F       MOV   E,A       -->|
```

Wenn an den hier beschriebenen Mikrocomputer (MICO 85) ein Drucker angeschlossen wird (Kap.11), kann ein Speicherinhalt wahlweise in den beiden Formen des Beisp.4.15 dargestellt werden (Kommando 81 oder 82).

Eine Liste gemäß Beisp.1.14b ist insbesondere dann von großem Vorteil, wenn Lücken erzeugt und neue Befehle in das Programm eingefügt worden sind. Enthält ein Programm eine Tabelle, etwa wie im Beisp.4.15, so erkennt der Rückübersetzer dies an dem Kennzeichenbyte 49 (MOV C,C).

Wurde das Programm dagegen ohne Kennzeichenbytes geschrieben, so werden die Bytes der Tabelle als Befehlsbytes aufgefaßt und in den entsprechenden Mnemonic-Code übersetzt. Angewandt auf das Beisp.4.15 bedeutet dies z.B., daß die Tabellenbytes 30...39 und 41...46 als die folgenden Befehle interpretiert und ausgedruckt würden: SIM; LXI SP,3332; INR M; DCR M; MVI M,37; SDSI 39; MOV B,B; MOV B,C; MOV B,D; MOV B,E;...

Programmtest

Jeder Programmierer macht die Erfahrung, daß ein neues Programm in den meisten Fällen - wenn es sich nicht um ein Trivialprogramm handelt - fehlerbehaftet ist.

Die häufigsten Fehlerarten sind:

- Logische Fehler beim Entwurf des Programmes,
- Schreibfehler bei der Programmeingabe,
- Unbeabsichtigte Veränderungen von Register- und Speicherinhalten,
- Schwierigkeiten mit der Hardware.

Wie können solche Fehler aufgespürt werden?
Die primäre Möglichkeit, nämlich eine Kontrolle der Programmliste, führt nicht immer zum Ziele, ist sehr aufwendig, oder bei unzureichenden Kenntnissen über die genauen Hardwarebedingungen nicht anwendbar. In solchen Fällen müssen zusätzliche Hilfmittel herangezogen werden.

Der MICO 85 verfügt deshalb, wie viele andere Systeme auch, über ein spezielles Programm (debugger) zum Test der erstellten Software. Dieses sog. Einzelschritt-Testsystem wird durch das Hauptkommando E aktiviert.

Eine Vielzahl von Unterkommandos gestatten dann die Durchführung zahlreicher Tests und Einflußnahmen auf den Programmablauf.
Die wichtigste Fähigkeit solcher Testsysteme besteht in der Möglichkeit, ein zu testendes Programm im Einzelschritt, d.h. Befehl für Befehl, abarbeiten zu lassen. Auf diese Weise wird die Wirkung jedes einzelnen Befehles beobachtbar.

Beobachten lassen sich Wirkungen die nach außen gehen, aber auch mikroprozessorinterne Wirkungen, wenn das Testsystem die Inhalte der Register und Flags nach jedem Befehl speichert. Über spezielle Unterkommandos können diese Inhalte dann zur Anzeige gebracht und damit kontrolliert werden.

Stellt sich heraus, daß ein Inhalt anders lauten sollte, so ist er jederzeit veränderbar.

Beispiel 4.16: Programmabarbeitung mit einem Testsystem. (E) bedeutet, drück die Taste E.

Test des Unterprogrammes (ab Adr.1014H) aus Beisp.4.15:

Eingabe	Beschreibung	Anzeige
(E)(1014) :	Start des Testsystemes (E) von der Adresse 1014H an. Der erste Befehl (LXI H,1021) wird abgearbeitet und auf der Anzeige erscheint die Adresse des nächsten Befehles (1017) und das erste Byte (Operationsteil des Befehls) mit dem Wert 5F (MOV E,A).	EF.1017.5F
(F) :	Inhalt des Registerpaares HL anzeigen.	EF.1021.5F
(0) :	Nächsten Befehl abarbeiten (MOV E,A) und den übernächsten (MVI D,00 --→ 1600) anzeigen.	E0.1018.16
(E) :	Inhalt des Registers E inspizieren (z.B.7A). Dieser Wert ist für das UP unsinnig und muß geändert werden.	EE.1018.7A
(1)(05) :	Registerinhalt ändern (1) in den Wert 05.	E1.1018.05
(0) :	Nächsten Befehl abarbeiten (MVI D,00).	E0.101A.19
(0) :	Nächsten Befehl abarbeiten (DAD).	E0.101B.46
(0) :	Nächsten Befehl abarbeiten (MOV A,M).	E0.101C.C9
(A) :	Richtiger Akku-Inhalt (ASCII-Code von 5)?	EA.101C.35
(7)(1)(1014) :	Programmzähler (7) erneut auf die Anfangsadresse setzen (1) und neuer Durchlauf.	E7.101C.C9 E1.1014.21

Die Abarbeitung langer Programmteile, deren Wirkung bekannt ist (z.B. eine Zeitschleife), durch wiederholten Druck auf die Taste 0, kann sehr zeitraubend sein, deshalb gibt es Möglichkeiten dies zu umgehen. Zum einen kann der Inhalt des Programmzählers (Befehlszähler) verändert werden (Unterkommando, UK 7), dann müssen allerdings auch die Inhalte der Register und Flag geändert werden, welche in dem übersprungenen Programmteil beeinflußt wurden. Ein Beispiel hierfür enthält Beisp.4.16, denn dort wurden die ersten Befehle des Programmes aus Beisp.4.15b übersprungen und dafür später der in diesem Teil eingelesene Tastenwert (05) direkt ins Register E geschrieben (s.a. Beispiel 4.17).

Zum anderen ist es möglich, bestimmte Programmteile schnell durchlaufen zu lassen (real time, Echtzeit-Abarbeitung), d.h. mit der normalen Arbeitsgeschwindigkeit des Mikroprozessors. In diesem Fall wird über ein Unterkommando (UK 4) eine Haltepunkt-Adresse ins Testsystem eingegeben und dann über UK 5 der Freilauf gestartet. Sobald das zu testende Programm die vorgewählte Halteadresse (break point) erreicht, wird es gestoppt und die Kontrolle über das Programm wieder dem Testsystem übergeben.

Beispiel 4.17: Programm-Freilauf mit Haltepunkt.
Test des Programmes aus Beisp.4.15:

Eingabe	Beschreibung	Anzeige
(E)(1000) :	Start des Testlaufes ab Adr.1000 (CALL 0013), der 1. Befehl des UPs lautet JMP Adr. (C3....).	EF.0013.C3
(7)(1)(1003) :	Den Programmzähler auf 1003 setzen und damit das UP überspringen.	E7.0013.C3 E1.1003.47
(A)(1)(05) :	Akku mit dem Tastenwert 05 laden, da im obigen UP die Tastatur abgefragt worden wäre.	EA.1003.XX E1.1003.05
(4)(100A) :	Haltepunkt setzen.	E4.100A.47
(7) :	Programmzähler abfragen.	E7.1003.47
(5) :	Freilauf starten.	E5.100A.47
(A) :	Kontrolle, ob der Akku den ASCII-Code der Zahl 5 enthält.	EA.100A.35

Ein weiteres Problem beim Test eines Programmes kann darin bestehen, daß der Inhalt von Speicherplätzen oder von Ein-/Ausgabeports überprüft oder geändert werden muß. Sollte dies im Verlauf eines Tests erforderlich sein, so besteht häufig die Möglichkeit, zwischendurch in den allgemeinen Kommandobereich des Monitors zurück zu springen, dort entsprechende Aktivitäten zu entwickeln und dann, ohne Einfluß auf den Bearbeitungsstand des zu testenden Programmes, in das Testsystem zurück zu kehren.

Beispiel 4.18: Zwischenzeitliche Ausführung anderer Monitor-Hauptkommandos.

Weiterführung des Programmtests aus Beispiel 4.17, unter der Annahme, daß im letzten Schritt der Akku fälschlicher Weise den Wert 34 beinhaltet hat.

Eingabe	Beschreibung	Anzeige
(A) :	Akku-Inhaltskontrolle mit dem Ergebnis, daß dieser falsch ist.	EA.100A.34
(6) :	Rücksprung in den Monitor-Hauptkommandobereich mit dem Ziel, in der Programmtabelle nachzusehen, ob dort ein Fehler vorliegt.	E6.100A.34
(A)(1026) :	Inhalt des 6. Tabellenplatzes (Adr. 1026H) des Programmes aus Beisp.4.15b inspizieren. Annahme: dieser Wert ist falsch (34 statt 35).	AF.1026.34
(1)(35) :	Ändern dieses Wertes durch das UK 1 zum Hauptkommando (HK)A.	A1.1026.35
(3) :	Beendigung des HK A und Rücksprung ins Testsystem (HK E).	E7.100A.47

Die Benutzung solcher Testsysteme erfordert zunächst eine gewisse Einarbeitungszeit, dieser Aufwand wird jedoch später durch kürzere Programmtestzeiten vielfach wieder wett gemacht.

Der Testablauf läßt sich noch übersichtlicher gestalten, wenn ein Bildschirm zur Verfügung steht.

4.4 Übungsaufgaben und Beispielprogramme

4.1 siehe Kapitel 4.2

4.2 Blinkprogramm

Bei den meisten Mikroprozessor-Anwendungen ist es erforderlich, Steuersignale zu erzeugen. Dies bedeutet, daß auf einem oder mehreren Ausgabekanälen, für eine bestimmte Zeit, ein H- oder L-Signal anstehen muß, je nachdem ob das zu steuernde Gerät einen high- oder low-aktiven Steuereingang besitzt.

Ein träger Steuereingang - z.B. ein Relais - benötigt außerdem viel Zeit um anzusprechen, deshalb muß der Steuerimpuls eine bestimmte Länge haben. Hierzu gibt es bei einem Mikrocomputer Ausgabekanäle (Kap.9), die den einmal übermittelten logischen Zustand (L oder H) speichern und erst durch eine erneute Ausgabe verändert werden. Auf diese Weise lassen sich Impulse beliebiger Länge erzeugen.

Auch die Anzeigeeinheit des MICO 85 ist an einem Ausgabeport angeschlossen.

Eine Ausgabe auf die Anzeigeeinheiten 1 und 2 wird hierbei durch die Monitor-Service-Routine ANZ12 (Aufrufadresse 3F) vermittelt. Wenn auf dieser Anzeige nacheinander unterschiedliche Segmente zum Leuchten gebracht werden, entsteht ein Blinklicht, d.h. die Segmente werden mit einer bestimmten Frequenz ein- und ausgeschaltet. Beispiel 4.19 zeigt den Aufbau eines solchen Programmes.

Wichtigster Bestandteil ist die sog. Zeit- oder Warteschleife. Diese Zeitschleife, die hier als Unterprogramm ausgeführt ist, hat die Aufgabe, den Mikroprozessor für eine genau festgelegte Dauer daran zu hindern den nächsten Befehl im Hauptprogramm auszuführen. Der Mikroprozessor wird hierbei meistens damit beschäftigt, zu zählen. Im vorliegenden Beispiel (Befehlsadresse 101B) wird ins Register D die Zahl FF eingeladen und dann immer 1 abgezogen, bis der Inhalt 0 ist.

Die hierbei verstreichende Zeit soll jedoch noch verlängert werden, deshalb wird diese Zeitschleife so häufig ausgeführt, wie es der Inhalt des Registers A angibt. Dies bedeutet, daß beim Aufruf der Zeitschleife durch den Inhalt des Registers A festgelegt wird, nach welcher Zeit der Rücksprung ins Hauptprogramm erfolgt.

Beispiel 4.19: Programm zur Erzeugung einer blinkenden Anzeige (1 bzw.E) mit variabler Frequenz.

```
1000  CD1B00  CALL  001B  ;Tastatur abfragen -→(A) (Zeitparameter)
1003  07      RLC         ;(A)*2
1004  07      RLC         ;Akku-Inhalt insgesamt mit 16
1005  07      RLC         ;multiplizieren.
1006  07      RLC
1007  0611    MVI   B,11  ;  Anzeigezahl laden und auf die Anzeigen
1009  CD3F00  CALL  003F  ;  1+2 bringen
100C  CD1A10  CALL  101A  ;Zeitschleife
100F  00EE    MVI   B,EE  ;  Anzeigezahl laden und auf die Anzeigen
1011  CD3F00  CALL  003F  ;  1+2 bringen
1014  CD1A10  CALL  101A  ;Zeitschleife
1017  C30010  JMP   1000

      ;Unterprogramm Zeitschleife
101A  F5      PUSH  PSW   ;Akkuinhalt retten (Zeitparameter)
101B  16FF    MVI   D,FF  ;Innere Schleife 255 mal durchlaufen
101D  15      DCR   D     ;(D) - 1
101E  C21D10  JNZ   001D  ;Sprung, wenn (D) ungleich 0
1021  3D      DCR   A     ;(A) - 1
1022  C21B10  JNZ   101B  ;Sprung, wenn (A) ungleich 0
1025  F1      POP   PSW   ;Inhalt des Akkus zurück laden
1026  C9      RET         ;Zurück ins Hauptprogramm
```

Zur Beeinflussung der Blinkfrequenz muß der Inhalt des Registers A von außen veränderbar sein, hierzu dient der erste Unterprogramm-Aufruf. Über die Tastatur kann eine Zahl zwischen 00 und 0F in den Akku geladen werden. Würde diese Zahl direkt als Frequenzparameter benutzt, entständen recht hohe Frequenzen, deshalb wird der Akku-Inhalt durch 4malige Rotation nach links mit 16 multipliziert. Der Zeitschleifenparameter überstreicht dann in 16 Stufen den folgenden Wertbereich 00, 10, 20...F0.

4.3 Welcher Zeit- bzw. Frequenzparameter entsteht im Blinkprogramm (4.2) aus Beisp.4.19, wenn keine Taste gedrückt wird?

4.4 Wie lang ist der kürzeste Impuls, der sich mit der Zeitschleife aus Beisp.4.19 erzeugen läßt (3MHz Takt, 8085)?

4.5 Speichertest

Ein Vorteil beim Einsatz von Mikrocomputern besteht unter anderem darin, daß es mit ihrer Hilfe möglich ist, Selbstdiagnosen zu stellen. Hierunter versteht man die Fähigkeit, mittels spezieller Testprogramme Fehler im System zu erkennen und zu lokalisieren. Solche Tests kann der Mikrocomputer, z.B. jedes Mal wenn er eingeschaltet wird, durchführen.

Beispiel 4.20: Speichertest-Unterprogramm

```
                ;Die Testbereichs-Anfangsadresse steht im Reg. paar HL
                ;Die Testbereichs-Endadresse steht im  Registerpaar DE
0000 C5             PUSH B      ;Registerinhalt B + C sichern
0001 2B             DCX  H
0002 23       ANFA: INX  H
0003 46             MOV  B,M    ;Speicherinhalt im Reg. B sichern
0004 36AA           MVI  M,0AAH ;Speicherpl. mit HLHLHLHL füllen
0006 7E             MOV  A,M    ;Speicherinhalt lesen
0007 FEAA           CPI  0AAH   ;Wurde das richtige Byte gelesen?
0009 C21F00         JNZ  FEHLER ;Beendigung bei Fehler
000C 3655           MVI  M,55H  ;Speicherpl. mit LHLHLHLH füllen
000E 7E             MOV  A,M    ;Speicherinhalt lesen
000F FE55           CPI  55H    ;Wurde das richtige Byte gelesen?
0011 C21F00         JNZ  FEHLER
0014 70             MOV  M,B    ;Alten Speicherinhalt rückladen
0015 7C             MOV  A,H    ;  Sprung zum
0016 BA             CMP  D      ;  Anfang
0017 C20200         JNZ  ANFA   ;  wenn die
001A 7D             MOV  A,L    ;  Endadresse
001B BB             CMP  E      ;  noch nicht
001C C20200         JNZ  ANFA   ;  erreicht wurde.
001F C1       FEHLER:POP B
0020 C9             RET
```

Ein Beispiel hierfür ist der Speichertest. Mit seiner Hilfe ist es möglich defekte Speicherzellen im Schreib/Lese-Speicher (RAM) aufzufinden und dem Anwender zu melden.

Das Speichertest-Programm in Beisp.4.20 wird als Unterprogramm aufgerufen. Vor dem Aufruf muß durch die Registerpaare HL und DE der zu testende Bereich definiert werden. Damit der Mikrocomputer auch während seines normalen Betriebes die Speicher testen kann, wird der Inhalt des zu testenden Speicherplatzes zunächst im Register B gesichert und später wieder zurückgeladen.

Beim Test des Speicherplatzes muß jede der 8 Speicherzellen einmal den Zustand L und H annehmen. Hierzu werden entsprechende Daten in den Speicher eingeschrieben und zur Kontrolle wieder gelesen. Wird beim anschließenden Vergleich des gelesenen und des eingeschriebenen Bytes keine Übereinstimmung festgestellt, erfolgt ein Rücksprung ins Hauptprogramm. Anhand des Null-Kennzeichenbits (zero flag) kann im Hauptprogramm festgestellt werden, ob ein Speicherfehler vorliegt und damit eine entsprechende Reaktion erfolgen. Bei einem Speicherfehler ist das Null-Kennzeichenbit immer L, da der Vergleich ja ungleich Null ergab. Im Registerpaar HL steht dann die Adresse des defekten Speicherplatzes.

Auch der MICO 85 macht nach jedem Reset einen Speichertest und zeigt das Ergebnis an. Hierzu sucht das Speicher-Testprogramm zunächst die niedrigste Adresse eines RAM-Speicherplatzes und zeigt sie auf den linken 4 Anzeigeeinheiten an, dann wird dieser gefundene Speicherbereich solange weiter getestet, bis die Endadresse feststeht und diese auf den vier rechten Anzeigeeinheiten dargestellt: z.B. 1000 17FF.

Durch das Kommando O kann diese Suche fortgeführt werden, um weitere RAM-Speicherbereiche zu finden und anzeigen zu lassen: z.B. 2000 27FF (Arbeitsspeicher des Monitors).

4.6 Was geschieht, wenn bei dem Speichertestprogramm aus Beisp.4.20 die Endadresse kleiner als die Anfangsadresse angegeben wird?

4.7 Schreiben Sie ein Programm, welches beim MICO 85 feststellt, wo ein RAM-Speicherbereich beginnt und bei welcher Adresse er endet.

Stellen Sie beide Adressen auf der Anzeigeeinheit nebeneinander dar.

4.8 Kopierprogramm

Manchmal tritt das Problem auf, den Inhalt eines Speicherbereiches in einen anderen Adreßbereich zu übertragen. Wenn hierbei der Inhalt des ursprünglichen Bereiches erhalten bleibt, wird dieser Vorgang 'kopieren' genannt (Kommando C beim MICO 85).
Soll dieser Kopiervorgang zu einer beliebigen Zieladresse hin möglich sein, müssen bestimmte Sonderfälle berücksichtigt werden. Hierbei handelt es sich um Überlappungen zwischen dem Quell- und dem Ziel-Speicherbereich.

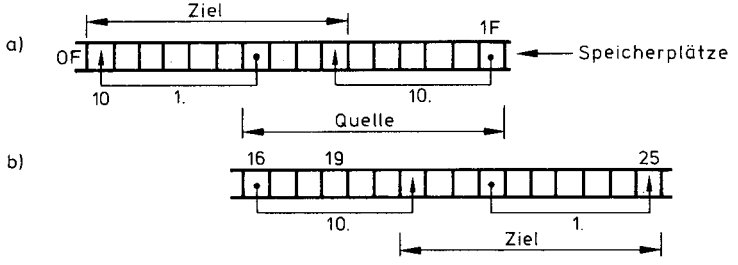


Abb.4.4: Kopien mit Überlappung:
a) zu niedrigeren Adressen hin, b) zu höheren Adressen hin.

Ein universelles Kopierprogramm muß zu Beginn prüfen, ob eine Überlappung der Bereiche am Anfang oder am Ende der Quelle vorliegt. In Abb.4.4a ist eine Überlappung am Anfang des Quellbereiches dargestellt. Der Inhalt der Speicherplätze mit den Adressen 16H bis 1FH soll in den Adreßbreich 10H bis 19H kopiert werden. Hierbei wird allerdings, infolge der Überlappung, der Quelladreßbereich von 16H bis 19H überschrieben und damit der alte Inhalt zerstört.
Es ist wichtig, in diesem Zusammenhang zu beachten, daß die Inhalte hierbei, mit der niedrigsten Adresse beginnend, kopiert werden müssen.

Beispiel 4.21: Speicherbereich-Kopierprogramm

```
      ;Anfangsadresse "A" des zu kopierenden Bereiches in HL
      ;Endadresse "E"  des  zu  kopierenden Bereiches  in BC
            ;Anfangsadresse des Zielbereiches "ZA"  in DE

0000 E5       KOPIE: PUSH  H        ; A
0001 C5              PUSH  B        ; E
0002 D5              PUSH  D        ; ZA
0003 C1              POP   B
0004 08              DB    8;(DSUB); A - ZA
0005 C1              POP   B        ;  Spring, wenn die Zieladresse
0006 E1              POP   H        ;  (ZA) kleiner ist als die
0007 D21500          JNC   KLEIN    ;  Anfangsadresse (A).
000A F5              PUSH  PSW
000B C5              PUSH  B
000C E5              PUSH  H
000D C5              PUSH  B
000E E1              POP   H        ; E
000F C1              POP   B        ; A
0010 08              DB    8;(DSUB); (HL) = E - A
0011 19              DAD   D        ;      = E - A + ZA = ZE
0012 EB              XCHG           ; (DE) = Zielendadresse ZE
0013 E1              POP   H        ; E
0014 F1              POP   PSW

0015 F5       KLEIN: PUSH  PSW      ;Carry Flag sichern
0016 7E              MOV   A,M      ;  Erstes Byte
0017 12              STAX  D        ;  kopieren und
0018 EB              XCHG           ;  vergleichen ob
0019 BE              CMP   M        ;  richtig angekommen , sonst
001A C23200          JNZ   ENDE     ;  Abbruch -> Speicherfehler!
001D EB              XCHG
001E E5              PUSH  H        ; Ist das  Ende
001F 08              DB    8;(DSUB); des zu kopierenden
0020 E1              POP   H        ; Bereiches erreicht,
0021 CA3200          JZ    ENDE     ; dann Sprung zum Ende.
0024 F1              POP   PSW      ;Carry-Flag laden
0025 D22D00          JNZ   HOCH     ; Sprung, wenn zu einer kleineren
0028 2B              DCX   H        ; Adresse hin kopiert wird.
0029 1B              DCX   D
002A C31500          JMP   KLEIN    ;Nächstes Byte kopieren

002D 23       HOCH:  INX   H
002E 13              INX   D
002F C31500          JMP   KLEIN    ;Nächstes Byte kopieren

0032 33       ENDE:  INX   SP
0033 33              INX   SP
0034 C9              RET
```

Im Gegensatz dazu muß, bei einer Überlappung am Ende des Quellbereiches, der Kopiervorgang mit der höchsten Adresse beginnen und dann Byte für Byte durchgeführt werden. Andernfalls würden im Quellbereich Inhalte durch die Überlappung zerstört.

Das Kopier-Programm im Beisp.4.21 ist universell einsetzbar und gestattet es auch überlappende Kopien anzufertigen. Zur Verkürzung des Programmes wurde der 8085-Befehl DSUB ((HL) = (HL) - (BC)) eingesetzt. Da der Assembler diesen Befehl nicht kannte, mußte über die Assembleranweisung DB der Objektcode dieses Befehles (08) direkt angegeben werden.

Um herauszubekommen, ob ein in den Zielbereich gesendetes Byte dort auch angekommen ist, wird jeweils eine Kontrollesung durchgeführt. Ist das Ergebnis negativ, so erfolgt ein Abbruch des Programmes. Das Hauptprogramm, in das dann der Rücksprung erfolgt, erkennt am Inhalt des Null-Kennzeichenbits - Speicherfehler = L - ob ein Fehler aufgetreten ist.

Ein Fehler tritt z.B. auf, wenn der Speicher im Zieladressbereich aus einem ROM (Lesespeicher) besteht.

Dieses Programm wird auch im Monitor des MICO 85 benutzt.

5. Programm – Entwicklungssysteme

Nachdem im vorangegangenen Kapitel gezeigt wurde, wie mit einem kleinen Mikrocomputersystem - unter Zuhilfenahme leistungsfähiger Dienstprogramme - Programme im Hex-Code erstellt werden können, wird hier beschrieben, welche zusätzlichen Hilfsmittel das Programmieren noch weiter erleichtern.

In Abb.5.1 ist zu sehen, daß prinzipiell jeder Computer zur Entwicklung von Mikroprozessorprogrammen herangezogen werden kann.

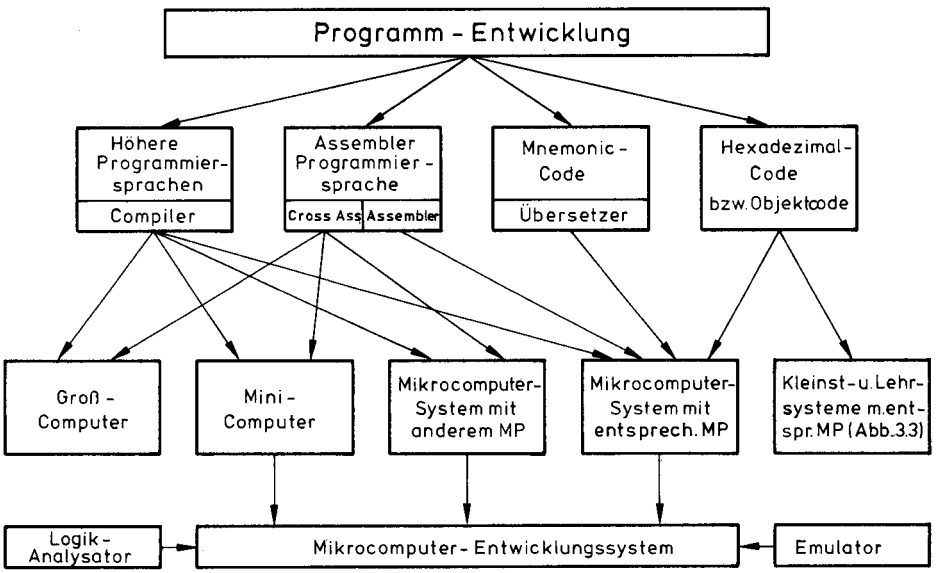


Abb.5.1: Möglichkeiten zur Entwicklung von Mikroprozessor-Programmen

Ein wichtiger Unterschied besteht jedoch darin, ob das zu entwickelnde Programm in dem benutzten Computer ablauffähig ist oder nicht.

Soll ein Programm,welches mit Hilfe eines Computers entwickelt wird, in diesem gestartet werden können, oder gar für ihn selbst bestimmt sein, so muß er den entsprechenden Mikroprozessor beinhalten. Wenn dagegen der zur Programmentwicklung benutzte Computer nicht über den gleichen Mikroprozessor verfügt, ist es erforderlich, das durch den Compiler oder den Cross-Assembler übersetzte Programm - d.h. den Objektcode - über ein Speichermedium in den Mikrocomputer zu bringen für den es geschrieben wurde.

Das Programm, welches ein in der Assembler-Programmiersprache erstelltes Programm in den Objektcode übersetzt nennt man einen Assembler, wenn das System den gleichen Mikroprozessor enthält, und einen Cross-Assembler, wenn der zur Programmentwicklung herangezogene Computer einen anderen, oder garkeinen MP enthält.

Die Eingabe von Programmen im Hex-Code (Objektcode) und im Mnemonic-Code, wurde bereits in den Kapiteln 3.u.4. beschrieben.

Dieses Kapitel dagegen enthält Hinweise und Erklärungen zur Anwendung der Assembler-Programmiersprache.

Der Einsatz höherer Programmiersprachen, bei denen z.B. ein in der Programmiersprache FORTRAN 80 geschriebenes Programm von einem speziellen Compiler in den Objektcode des MP-8080 übersetzt wird, soll hier nicht weiter behandelt werden. Dem Vorteil der besseren Übersichtlichkeit beim Programmieren, steht ein wesentlich längeres Objektcode-Programm entgegen, als wenn das gleiche Problem in der Maschinensprache (Assembler) programmiert worden wäre. Hierdurch werden erheblich mehr Speicherplätze belegt und die Programmausführungszeiten drastisch verlängert.

Ein Programm-Entwicklungssystem heißt Mikrocomputer-Entwicklungssystem, wenn es durch verschiedene elektronische Testhilfen ergänzt wird. Zwei der wichtigsten Ergänzungen sind der Emulator (in-circuit-emulator) und der Logik-Analysator. Mit Hilfe dieser Testgeräte kann der neu zu entwickelnde Mikrocomputer an das Entwicklungssystem angeschlossen werden, und es sind dann umfangreiche Tests der Programme und der zu entwickelnden elektronischen Schaltung möglich.

Die Entwicklungssysteme wurden früher ausschließlich von den Herstellern der Mikroprozessoren angeboten und enthielten den gleichen Mikroprozessor. Dies hat den Vorteil, daß die zu entwickelnden Programme direkt auf dem Entwicklungssystem ablauffähig sind und getestet werden können. Desweiteren ist es dem qualifizierten Anwender möglich, Hilfsprogramme für sein Entwicklungssystem zu schreiben, die seinen speziellen Bedürfnissen gerecht werden.

Seit einigen Jahren werden jedoch auch verstärkt Entwicklungssysteme von Meßgeräte-Herstellern angeboten. Diese Systeme bestehen im allgemeinen aus Minicomputern und es stehen Cross-Assembler für alle gängigen Mikroprozessoren zur Verfügung.

Ein wesentlicher Vorteil dieser Entwicklungssysteme ist zur Zeit vorallem in ihrer leichteren Bedienbarkeit zu sehen.

Die Entwicklung von Programmen läßt sich jedoch auch mit Hilfe sog. Personal-Computer durchführen. Ein Personal-Computer besteht aus einem Mikrocomputer, der, mit Tastatur und Bildschirm versehen, als Tischrechner Anwendung findet; doch hierüber mehr im nächsten Kapitel.

5.1 Personal-Computer

Unter dem Begriff Personal-Computer, oft auch einfach PC genannt, versteht man heute Mikrocomputersysteme von einfachen bis hin zu recht komplexen Ausstattungen, die folgendes gemeinsam haben:

- Aufgebaut auf einem Mikroprozessor,
- als Eingabeeinheit dient eine ASCII-Tastatur (Kap.10.1),
- als Ausgabeeinheit ein Bildschirm (Kap.10.3),
- es sind zusätzliche periphere Geräte, wie z.B. Massenspeicher (Kassettengeräte, Floppy, Kap.10.5) und Drucker (Kap.10.4) anschließbar,
- und als Grundsoftware wird ein BASIC-Interpreter mitgeliefert, um das System als Tischrechner einsetzen zu können.

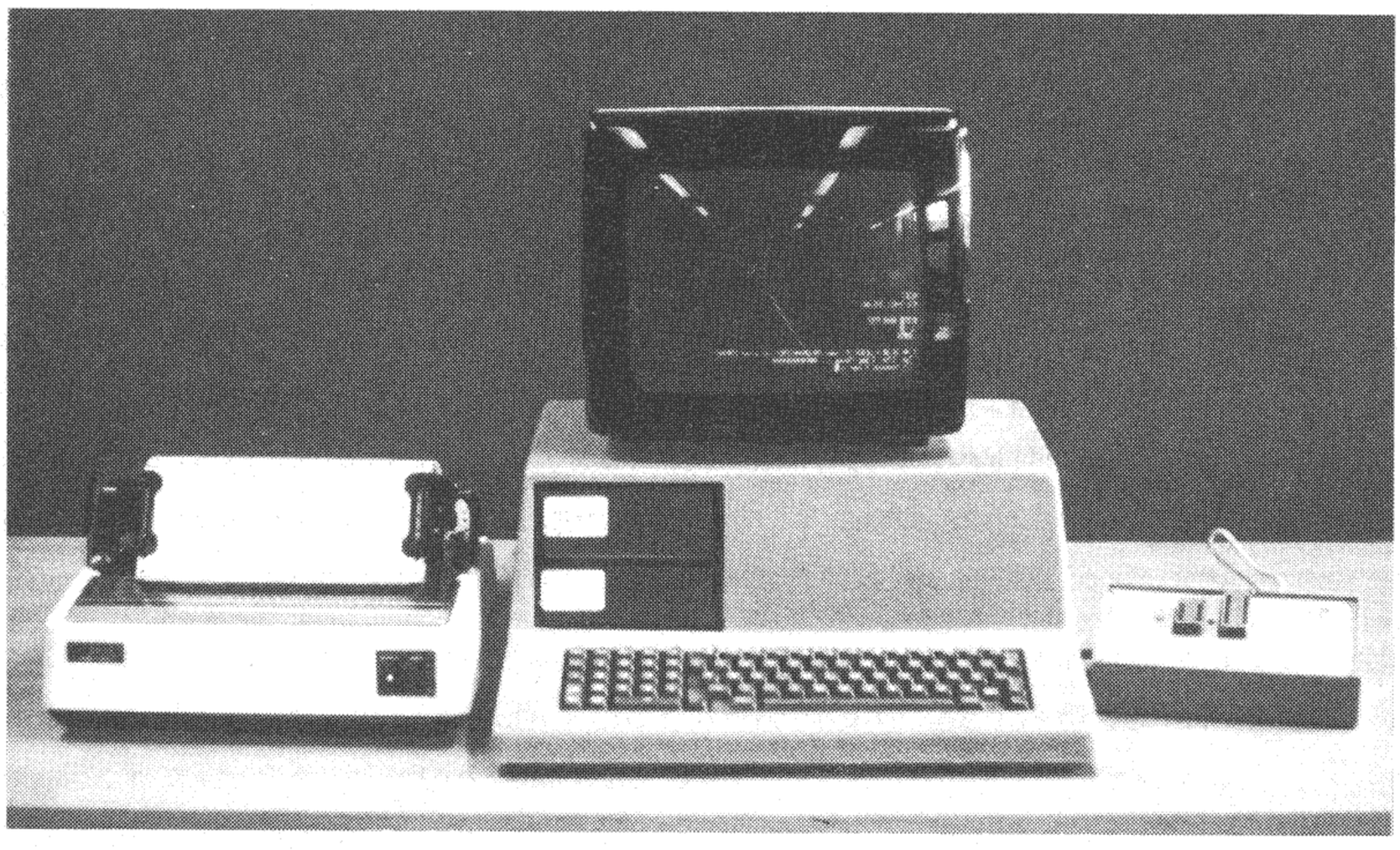


Abb.5.2: Programm-Entwicklungssystem auf der Basis eines Personal-Computers

Die Abb.5.2 zeigt einen solchen Personal-Computer mit zwei integrierten Floppy-Laufwerken, einem Drucker und einem Gerät (links neben dem PC) zur Einbringung eines entwickelten Programmes in einen Halbleiterspeicherbaustein (EPROM, Kap.8.4).

Der in Abb.5.2 gezeigte Personal-Computer stellt ein System mittlerer Komplexität dar. Einfachere Systeme enthalten häufig keinen eigenen Bildschirm, sondern lassen sich an jedes Fernsehgerät anschließen. Die Auflösung (Bildschärfe) ist dann jedoch geringer. Auch haben diese Kleinstsysteme als Massenspeicher meistens nur einen Audiokassettenrecorder.

Die leistungsstärksten Personal-Computer verfügen über Farbbildschirme und die Möglichkeit zur Darstellung farbiger Grafiken in 8 Farben, mit einer Auflösung von z.B. 520x520 Punkten; Floppy-Laufwerken zur Aufzeichnung von bis zu 1Millionen Byte pro Diskette und zusätzlichen Festplattenlaufwerken mit einer Kapazität von einigen 10 Megabytes. Außerdem enthält die neue Generation bereits 16bit-Mikroprozessoren (z.B. 8086) und zusätzlich einen Arithmetikprozessor (z.B. 8087) zur schnelleren Ausführung hochgenauer Berechnungen (64 bis 80bit). Der Anwenderspeicher kann dann bis auf 1 MB ausgebaut werden.

Die mittleren und großen PC-Systeme (1983: 4-10TDM) finden vorallem im Bürobereich (kaufmännisch, technisch, wissenschaftlich) Anwendung. Für diesen Einsatzbereich stehen Betriebssysteme (Kap.5.2), alle wichtigen höheren Programmiersprachen (FORTRAN, ALGOL, PASCAL, COBOL, etc.), sowie unzählige spezielle Programmpakete zur Verfügung.

Die kleinen Systeme (1983: 200-2000DM) wurden vorallem für den privaten Anwender und den Hobby-Bereich konzipiert.

Durch die Verwendung alphanumerischer Tastaturen und eines Bildschirmes, lassen sich gegenüber den Systemen aus Kap.4 komplexere und übersichtlichere Ein-/Ausgaben realisieren.

Der Cursor

Zur Anzeige dafür, an welcher Stelle des Bildschirmes das nächste Zeichen sichtbar gemacht wird, erscheint an dieser Stelle ein sog. Cursor-Zeichen. Üblich ist die Verwendung eines leuchtenden oder blinkenden Rechteckes oder Striches als Cursor-Zeichen. Wird z.B. ein Kommando eingegeben (s.a. Kap.4.2), so erscheint - nach dem Druck auf eine Taste die das Kommando aktivieren soll (z.B. A) - an der Stelle des Cursors z.B. ein A auf dem Bildschirm und der Cursor geht um eine Position weiter nach rechts.
Häufig ist es sehr hilfreich, wenn der Cursor frei über den gesamten Bildschirm bewegt werden kann. Wenn z.B. ein Text auf dem Bildschirm steht und es sollen Veränderungen an ihm vorgenommen werden, dann kann man den Cursor an die entsprechende Stelle bewegen und dort direkt neue Zeichen im Text erzeugen.
Zur Bewegung des Cursors über den Bildschirm verfügt die Tastatur über 5 spezielle Tasten:

Cursor rauf, runter, nach rechts, nach links, zurück

↑ ↓ → ← ↖ (HOME)

Die Betätigung einer der ersten 4 Tasten rückt den Cursor jeweils um eine Position in die angegebene Richtung. Die Taste "zurück" bedeutet, daß der Cursor zu der Position zurück kehrt, von der aus er wegbewegt wurde.

Programm-Entwicklungssystem

Während spezielle Mikrocomputer-Entwicklungssysteme meistens recht teuer sind (1983: 30-100TDM), kann auch ein Personal-Computer zu dieser Aufgabe herangezogen werden (Preis für einen mittleren PC, 1983: 5TDM). In jedem Fall muß jedoch der Personal-Computer noch um eine periphere Einheit (Kap.11.4) erweitert werden, die es gestattet, den Objektcode (Hex-Code) des erstellten Programmes in einen Halbleiter-Speicherbaustein zu schreiben. Dieser Baustein (z.B. EPROM, Kap.8.4) wird dann in die Schaltung gesteckt, für die das Programm entwickelt wurde.

Soll das Programm in der neu zu entwickelnden Schaltung, unter der Kontrolle des PC, getestet werden können, so gibt es auch hierzu Erweiterungsgeräte, die an den PC anzuschließen sind und einen sog. In-circuit-Test erlauben.

Dies sind die Hardware-Voraussetzungen für ein Entwicklungssystem, auf der Basis eines Personal-Computers.

Zusätzlich werden noch leistungsfähige Dienstprogramme benötigt. Hierzu mehr im folgenden Kapitel.

5.2 Dienstprogramme und Betriebssysteme

Durch das Einschreiben spezieller Programme in den Speicher des Mikrocomputers, erhält dieser bestimmte Fähigkeiten.

Aktivieren lassen sich solche "erlernten" Fähigkeiten durch die Erteilung von Kommandos (Kap.4.2) an das Programm.

Bei Bildschirm-Computersystemen werden diese Kommandos über die Tastatur eingegeben und auf dem Bildschirm dargestellt. Während bei den Kommandos in Kap.4.2, nach der Eingabe des letzten Zeichens, sofort die Bearbeitung begann, wird hierzu bei den größeren Systemen meistens die Eingabe eines speziellen Kommando-Abschlußzeichens verlangt. Hierdurch ist es möglich, ein eingegebenes Kommando - durch Benutzung der Cursor-Tasten - noch zu korrigieren, ehe es abgearbeitet wird. Als Startzeichen für die Bearbeitung des Kommandos (Kommandoabschluß) wird häufig die Betätigung der Taste "Neue Zeile" (RETURN, SP,..)verlangt.

Im folgenden wird kurz auf den Monitor und das Editor-Dienstprogramm eingegangen. Dem Assembler ist, wegen seiner großen Bedeutung, das ganze nächste Kapitel gewidmet.

Der Monitor

Bereits in Kap.4.2 wurden die Aufgaben und der Aufbau eines Monitor-Programmes beschrieben.
Bei größeren Mikrocomputersystemen, wie z.B. den Personal-Computern, hat der Monitor meistens noch einige zusätzliche Aufgaben zu erfüllen. Diese weiteren Aufgaben resultieren aus dem größeren Hardware-Umfang und beinhalten z.B.:

- Zeichen von der Tastatur einlesen und auf dem Bildschirm sichtbar machen,
- Programme aus dem Massenspeicher in den Anwenderspeicher zu bringen,
- Programmtreiber für den Drucker bereitstellen uvm.

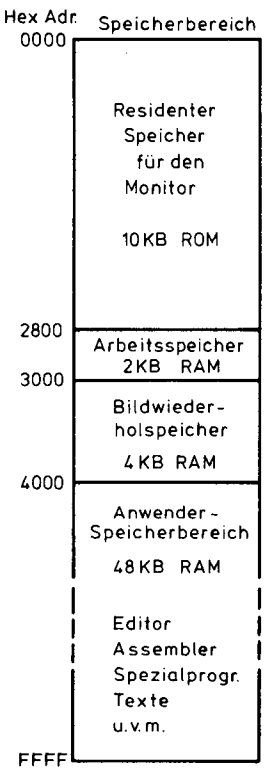


Abb.5.3: Beispiel einer Speicheraufteilung

Wenn, wie beim MP-8080/85, 16 Adreßleitungen und damit 64KB Adreßraum zur Verfügung stehen (s.a. Kap.4.1),kann z.B. die in Abb.5.3 dargestellte Speicheraufteilung Anwendung finden (vgl. auch Abb.4.1).

An den Festwertspeicherbereich (ROM) für den Monitor, grenzt ein Arbeitsspeicher (RAM) der unter anderem den Stapel (stack), sowie Tabellen und Variable aufnimmt.

Der Bildwiederholspeicher dient zur Abspeicherung des ASCII-Codes derjenigen Zeichen, die auf dem Bildschirm sichtbar sein sollen (Kap.10.3). Dieser Speicher kann jedoch auch außerhalb des Computers, in einem separaten Gerät, angeordnet sein und belegt dann keinen Adreßbereich des Computers.

Hieran schließt sich der sog. Anwenderspeicherbereich an. Dieser Schreib/Lese-Speicher kann aus dem Massenspeicher (z.B. Floppy-Disk) mit verschiedenen Programmen geladen werden. Über den Monitor lassen sich diese Programme dann starten, d.h. es wird ein Sprung aus dem Monitor, auf die Anfangsadresse des entsprechenden Programmes durchgeführt.

Dieser Speicherbereich nimmt auch die zur Entwicklung von Mikroprozessor-Programmen nötigen Dienstprogramme auf.

Aber auch die zu entwickelnden Programme werden dort in ihrer Quellform und als Objektcode abgelegt.

Die Aktivierung der Fähigkeiten des Monitors geschieht über Kommandos, wie bereits in Kap.4.2 dargelegt wurde.
Eine weitere wichtige Aufgabe des Monitors ist es, die Datenströme zu steuern. Hierzu gibt es ebenfalls verschiedene Kommandos.
In Abb.5.4 sind die wichtigsten Datenströme, die der Monitor verwalten muß, dargestellt. Hierzu besitzt der Monitor Programmteile, die man als Strom-Routinen und Geräte-Treiber bezeichnen kann. Die Verbindung dieser Programmteile geschieht über Schalter.
Ein solcher Schalter besteht beispielsweise aus einem Datenbyte, das auf einem bestimmten Arbeitsspeicherplatz abgelegt ist.
Die einzelnen Bits dieses Schalterbytes charakterisieren die verschiedenen Schalterstellungen.

Beispiel 5.1: Schalterbyte zu Abb.5.4

Schalter:	5	4	3	2	1	0
Byte:	X	L	L	L	L	L	L	L

L: Schalterstellung 0
H: Schalterstellung 1

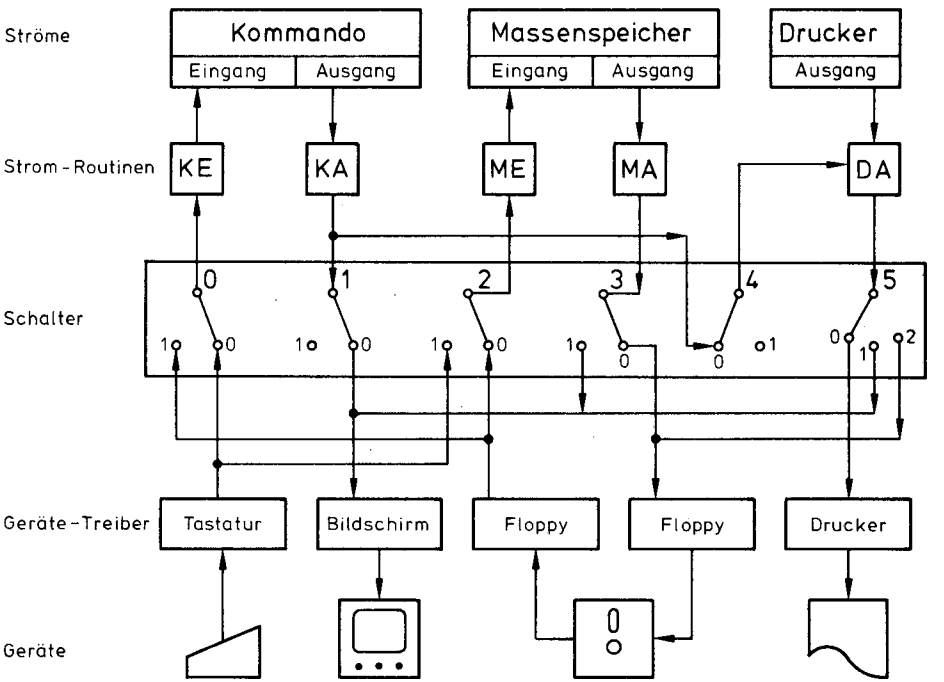


Abb.5.4: Datenströme in einem Mikrocomputersystem

Da der Monitor zur Lenkung der Ströme das Schalterbyte abfragt, kann durch eine Veränderung der einzelnen Bits, in einem bestimmten Bereich, eine Umlenkung der Ströme erreicht werden.

Der Datenstrom zum Bildschirm kann - z. B. durch den Schalter 4 - gleichzeitig auch auf dem Drucker zur Ausgabe gelangen. Oder die Kommandos an den Monitor können, statt über die Tastatur, direkt von der Floppy eingelesen werden (Schalter 0).
Weiterhin gibt es dann eine Reihe von Kommandos, die direkt und ausschließlich, oder nebenbei, auf diese Schalter wirken.

Beispiel 5.2: Einlesen und Start eines Programmes

Ein solches Kommando wird häufig BATCH MODE genannt.
Der Schalter 0 nimmt dann die Stellung 1 an, so daß Kommandos von der Floppy direkt an den Monitor gegeben werden können.

Diese Kommandos stehen, als sog. Batch-Vorspann, vor dem eigentlichen Programm auf der Floppy und werden vom Monitor zunächst eingelesen. Danach weiß der Monitor, wohin er das dahinter stehende Programm laden soll und wie es gestartet wird.

Die hier dargestellten Aufgaben des Monitors eines Mikrocomputers, stellen nur eine Auswahl dar. In Abhängigkeit vom elektronischen Aufbau und den angeschlossenen Peripherie-Geräten, können wechselnde Anforderungen an den Monitor gestellt werden.

Der Editor

Der Editor - oder auch Text-Editor - stellt ein spezielles Dienstprogramm zum Einschreiben von Texten in den Anwenderspeicher des Computers dar.
Zunächst wird das Editor-Programm mit Hilfe des Monitors vom Massenspeicher in den Anwenderspeicher gebracht und gestartet (Beisp.5.2).
Danach übernimmt der Editor - d.h. das gestartete Programm - die Kontrolle über den Computer. Hierzu bedient sich der Editor allerdings der Service-Routinen des Monitors.
Genau wie dem Monitor, können nun dem Editor Kommandos erteilt werden. Ausgeführt werden diese Kommandos z.B., nachdem die Taste "Neue Zeile" gedrückt worden ist.

Beispiel 5.3: Texteingabe-Kommandos

EI: Einschreiben neuer Texte.

Auf dem Bildschirm erscheint die erste Zeilennummer (0001) und es kann ein fortlaufender Text eingegeben werden. Z.B.:

```
0001  ;Unterprogramm HEX - ASCII
0002  HEXAS: ANI  0FH    ;Höherwertige Tetrade von A=0
0003         CPI  0AH    ;Buchstabe?
usw.
```

Die Texteingabe wird durch ein Unterkommando abgeschlossen. Hierzu kann nur eine Taste, oder Tastenkombination dienen, die keinem sichtbaren Zeichen zugeordnet ist (z.B. Steuerzeichen).

H1: Hereinholen des Textes Nr.1 vom Massenspeicher.

Die eingegebenen Textzeilen legt der Editor im Anwenderspeicher ab.

Dieser Speicherbereich beginnt beispielsweise hinter dem Editorprogramm, d.h. wenn der Editor den Speicherbereich von 4000-5FFF belegt, so kann er ab 6000H die Texte abspeichern.
Bei einem Programm-Entwicklungssystem haben die Textzeilen meistens die Bedeutung einer Befehlszeile. Der Programmierer gibt also eine Befehlszeile nach der anderen über die Tastatur in den Computer ein. Die einzelnen Zeichen werden dann sowohl auf dem Bildschirm angezeigt, als auch - codiert im ASCII-Code - in den Arbeitsspeicher gebracht. Das Ende einer Zeile wird durch die zwei Steuerzeichen OD und OA ("geh zum Zeilenanfang" und "nächste Zeile") abgeschlossen.

Beispiel 5.4: Speicherbelegung durch den Text aus Beisp.5.3

Adresse	Inhalt
6000	3B 55 6E 74 65 72 70 72 6F 67 72 61 6D 6D 20 48
6010	45 58 20 2D 20 41 53 43 49 49 0D 0A 48 45 58 41
6020	53 3A 20 41 4E 49 20 20 30 46 48 20 20 20 20 3B
6030	48 7C 68 65 72 77 65 72 74 69 67 65 20 54 65 74
6040	72 61 64 65 20 76 6F 6E 20 41 3D 30 0D 0A 20 20
6050	usw.

Das Editieren dient damit der Erstellung des Quellprogrammes. Die einzelnen Befehle werden hierbei noch nicht als solche erkannt, oder interpretiert. Ein ablauffähiger Objektcode wird erst später durch den Assembler erzeugt.

Betriebssysteme

Der nutzbringende Einsatz von Mikrocomputern, insbesondere auf der Basis der Personal-Computer, wird stark behindert durch die Vielzahl der verschiedenen Mikroprozessoren und der unterschiedlichen, daraus aufgebauten, Mikrocomputer.
Ein Programm, welches für einen speziellen Mikrocomputer-Typ geschrieben wurde, läuft i.a. auf einem anderen nicht ab. Dies liegt nicht nur an den unterschiedlichen Mikroprozessoren, sondern auch an den verschiedenen Systemkonfigurationen beim Einsatz des gleichen Mikroprozessors.
Es gibt also zwei Stufen der Unverträglichkeit: Die eine ist bedingt durch die unterschiedlichen Befehlscodes der verschiedenen Mikroprozessoren und die andere durch die differierenden Systemkonfigurationen der einzelnen Mikrocomputer.

Was kann man dagegen unternehmen?

Wenn der GLEICHE MIKROPROZESSOR Anwendung findet, ist der von einem Assembler erzeugte Objektcode jedem der Systeme verständlich. Probleme ergeben sich nur bei der Benutzung von Schnittstellen, denn diese können bei den einzelnen Mikrocomputern sehr unterschiedlich beschaffen sein. Wenn also z.B. mathematische,oder organisatorische Unterprogram-

me - die keine Hardware-Ein-/Ausgaben verlangen - geschrieben werden, so sind diese auf jedem der Computer ablauffähig und können in andere Programme mit eingebunden werden. Wenn ein solches Unterprogramm nicht in einem Schreib-/Lesespeicher (RAM) zur Ausführung gelangt, sondern in einem Festwertspeicher (ROM, Nur-Lese-Speicher), kann es jedoch noch erforderlich werden, dem Programm einen gesonderten Arbeitsspeicherbereich für die Ablage von Zwischenwerten etc. zur Verfügung zu stellen. Der Standardisierung der Ein-/Ausgabe-Schnittstellen, wie z.B. für Tastatur, Bildschirm, Massenspeicher, bei gleichem Mikroprozessor, dienen die sog. Betriebssysteme.

Wenn dagegen außerdem noch die MIKROPROZESSOREN UNTERSCHIEDLICH sind, kann ein Programm nur dann universell eingesetzt werden, wenn es nicht im Objektcode, sondern in Quellform vorliegt und durch einen Compiler (Übersetzer) des jeweiligen Mikrocomputers zunächst in den entsprechenden Objektcode übersetzt, oder durch einen Interpreter abgearbeitet wird.

Eine Übersicht über die vier möglichen Zustände zeigt Abb.5.5.

	Gleicher MP	Ungleicher MP
Gleiches System	Ein Programm, hierfür geschrieben, kann als Objektcode-Programm (Hex-Code), z.B. von einer Diskette eingelesen, direkt gestartet werden.	Die Programme sollten in einer höheren Programmiersprache geschrieben sein und müssen vor der Abarbeitung im System, von diesem durch einen entsprechenden Compiler übersetzt, oder von einem Interpreter abgearbeitet werden.
Ungleiches Sytem	Wenn ein gemeinsames Betriebssystem vereinbart wurde, kann wie oben verfahren werden, andernfalls müssen zumindest die Ein- und Ausgaberoutinen neu geschrieben werden.	Eventuell sind einige Anpassungen, des in einer höheren Programmiersprache geschriebenen Programmes, erforderlich, ansonsten wird wie oben verfahren.

Abb.5.5: Programmkompatibilität bei verschiedenen Systemarten

Wir wollen uns im folgenden erneut auf den MP-8080/85 beschränken.

Für diese beiden Mikroprozessoren und gleichzeitig auch für den MP-Z80, denn alle drei sind im wesentlichen softwarecompatibel, gibt es ein weltweit sehr verbreitetes Betriebssystem, welches eine weitgehende Austauschbarkeit aller Programme gestattet (s.Beisp.5.5).

Ein solches Betriebssystem stellt im wesentlichen eine Ansammlung von Unterprogrammen - ähnlich wie der Monitor - und fest vorgegebene, unveränderliche Konventionen für die Benutzung dieser Programme dar.

Das Betriebssystem wird dann beim Start des Computers vom Massenspeicher (Diskette) in den RAM-Speicher geladen und automatisch gestartet. Dieser Vorgang wird, nach dem Einschalten, von einem kleinen Programm übernommen, das man den Urlader (bootstrap) nennt.

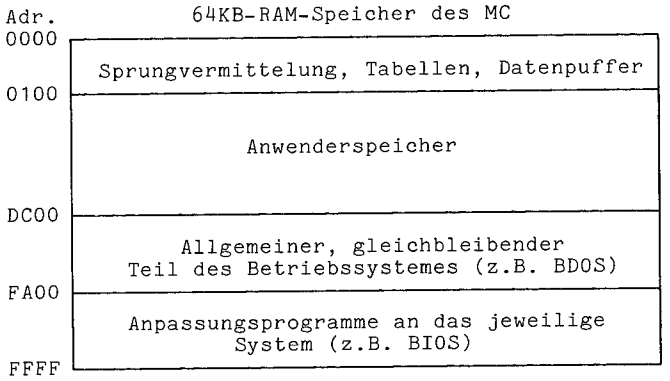


Abb.5.6: Beispiel für eine Speicheraufteilung bei der Benutzung eines Betriebssystemes. Die Abkürzungen BDOS und BIOS werden in Beisp.5.5 erklärt.

Der allgemeine Teil des Betriebssystemes enthält einen Kommandoteil, ähnlich dem in Kap.4.2 für einen Monitor beschriebenen, welcher nach dem Start vom Bediener weitere Kommandos erwartet.

Z.B. ist es möglich in den Anwenderspeicher (z.B. ab Adresse 0100, wie in Abb.5.6) ein fertiges Programm von einer Diskette aus einzuladen und zu starten.

Wurde dieses Programm für den entsprechenden Mikroprozessor und das entsprechende Betriebssystem geschrieben, so ist es auf jeder Systemkonfiguration - mit gewissen Minimalanforderungen - ablauffähig. Dies wird durch die im Betriebssystem unveränderbar festgelegten Konventionen für die Benutzung der Unterprogramme erreicht. Das eingeladene Programm bedient sich nämlich der speziellen, systemabhängigen Programmteile des Betriebssystemes, durch die Anwendung der in jedem System gleichen Sprungvermittler. Diese Technik wurde bereits bei der Beschreibung des Monitors in Kap.4.2 dargestellt.

Zur Anpassung des Betiebssystemes an ein anderes Mikrocomputersystem werden die systemspezifischen Programme, wie z.B. die Ein- und Ausgabe-Routinen für die Tastatur, den Bildschirm, den Drucker und den Massenspeicher neu geschrieben und dabei die vom Betriebssystem vorgegebenen Konventionen (Datenformat, Übergaberegister etc.) eingehalten. Dies hat den großen Vorteil, daß dann auf eine hohe Anzahl verfügbarer Programme zurück gegriffen werden kann.

Beispiel 5.5: Das Betriebssystem CP/M für die Mikroprozessoren 8080, 8085, Z80. CP/M steht für "control programm for microprozessors" und wurde von der Firma Digital Research, USA, entwickelt.

Das Betriebssystem belegt z.B. den in Abb.5.6 dargestellten Speicherraum.

Die immer gleiche Grundsoftware wird BDOS (basic disk oparating system) genannt und stellt das allgemeine Ein- und Ausgabe-Verwaltungssystem dar. Dieses Programm enthält somit die Routinen für den Datenverkehr mit dem Bildschirm, der Tastatur, dem Drucker und dem Diskettenspeicher, sowie die Dateiverwaltung und das Kommandoprogramm. Wobei das Kommandoprogramm die Kommandos entgegen nimmt und eigentlich einen weiteren selbständigen Programmteil darstellt, welcher im Speicher unmittelbar unter dem BDOS-Teil eingeladen wird (CCP: "console command processor"). Die für das Betriebssystem reservierten Speicherplätze von 0000 bis 00FF (256 Byte, Abb.5.6), beinhalten auf den Adressen 5, 6 und 7 immer einen Sprung (JMP DC00) in das Unterprogrammsystem von BDOS. Durch den Aufruf CALL 0005 und eine UP-Auswahlnummer im Register C stehen deshalb jedem Programm diese UPs zur Verfügung.

```
Programmaufruf: (C)= Kennzahl des gewünschten UPs
                (E) oder (DE)= Übergabeparameter
Rückgabe:       (A)= 8bit-Werte, (HL)= 16bit-Werte
```

Zur Anpassung an die speziellen Hardware-Bedingungen eines Systemes muß ein spezielles Anpassungsprogramm an das BDOS angefügt werden (Abb.5.6). Dieses systemspezifische Programm wird meistens BIOS (basic input/output system) genannt und bedeutet so viel wie "Ein- und Ausgabe-Grundsystem". Es dient der Bereitstellung der Schnittstellen zum BDOS, führt die Ein- und Ausgaben durch und liefert den Pufferspeicher für das BDOS.

Auch diese Unterprogramme sind über eine Sprungvermittlung allgemein benutzbar. Das BIOS beginnt immer mit einer ein für allemal festgelegten Folge von Sprungbefehlen zu den einzelnen Programmen:

```
FA00  JMP  KALT  ; Kaltstart-Initialisierung
FA03  JMP  WARM  ; Warmstart
FA06  JMP  TASAB ; Tastaturabfrage
FA09  JMP  TASDR ; Warten auf Tastendruck
FA0C  ....
   :
```

Je nach Ausbau des Systemes, kann die hier gewählte Anfangsadresse FA00 auch eine andere sein, deshalb steht auf den Speicherplätzen 0, 1 und 2 ein Sprungbefehl zum Warmstartsprung (JMP FA03). Dies ist immer so und gestattet dem Benutzer, die Einsprungadressen aller anderen Programme in Erfahrung zu bringen und zu nutzen (Üb.Aufg.5.6).

```
Programmaufruf:
(C)= 8bit-Werte, (BC)= 16bit-Werte, (DE)= Zweiter 16bit-Wert.

Datenrückgabe:
(A)= 8bit-Werte, (HL)= 16bit-Werte.
```

Der Sprungvermittler 0 bis FF enthält immer, neben Datenpuffern und Interruptanlaufstellen, die folgenden Speicherzuordnungen:

```
0000  JMP  0FA03 ; Warmstart im BIOS
0003  DS   1     ; Schalterbyte (Beisp:5.1)
0004  DS   1     ; Bezugslaufwerk und Benutzernummer (00)
0005  JMP  0DC00 ; Sprung zum BDOS-Anfang
0008  DS   40    ; Reserviert für die Interrupts RST1-5.
   :

0080  DS   128   ; Datenpuffer
0100  ...        ; Beginn der Anwenderprogramme
```

Für das in Beisp.5.5 beschriebene Betriebssystem CP/M gibt es sehr viele fertige Programme für alle Bereiche, sowohl wissenschaftliche, als auch technische und kommerzielle. Auch sind Interpreter und Compiler für alle gängigen Programmiersprachen verfügbar.
Eine Systemdiskette enthält auf den ersten Spuren das Betriebssystem, welches durch den Urlader in den Speicher geschrieben wird (Abb.5.6). Daran anschließend können die unterschiedlichsten Daten und Programme als sog. Dateien auf der Diskette Platz finden. Die Programme werden dann über ihren Namen (Dateiname) aufgerufen, d.h. in den Speicherbereich ab 0100H geschrieben, und gestartet.
Insbesondere enthält eine Systemdiskette eine Reihe wichtiger Hilfsprogramme (utility programs), diese sind den sog. nicht residenten (transienten) Kommandos des Betriebssystemes zugeordnet. Zur Ausführung dieser Kommandos wird also erst das entsprechende Hilfsprogramm geladen.

Beispiel 5.6: Einige Hilfsprogramme des Betriebssystemes CP/M

```
SYSGEN : Erzeugung einer Copie von der Systemdiskette
PIP    : Übertragung von Dateiinhalten
ED     : Texteditor
STAT   : Statusanzeigen
ASM    : Assembler
LOAD   : Aus einer Quellprogrammdatei ein nicht residentes Kommando machen.
DUMP   : Dateiinhalt auf den Bildschirm bringen.
```

5.3 Die Assemblersprache

Eine Assembler-Programmiersprache bedient sich der mnemonischen Abkürzungen wie sie bereits in Kap.3 beschrieben wurden. Darüber hinaus bietet sie eine Reihe weiterer Erleichterungen beim Programmieren. Insbesondere können symbolische Marken und Namen benutzt werden. Dies hat zur Folge, daß beispielsweise bei einem Sprungbefehl als Zieladresse ein Name angegeben wird, der an anderer Stelle, nämlich dem Sprungziel, als Marke vor dem entsprechenden Befehl steht.

Beispiel 5.7: Symbolische Adressen

```
            JMP     ANTON
             :  .
ANTON:      MVI     A,0
```

Auf diese Weise ist es nicht erforderlich Sprungadressen zu ändern, wenn Befehle eingeschoben werden.
Ein Assembler ist ein spezielles Programm, das ein, in der entsprechenden Assemblersprache geschriebenes Quellprogramm, in den zugehörigen Objektcode übersetzt.
Es ist zu beachten, daß im allgemeinen jeder Mikroprozessortyp eine andere Assemblersprache und einen anderen Objektcode benutzt. Die hier

verwendeten Beispiele beziehen sich auf den MP-8080/85. Auch werden hier nur die wichtigsten Elemente der 8080/85 Assembler-Programmiersprache behandelt. Detailiertere Angaben sollte man der jeweiligen Assembler-Beschreibung entnehmen.

Ein Assembler der in einem Computer läuft, welcher nicht den entsprechenden Mikroprozessor enthält, wird Cross-Assembler genannt: In diesem Fall ist der erzeugte Objektcode, in dem Computer der zum assemblieren benutzt wurde, nicht ablauffähig.

Zum Rückübersetzen eines Objektcode-Programmes in die Assemblersprache wird ein sog. Disassembler benutzt.

Aufbau einer Befehlszeile

Damit der Assembler eine Befehlszeile richtig übersetzen kann, müssen bestimmte Vereinbahrungen oder Festlegungen getroffen werden:
Eine Befehlszeile besteht maximal aus 4 Feldern. Hiervon sind uns aus dem Vorangegangenen die letzten drei Felder bereits bekannt. Es kommt lediglich ganz links noch ein Feld für Marken oder Namen hinzu.

Beispiel 5.8 Aufbau von Befehlszeilen und Operanden-Strukturen

```
Marke:   Operationscode   Operand
 oder         bzw.         oder          ;Kommentar
 Name     Mnemoniccode   Operanden
-------------------------------------------------------
          LXI           SP,1100H        ;Hexadezimaldaten
ANFA:     MVI           B,55            ;Dezimaldaten
          ANI           10111001B       ;Binärdaten
          LXI           H,PUFFER        ;Datenmarke --> (HL)
          CPI           '*'             ;Vergleich den Akku-Inhalt
                                        ;mit dem ASCII-Code von *
          JMP           ANFA            ;Sprung zur Befehlsmarke
```

Den Aufbau von Befehlszeilen zeigt Beisp.5.8. Insbesondere das Operandenfeld enthält die häufigsten Formen der Festlegung von Daten oder Adressen.

Wichtig ist, daß die unterschiedlichen Angaben in einer Befehlszeile durch sog. Begrenzer getrennt werden.

Beispiel 5.9: Die Begrenzer

```
Leerzeichen (blank):  Feldtrenner (JMP ANFA)
Komma           ,   : Trennt Operanden (MOV A,B)
Apostroph       ' ' : Begrenzt ASCII-Zeichen (CPI 'A').
Semikolon       ;   : Beginn des Kommentarfeldes
Doppelpunkt     :   : Begrenzt Marken (ANFA:XCHG)
```

Beim Eintragen einer Hexadezimalzahl ins Operandenfeld ist darauf zu achten, daß sie nicht mit einem Buchstaben beginnt. Dies läßt sich durch das Einfügen einer führenden Null verhindern. Statt LXI H,A000H wird LXI H,0A000H geschrieben.

Eine Angabe im Operandenfeld wird auch Ausdruck genannt. Neben den in Beisp.5.8 dargestellten Formen solcher Ausdrücke können hier auch sog. Operatoren Anwendung finden. Zu diesen Operatoren zählen z.B. die arithmetischen und logischen Operatoren.

Beispiel 5.10: Operatoren im Operandenfeld

```
Adr.  Objekt   Marke     Oper.  Operanden
      Code               Code
-------------------------------------------
4000  3E0A               MVI    A,5*2
4002  210840             LXI    H,PUFFER+3
4005  00       PUFFER:   NOP
```

Assembleranweisungen

Assembleranweisungen dienen zur Steuerung des Assemblers beim Generieren des Objektcodes (Beisp.5.11).

Beispiel 5.11: Assembleranweisungen, (...)bedeutet wahlweise.

```
Marke    Op.-Code  Operand     ;          Kommentar
-------------------------------------------------------------------------
(marke:) ORG       ausdruck    ;Zuweisung einer Adresse für das erste Byte
         ORG       1000H       ;des folgenden Programmes. Z.B. 1000H.

(marke:) END       (ausdruck)  ;Kennzeichnung des Endes des Quellprogramms
         END                   ;= letzte Anweisung.

name     EQU       ausdruck    ;Es wird dem  Namen im  Markenfeld der Wert
PUFFER   EQU       25A0H       ;des Ausdruckes zugewiesen. PUFFER =  25A0H

name     SET       ausdruck    ;Wirkung wie EQU , kann jedoch mit gleichem
PEGEL    SET       5           ;Namen mehrfach auftreten. Z.B. PEGEL =  05

(marke:) DB        liste       ;Es werden die durch die Liste festgelegten
         DB        18H         ;Daten in unmittelbar  aufeinanderfolgenden
Text:    DB        'Fehler'    ;Speicherstellen, beginnend beim momentanen
                               ;Stand des Adreßpegels abgespeichert.

(marke:) DS        ausdruck    ;Es werden vom momentanen  Stand des Adreß-
         DS        35          ;pegels an z.B.35 Speicherplätze reserviert

(marke:) DW        ausdruck-   ;Es wird jeder 16bit-Wert aus der Ausdruck-
                   liste       ;liste als  Adresse gespeichert,  beginnend
ADRE1:   DW        58B1H       ;beim aktuellen  Stand des Adreßpegels.
```

ORG-Anweisung:

Wenn der Assembler beispielsweise die symbolische Adresse eines Sprungbefehles in eine absolute umwandeln soll, muß ihm bekannt sein, auf welcher Adresse das erste Byte des Programmes später stehen soll, wenn es gestartet wird. Nur so kann er sich die absolute Sprungadresse des Befehles ausrechnen, vor dem die Marke steht, zu der gesprungen werden soll. Zur Festlegung der Adresse des ersten Programmbytes wird die ORG-Anweisung (origin) benutzt. Wenn keine ORG-Anweisung gegeben wird, beginnt der Assembler mit der Adresse 0000.

END-Anweisung:
Die END-Anweisung darf in keinem Programm fehlen. Sie steht immer in der letzten Zeile eines Quellprogrammes. Wenn der wahlweise Ausdruck vorhanden ist, wird sein Wert als Startadresse zur Programmausführung benutzt.

EQU-Anweisung:
Durch die EQU-Anweisung (equate) kann einem Namen ein Wert zugewiesen werden. In Beisp.5.11 erhält der Name 'PUFFER' den Wert 25A0 zugewiesen. Hierbei ist zu beachten, daß hinter dem Namen als Trennzeichen kein Doppelpunkt sondern ein Leerzeichen folgt.

Die einmal getroffene Zuordnung eines Namens zu einem Wert, hat für das gesamte Quellprogramm Gültigkeit. Wenn z.B. irgendwo der Befehl CALL PUFFER steht, übersetzt der Assembler dies in CDA025.

SET-Anweisung:
Demgegenüber kann eine SET-Anweisung in einem Programm häufiger mit gleichem Namen und unterschiedlichem Ausdruck verwandt werden.

Beispiel 5.12: Die SET-Anweisung

```
 Adr. Ob-Code Name       Operand
-------------------------------
              WERT  SET  5
1150  FE05          CPI  WERT
              WERT  SET  10H
1152  FE10          CPI  WERT
```

DB-Anweisung:
Die DB-Anweisung (Datenbyte) dient zur Eingabe von Daten in einen Speicherbereich. In der Liste können Zahlen oder Zeichenfolgen stehen. Bis zu 8, durch Kommata getrennte Eintragungen sind zulässig. Die Ausdrücke müssen, bei der Ausrechnung durch den Assembler, 1Byte-Zahlen im Bereich von - 256 bis 255 ergeben. Zeichenfolgen dürfen höchstens 64 ASCII-Zeichen umfassen und müssen in Apostrophen eingeschlossen sein.

Beispiel 5.13: Die DB-Anweisung

```
Adr. Assemblierter Marke Operations Operanden
       Code              Code
-------------------------------------------
1A00   42697474   TEXT:   DB      'Bitte Adresse'
1A04   65204164
1A08   72657373
1A0C   65
1A0D   20616E67           DB      'angeben'
1A11   6562656E
1A15   A3         NEU:    DB      0A3H
1A16   FD0A       WORT1:  DB      -03H,10
```

DW-Anweisung:
Die DW-Anweisung (Datenwort) hat eine ähnliche Wirkung wie die DB-Anweisung, allerdings werden hier, wie es bei den Adressen des MP-8080/85 üblich ist, das niederwertige und das höherwertige Byte vertauscht.

Beispiel 5.14: Die DW-Anweisung

```
Adr.   Assem-        Marke      Operand
       blierter Code
-------------------------------------
20B0   3B1C          ADR1:  DW  1C3BH
20B2   0400          ADR2:  DW  4
20B4   0A005002      ADR3:  DW  10,250H
```

DS-Anweisung:
Die DS-Anweisung (Datensegment) dient zur Reservierung eines Speicherbereiches.

Beispiel 5.15: Die DS-Anweisung

```
Adr.  Marke  Op.-Code  Operand
-----------------------------
34A0         NOP
34A1  ARSP:  DS        6
34A7         NOP
```

Symbolische Adressierung

Bei der symbolischen Adressierung wird jeder Adresse im Quellprogramm ein anderer Name zugeordnet. Erst der Assembler weist den Namen dann eine absolute Adresse zu.

Beispiel 5.16: Symbolische Adressierung

```
                     ORG   4000H
4000  3E00           MVI   A,0
4002  CD3A08         CALL  UP1
4005  C20B40         JNZ   ENDE
4008  C30040         JMP   4000
400B  D318     ENDE: OUT   PORT1
               UP1   EQU   83AH
               PORT1 EQU   18H
```

Im Beisp.5.16 wurde für den Sprung JMP eine absolute und für den Sprung JNZ eine symbolische Adressierung gewählt. Durch die ORG-Adresse 4000H wird die absolute Adresse des symbolischen Sprungzieles eindeutig festgelegt und der Assembler kann ihr den Wert 400B zuweisen. Wenn dieses Programm dagegen in einem anderen Adreßbereich laufen soll und deshalb die ORG-Adresse verändert wird, müßte die absolute Sprungadresse vom Programmierer geändert werden, während die symbolische vom Assembler automatisch angepaßt wird. Bei symbolischen Adressen, die aus dem Programmadressbereich hinausführen und deshalb keine entsprechenden Marken vorhanden sind, müssen den Namen über eine EQU-An-

weisung die entsprechenden absoluten Adressen zugewiesen werden. Dies wurde im Beisp.5.16 für die Namen UP1 und PORT1 durchgeführt.

Namen und Marken können im allgemeinen aus bis zu 5 oder 6 Zeichen bestehen, wobei das erste Zeichen jedoch immer ein Buchstabe sein muß.

Der Assembler

Das Assembler-Programm hat die Aufgabe aus einer Befehlszeile des Quellprogrammes den entsprechenden Objektcode zu erzeugen. Hierbei dienen die Assembler-Anweisungen zur Steuerung des Übersetzungsvorganges.

Beispiel 5.17: Vom Assembler erstellte Programmliste (listing)

```
;Befehlsadresse (Adresse des 1.Bytes)
   |  ;Objektcode (Hex-Code)
   |     |        ;Marke oder Name
   |     |           |     ;Operationscode (Mnemonic-Code)
   |     |           |        |     ;Operanden
   |     |           |        |        |           ;Kommentare
0000                 |  ORG   |  4000H |  ;Anfangsadresse festlegen
4000 DB18      ANFA: |  IN    |  PORT1 |  ;Parameter einlesen
4002 0F              |  RRC   |        |  ;Akku rotieren
4003 D20040          |  JNC   |  ANFA  |  ;Sprung zum Anfang, wenn das
                     |        |        |  ;niederwertigste Akku-bit = L
4006 CD0E40          |  CALL  |  UP1   |
4009 D319            |  OUT   |  PORT2 |  ;Ausgabe des Ergebnisses
400B C30040          |  JMP   |  ANFA  |  ;Programm erneut durchlaufen
                ;**** Unterprogramm ****
400E 17         UP1: |  RAL   |        |
400F 47              |  MOV   |  B,A   |
4010 213240          |  LXI   |  H,ANFA+50 ;Erhöhe den Adresspegel von
                     |        |        |  ;ANFA um 50 und schreibe das
                     |        |        |  ;Ergebnis (4032H) ins Reg.paar HL
4013 1100E4          |  LXI   |  D,0E400H |
4016 FE41            |  CPI   |  'A'   |  ;Vergleiche den Akku-Inhalt
                     |        |        |  ;mit dem ASCII-Code von A
4018 CCD0B1          |  CZ    |  0B1D0H |
401B C60D            |  ADI   |  13    |
401D C9              |  RET   |        |
0018            PORT1|  EQU   |  18H   |  ;Port 1 definieren
0019            PORT2|  EQU   |  19H   |
0000                 |  END   |        |
                     |        |        |
                     |  SYMBOL TABLE   |
                     |        |        |
ANFA   4000     PORT1   0018     PORT2   0019     UP1    400E
```

Die Übersetzung geschieht meistens in zwei Schritten. Zunächst wertet der Assembler die Anweisungen, die Operanden und die Marken aus. Hierbei wird insbesondere eine sog. Symbol-Tabelle erstellt, die eine Zuordnung der absoluten Adressen zu den symbolischen Namen enthält (Beisp.5.17).

Im zweiten Durchgang werden dann die mnemonischen Abkürzungen in den entsprechenden Binär-Code umgewandelt. Der hierbei entstehende Objekt-

code (Hex-Code) wird im Anwenderspeicher und evtl. direkt auf einer Diskette, des zur Übersetzung benutzten Computers, abgelegt.

Da das Quellprogramm, infolge der Kommentar-Texte, häufig einen großen Speicherraum belegt, wird es nach dem Editieren in den Massenspeicher gebracht. Beim Assemblieren wird das Quellprogramm dann Zeile für Zeile aus dem Massenspeicher geholt, wobei die Kommentare unberücksichtigt bleiben.

Damit der Assembler weiß, an welche Stelle des Anwenderspeichers er den erzeugten Objektcode ablegen soll, wird im allgemeinen beim Start der Übersetzung eine sog. Offset-Adresse angegeben. Die Adresse im Anwenderspeicher, von der ab der Objektcode abgespeichert wird, ergibt sich dann aus der Summe von ORG- und Offset-Adresse.

Der Assembler erzeugt darüberhinaus eine vollständige Programmliste. Hierbei werden die Befehls- und Anweisungszeilen des Quellprogrammes übernommen und durch die Befehlsadressen und den Objektcode ergänzt (Beisp.5.17).

Fehlersuche

Programmentwicklung bedeutet Problemdefinition, sowie das Schreiben, Testen, Verändern und Ergänzen von Programmen.

Insbesondere, wenn es um die Entwicklung umfangreicher Programme geht, schleichen sich zunächst einmal Fehler ein, die es auszumerzen gilt.

Wenn ein Fehler gefunden worden ist, wird das Quellprogramm entsprechend geändert und neu übersetzt (assembliert). Bei kleineren Fehlern ist es jedoch manchmal auch möglich die Änderung direkt im Objektcode (Hex-Code) vorzunehmen. Hierbei bewährt es sich, wenn eine gewisse Erfahrung im Umgang mit dem Objektcode vorliegt. Insbesondere wenn Fehler, oder kleine Änderungswünsche, am Einsatzort des Mikrocomputers auftreten, kann meistens nur durch Änderungen im Objektcode eine rasche Lösung herbeigeführt werden. Solche Programmänderungen lassen sich beispielsweise mit dem in Abb.3.3 dargestellten, hexadezimal programmierbaren, Mikrocomputer durchführen.

5.4 Übungsaufgaben

5.1 Was geschieht mit dem Objektcode, den ein Cross-Assembler erzeugt hat?
5.2 Was ist ein Cursor?
5.3 Wie wird ein Kommando erteilt?
5.4 Welche Aufgaben hat eir Monitor?

5.5 Das Schalterbyte, zur Lenkung der Datenströme, stehe auf der Adresse 2803H.
Schreiben Sie ein Programm, welches - entgegen der Darstellung in Abb.5.4 - den für den Drucker bestimmten Datenstrom zum Bildschirm lenkt.

5.6 Schreiben Sie ein Unterprogramm "Warten auf Tastendruck"; welches auf jedem Mikrocomputer läuft, der über das Betriebssystem CP/M verfügt (s.a. Beisp.5.5).

5.7 Welcher Begrenzer trennt in der Assemblersprache die Operanden?

5.8 Übersetzen Sie den Befehl MVI A,5*2 in den Hex-Code.

5.9
```
        ORG 1000H
        LXI H,PUFER+32
PUFER:  DB  'ABC'
```
Übersetzen Sie den ersten Befehl.

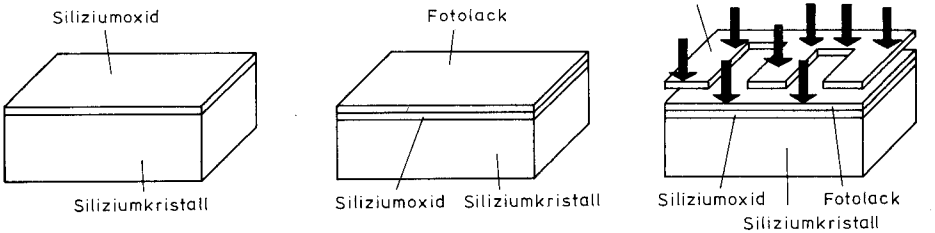
5.10 Übersetzen Sie den 2. Befehl aus Aufg.5.9.

B) Elektronischer Aufbau

Im Mittelpunkt dieses Teils steht die Beschreibung der Hardware von Mikrocomputern und wichtiger peripherer Einheiten. Doch zunächst erfolgt ein Einblick in die technologischen Grundlagen.

6. Integrierte Schaltungen

6.1 Planartechnik



a) Oxidation der Siliziumscheibe bei 1200 °C in feuchter Atmosphäre

b) Aufbringen eines lichtempfindlichen Lackes auf die Oxidschicht

c) Belichten des Lackes durch eine Maske, die die Struktur des Transistors aufweist

d) Entwickeln des Fotolacks und belichteten Teil chemisch entfernen

e) Wegätzen der fotolackfreien Stellen des Oxids

f) Entfernen des restlichen Fotolacks. Es verbleibt eine Oxidmaske für die nachfolgende Diffusion

g) Diffusion der Dotieratome (z. B. Bor) – an den oxidfreien Stellen – in den Kristall und Erzeugung einer P-leitenden Zone

h) Aufdampfen von Aluminium zur Metallisierung der Leitbahnen und Kontaktstellen

i) Entfernen des nicht benötigten Aluminiums mit Hilfe der Fototechnik

Abb. 6.1: Herstellungsschritte zum Aufbau eines Einzelbauelementes in der Silizium-Planartechnik. Durch Oxidmaskierung werden bestimmte kleine Bereiche des Halbleiterkristalls zu N- oder P-leitende Zonen dotiert.

Fünfzehn Jahre nach der Entwicklung des ersten Transistors - im Jahre 1947 - gelang es , mehrere Bauelemente auf einem Silizium-Plättchen (Chip) zu integrieren. Hierbei wurde die sog. Planartechnik angewandt. Diese neu entwickelte Technik gestattet es, Transistoren, Dioden, Kondensatoren und Widerstände nebeneinander auf einer Silizium-Scheibe zu erzeugen und durch Aluminium-Leiterbahnen untereinander zu einer Schaltung zu verbinden. Mit der Planartechnik ist es heute möglich, bis zu 100 000 Bauelemente auf einem Chip unterzubringen. Je nach der Menge der auf einem Chip integrierten Bauelemente, unterscheidet man vier Gruppen von integrierten Schaltungen:

SSI = Kleiner Integrationsgrad (small scale integration)
MSI = Mittlerer Integrationsgrad (medium scale integration)
LSI = Großer Integrationsgrad (large scale integration)
VLSI= Sehr großer Integrationsgrad (very large scale integration)

Den Aufbau der wichtigsten Bauelemente zeigt Abb.6.2.

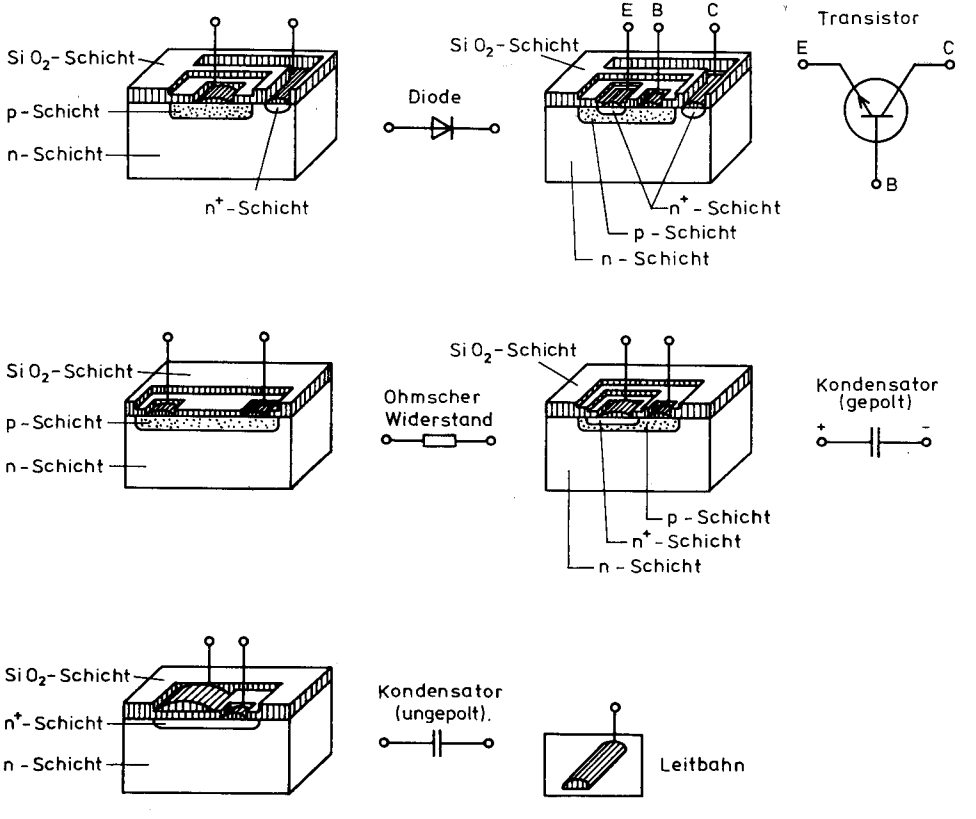


Abb.6.2: Aufbau von Bauelementen in der Planartechnik

Das Dielektrikum eines Kondensators kann beispielweise durch eine Siliziumoxidschicht gebildet werden, während die Kapazität einer Sperrschicht zwischen p- und n-Silizium einen gepolten Kondensator ergibt.
Bei der Herstellung integrierter Schaltkreise wird als Ausgangsmaterial höchstreines Silizium, in Form dünner Scheiben mit ca. 7,5cm Durchmesser (wafer), benutzt. Auf einem solchen Wafer werden dann nebeneinander so viele der zu erstellenden Schaltkreise erzeugt, wie auf ihm Platz haben (z.B. 100 - 150 Stück). Nach den zur Herstellung in der Planartechnik erforderlichen - bis zu sechs - Produktionsschritten, werden die einzelnen Chips - durch Anritzen des Wafers mit einem Diamanten - herausgebrochen und dann in ein Gehäuse (Abb.6.3) gebracht.

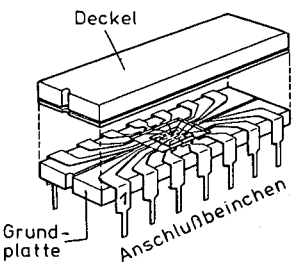


Abb. 6.3: Das aus dem Wafer herausgebrochene Si-Plättchen wird auf eine Grundplatte geklebt, mit Golddrähten an die Beinchen angeschlossen und dann mit einem Deckel versehen.

Zwischendurch und nach der Einkapselung, erfolgt eine Funktionskontrolle der Schaltkreise. Am Anfang der Fertigung eines neuen Schaltkreises ist die Ausbeute an funktionsfähigen Bausteinen oft noch gering (z.B. 10%) und steigt dann durch Verbesserungen im Herstellungsprozess langsam an.

6.2 Schaltkreisfamilien

Mit Hilfe der Planartechnik lassen sich sowohl bipolare, als auch unipolare Transistoren aufbauen.
Zu den bipolaren gehören die npn- und pnp-Transistoren. Sie heißen bipolar, weil sowohl Elektronen als auch Löcher am Ladungstransport beteiligt sind. Demgegenüber wird bei einem Feldeffekttransistor (FET) der Strom von nur einer Ladungsträgerart gebildet, man nennt sie deshalb auch unipolare Transistoren.

Bipolar-Technologie

Beispiele für die Herstellung von Bauelementen in bipolarer Technik wurden bereits in Abb.6.2 dargestellt. Die Bipolar-Technologie kommt vorallem bei solchen analogen und digitalen Schaltkreisen zur Anwendung, bei denen es auf eine hohe Schaltgeschwindigkeit ankommt.

Die am weitesten verbreitete bipolare Schaltkreisfamilie ist die TTL-Familie (Transistor- Transistor-Logik). Als Versorgungsspannung wird +5V±5% benötigt.Die logischen Pegel sind definiert mit L kleiner 0,8V und H ab ca. 2V.

Die TTL-Familie bietet heute das umfassendste und preiswerteste Standardprogramm digitaler Bausteine.

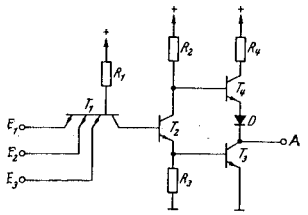


Abb. 6.4: Typischer Aufbau eines TTL-NAND-Gatters innerhalb einer integrierten Schaltung (7410)

Weiterentwicklungen führten dann zu Bausteinen mit speziellen Eigenschaften, wie z.B. niedrige Leistungsaufnahme oder hohe Schaltgeschwindigkeit. Einige dieser TTL-Unterfamilien werden in Abb.6.5 miteinander verglichen.

Desweiteren wurden und werden noch heute andere bipolare Bausteinfamilien entwickelt.

Baustein-Familie	TTL	-L	-S	-LS	ECL	NMOS	CMOS
Leistungsaufnahme je Gatter typ./mW	50	1	50	2	25	1	0,001-6
Maximale Frequenz f/MHz	20-50	2-5	80-125	50	125-....	10	5

Abb. 6.5: Vergleich verschiedener Baustein-Familien.
L= Low-power, S= Schottky-Technik, LS= Low-power-Schottky, ECL= Emitter-Coupled-Logic, NMOS= n-Kanal-MOS-Technik, CMOS= Komplemenentäre MOS-Technik.

MOS-Technologie

Ein großer Vorteil der Feldeffekttransistoren (FET) besteht darin, daß der Strom, welcher einen solchen unipolaren Transistor durchfließt, von einem elektrischen Feld gesteuert wird und nicht durch einen Steuerstrom wie beim bipolaren Transistor. Hierdurch lassen sich Eingangswiderstände von über 10MOhm erreichen.

Eine weitere Verringerung der Eingangsströme wird durch die Anwendung der MOS-Technologie erreicht (MOS = Metall, Oxid, Silizium). Hierbei sind Eingangswiderstände bis zu 10^{15}Ohm erreichbar.

Die MOS-Technologie findet vorallem Anwendung bei der Herstellung großintegrierter digitaler Schaltungen; so auch zur Herstellung von Mikroprozessoren.

In Abb.6.6 ist der Aufbau eines sog. selbstsperrenden n-Kanal MOS-FET-Transistors dargestellt.

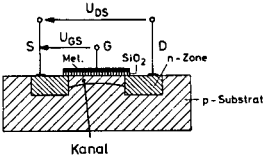


Abb. 6.6: Aufbau eines selbstsperrenden n-Kanal MOS-FET (Enhancement Typ)

In einem p-leitenden Silizium-Substrat werden zwei n-Zonen erzeugt. Dies geschieht durch Diffusion bei hohen Temperaturen oder aber ,zur Erhöhung der Packungsdichte, durch Ionenimplantation; hierbei werden die Dotierungsatome gezielt in den Kristall hineingeschossen.

Die eine der n-Zonen wird Quelle (S = source) und die andere Senke (D = drain) genannt.

Der Kanal zwischen den beiden n-Zonen ist normalerweise nichtleitend (selbstsperrend). Über diesem Kanal ist eine metallische Schicht (G = Gate) angebracht, die durch eine Siliziumoxidschicht (Quarz) vom Substrat getrennt ist. Wenn an die Gate-Elektrode eine positive Spannung in Bezug auf S angelegt wird, gelangen Elektronen aus den n-Zonen in die dazwischenliegende p-Zone. Diese ermöglichen einen Stromfluß zwischen S und D, falls eine Spannung U_{DS} anliegt. Die Steuerspannung U_{GS} erzeugt also einen n-leitenden Kanal zwischen S und D.

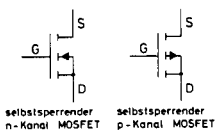


Abb. 6.7: Schaltsymbole für selbstsperrende MOS-FET

Wenn die p- und n-Dotierung umgetauscht wird, erhält man einen selbstsperrenden p-Kanal MOS-FET-Transistor.

Ein selbstleitender MOS-FET-Transistor (Depletion Typ) besitzt einen Kanal mit der gleichen Dotierung wie S und D, wodurch, bei fehlender Steuerspannung U_{GS}, eine leitende Verbindung zwischen S und D besteht.

Die meisten Mikroprozessoren werden heute in der n-Kanal-MOS-Technik (NMOS) gefertigt. Diese Technologie erlaubt eine große Packungsdichte. Die Weiterentwicklung zielt darauf ab, die einzelnen Strukturen immer kleiner werden zu lassen (0,001mm) und immer mehr einzelne Bauelemente

auf einem Chip zu integrieren (100 000 Stück). Die Größe eines solchen Chips beträgt etwa $6x7mm^2$.

Komplementäre MOS-Logig (CMOS)

Die CMOS-Technik kombiniert zwei komplementäre MOS-FET-Transistoren. Ein n-Kanal- und ein p-Kanal-Transistor werden nebeneinander auf dem

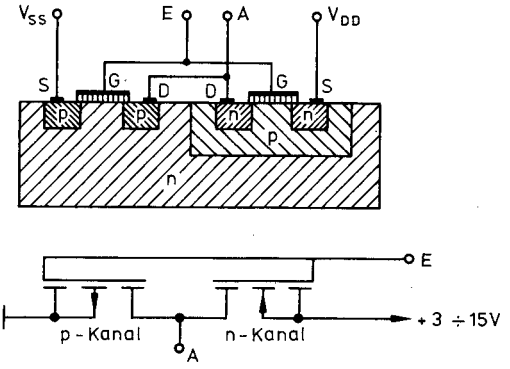


Abb.6.8: Aufbau und Schaltsymbol eines CMOS - Inverters.
E = Eingangssignal
A = Ausgangssignal

selben Substrat erzeugt, wobei der n-Kanal-Transistor durch eine p-leitende Wanne gegen das n-Substrat isoliert wird.

Da in beiden logischen Zuständen - L oder H - jeweils ein MOS-Transistor gesperrt und der andere leitend ist, fällt nur eine sehr geringe Ruheverlustleistung von ca. 10^{-8}W an. Weitere Vorteile dieser Logig-Familie sind in dem großen Versorgungsspannungsbereich und einer hohen Störsicherheit zu sehen. Wie in Abb.6.5 bereits gezeigt wurde, nimmt jedoch die Verlustleistung mit steigender Schaltfrequenz zu und ist bei hohen Frequenzen (ab. ca. 2MHz) größer als bei den Low-power-Schottky-TTL Bausteinen.

Nachteilig ist die Größe einer Zelle, wodurch sich nur geringere Integrationsdichten realisieren lassen.

Auch Mikroprozessoren werden in CMOS-Technik hergestellt, so beispielweise der 1800 von RCA. Der Mikroprozessor 8085 wird ebenfalls in naher Zukunft auch als CMOS-Typ verfügbar sein.

Neben den schon genannten Vorteilen der CMOS-Bausteine kommt noch hinzu, daß dieser CMOS 8085 mit einer beliebigen Taktfrequenz, unterhalb einer oberen Grenzfrequenz, betrieben werden kann.

Handhabung von MOS-Bausteinen

Bedingt durch die extrem hohe Eingangsimpedanz aller Arten von MOS-Bausteinen, sind besondere Vorkehrungen für den Umgang mit ihnen erforderlich.

Bei einem Eingangswiderstand von 10^{14}Ohm kann schnell die Durchbruchsspannungsgrenze des Gate-Oxids (ca. 100V) überschritten und damit der Baustein zerstört oder nachhaltig verändert werden. Solche Spannungen und noch viel höhere, entstehen leicht durch statische Aufladungen (z.B. Nylonhemd: 2kV).

Deshalb sollten stets folgende Vorsichtsmaßnahmen beachtet werden:

- Bausteine nur in leitenden Behältern oder auf leitendem Material aufbewahren.
- Boden- und Tischbelag am Arbeitsplatz sollten keine Reibungselektrizität hervorrufen.
- Signale dürfen die Bausteine nur erreichen, wenn die Versorgungsspannung anliegt.
- Alle nicht benutzten Eingänge sollten entweder an V_{SS} oder V_{DD} angeschlossen werden.

Ausfallrate

Für die Ausfallrate bei integrierten Schaltungen wird im allgemeinen die sog. 'Badewannenkurve' angegeben.

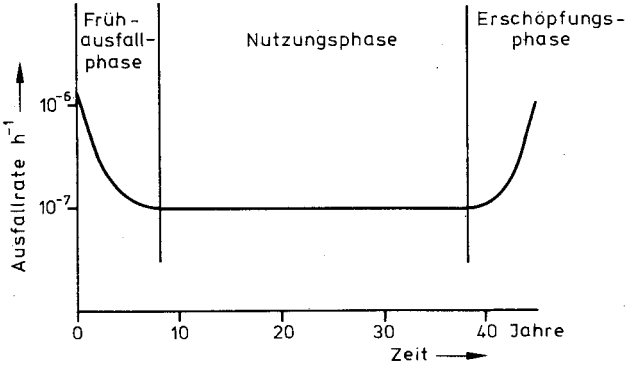


Abb.6.9: Theoretische Ausfallrate als Funktion der Zeit, für integrierte Bausteine

Diese Kurve (Abb.6.9) ist gekennzeichnet durch eine Frühausfallphase. In diesem Bereich 'sterben' vorallem Bausteine, die schon von der Produktion her nicht erkannte Mängel hatten. Wenn eine hohe Zuverlässigkeit gefordert wird, versucht man häufig die Bausteine durch ein sog. Burnin (Einbrennen) künstlich vorzualtern. Die Bausteine, oder auch ganze Schaltungen, werden dann 48h und mehr einer stark erhöhten Temperatur ausgesetzt. Man hofft, auf diese Weise den ersten Teil der Kurve in Abb.6.9 schneller durchlaufen zu können.

An diese erste Phase schließt die sog. Nutzungsphase an, welche durch eine geringere und konstante Ausfallrate gekennzeichnet ist. Später geht dann diese Phase in die Erschöpfungsphase über.

Praktische Erfahrungen zeigen, daß der Abfall in der Frühausfallphase wahrscheinlich noch schneller, als in Abb.6.9 dargestellt, vonstatten geht.

Weitere Erfahrungen liegen jedoch noch nicht vor, da hierzu die bisherige Einsatzdauer für integrierte Schaltungen zu kurz ist.

6.3 Anwendungshinweise

Beim Einsatz integrierter Schaltungen sind einige Besonderheiten zu beachten.

Logische Funktionen

In der Digitaltechnik interessieren im allgemeinen nicht die Spannungen, welche an Ein- und Ausgängen eines Bausteines anliegen, solange sie sich innerhalb bestimmter Grenzwerte bewegen. Hier sind nur die, den Spannungsbereichen zugeordneten, logischen Werte L oder H wichtig. Wenn die Zuordnung so ausfällt, daß dem L der niedrige und dem H der hohe Spannungsbereich zugeordnet wird, nennt man dies "positive Logig". Der umgekehrte Fall heißt dann "negative Logig".

Die logischen Funktionen lassen sich durch entsprechende integrierte Schaltungen realisieren. Solche Schaltungen haben einen oder mehrere

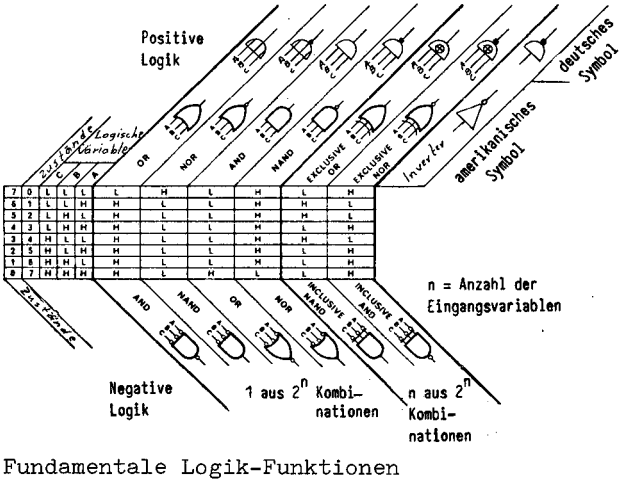


Abb. 6.10: Fundamentale Logik-Funktionen

Eingänge und einen Ausgang. Sie werden meistens als "Gatter" bezeichnet. Auch in Mikrocomputern findet man häufig noch Bausteine die einzelne Gatter enthalten.

Ausgänge

Das in Abb.6.4 dargestellte TTL NAND-Gatter enthält eine Gegentaktendstufe als Ausgang. Dies ist eine der gebräuchlichsten Ausgangsstufen. Daneben ist jedoch auch noch der sog. "offener Kollektor-Ausgang" (open collector) anzutreffen (Abb.6.11). Wie schon der Name sagt, besteht diese Endstufe lediglich aus einem npn-Transistor dessen Emitter an Masse liegt.

Vorteile dieser Schaltung sind, daß sich diese Ausgänge parallel schalten lassen (wired-AND- bzw. wired OR-Verknüpfung) und daß über einen sog. pull-up-Widerstand, der auch an eine höhere Spannung gelegt werden kann, eine Umwandlung, der den logischen Zuständen zugeordneten Spannungen, möglich ist.

Eine solche Pegelwandlung kann beispielweise erforderlich sein, um einen TTL-Ausgang an einen, mit höherer Spannung betriebenen, CMOS-Baustein anzupassen.

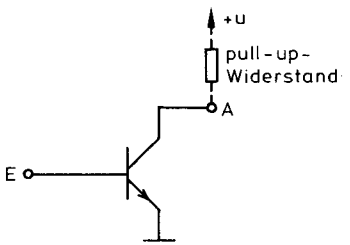


Abb.6.11: Ausgangsschaltung mit offenem Kollektor (open collector)

Insbesondere die Realisierung von Bus-Systemen in Mikrocomputern, bei deren viele Ausgänge parallel geschaltet sind, machte es jedoch erforderlich, einen Ausgang zu realisieren, der sich 'abschalten' läßt. Ein solcher Ausgang kann neben den beiden logischen Zuständen L und H auch noch einen sog. hochohmigen Zustand annehmen.

Während bei einer herkömmlichen Gegentaktendstufe (Abb.6.4) immer nur ein Transistor gesperrt ist, besteht bei der Endstufe in Abb.6.12 auch die Möglichkeit beide Transistoren zu sperren. Eine solche Endstufe wird Tristate- (oder auch three state-) Ausgang genannt.

Die Sperrung der Ausgangstransistoren T_1 und T_2 erfolgt über die beiden Transistoren T_3 und T_4. Wie schon früher erwähnt, wird der Eingang, welcher z.B. die Ausgänge eines Speichers in den inaktiven Zustand bringt, häufig mit CS (chip select = Bausteinauswahl) bezeichnet.

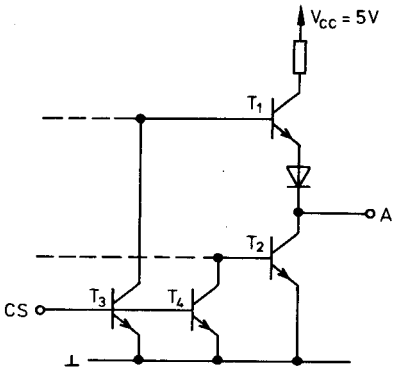


Abb.6.12: Tristate- Ausgang eines Speichers

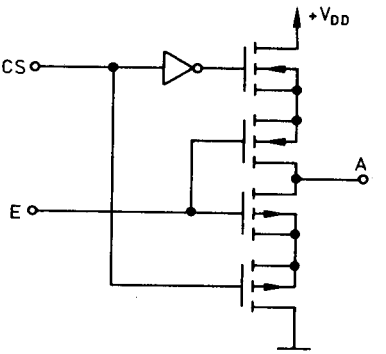


Abb.6.13: Tristate- Ausgang in CMOS- Technik

In Abb.6.13 ist noch ein Tristate-Ausgang in CMOS-Technik dargestellt. Ein solcher Ausgang ist im inaktiven Zustand besonders hochohmig (10 Gigaohm).

Im hochohmigen Zustand nimmt dann der Ausgang den logischen Pegel an, der ihm von außen aufgeprägt wird (indifferenter Zustand, floating).

Häufig sind in einer Schaltung an einem Ausgang mehrere Eingänge anderer Bausteine angeschlossen. Hierbei muß darauf geachtet werden, daß die Ausgangsbelastbarkeit (fan out) nicht überschritten wird. Ein Fan-Out von 10 gibt an, daß an diesen Ausgang 10 Eingänge der gleichen Bausteinfamilie anschließbar sind.

Beispiele:

Bausteinfamilie	TTL	NMOS	CMOS
Typisches Fan-Out	10	20	50

Verschiedenes

Die meisten Digitalbausteine haben das in Abb.6.3 dargestellte DIL-Gehäuse (Dual-In-Line).

Für Sonderanwendungen ist es jedoch auch möglich den einzelnen Chip ohne Gehäuse zu bekommen. Insbesondere zur Herstellung sog. Hybrid-Bausteine wird von dieser Möglichkeit Gebrauch gemacht. Bei der Hybridtechnik werden mehrere Chips direkt auf eine winzige Leiterbahnplatte geklebt und über Golddrähte mit den Leiterbahnen - und damit untereinander - und mit den Anschlußbeinchen verbunden. Anschließend wird das Gehäuse, so wie in Abb.6.3, verschlossen. Anwendung findet diese Technik beispielweise zur Herstellung sehr schneller Analog-Digital-Wandler.

Die Betriebsspannungen der verschiedenen Bausteine differieren infolge der unterschiedlichen Herstellungstechnologien. Hierdurch sind zum

Teil auch andere Spannungsbereiche für die logischen Zustände L und H bedingt.
Es dominiert jedoch die von den TTL-Bausteinen her bekannte Festlegung, dies wird häufig als "TTL-kompatibel" bezeichnet.

Beispiele:

Bausteinfamilie	TTL	ECL	PMOS	NMOS alt	NMOS neu	CMOS
Betriebsspannungen	+5V	-5,2V	+5V,-12V	+12V,±5V	+5V	+3÷15V

Die Impedanz der Stromversorgung sollte stets möglichst niedrig gehalten werden, um eine Kopplung der Bausteine untereinander - über die Versorgungsleitung - zu verhindern. Eine Entkopplung durch Kondensatoren mit guten HF-Eigenschaften und großem Verlustwinkel hat sich bewährt. Hierzu werden meistens keramische Kondensatoren mit Werten zwischen 100 bis 1000nF benutzt.

Es ist besonders wichtig, daß der Kondensator mit möglichst kurzen Leitungen an den Spannungs- und den Masseanschluß eines Bausteines angeschlossen wird.

Bei niedrig integrierten Bausteinen, (z.B. TTL) genügt ein Kondensator für 4 bis 6 Schaltkreise. Höher integrierte Bausteine, wie Mikroprozessoren oder Speicher, benötigen meistens einen höheren Entkopplungsaufwand, wobei dies jedoch auch noch von der internen Struktur des Chips abhängen kann.

7. Der Mikroprozessor

Ein erster Einblick in die Funktionsweise eines Mikroprozessors wurde bereits in Kap.2 gegeben.

Die meisten heute erhältlichen Mikroprozessoren haben einen 8bit-Datenbus und werden deshalb auch 8bit-Mikroprozessoren genannt. Bereits 1971 war der erste 8bit-Mikroprozessor (8008 der Firma Intel) erhältlich. Jedoch erst 1974 erschien der Mikroprozessor 8080 auf dem deutschen Markt. Er verfügt, neben vielen technischen Verbesserungen, über einen erweiterten Befehlssatz des 8008 und diente hiermit als Grundlage für viele spätere Entwicklungen.

8bit Mikroprozessoren

Seit 1971 wurden viele verschiedene Mikroprozessoren entwickelt. Insbesondere Verbesserungen in der Herstellungstechnologie führten zu immer komplexeren Systemen.

Typenbezeichnung	Minimale Befehlszykluszeit/µs	Menge der Grundbefehle	Menge der Allzweckregister	Lieferbar seit	Hersteller
CDP1802	2,5	91	16x16bit	1977	RCA
2650AN	1,5-2,4	75	7	1977	Valvo-Signetics
MD46802	0,8	72	2	1980	Mitel
6500	1-2	56	1	1976	Rockwell
6800	0,5-1	72	2	1975	Motorola
6802	1	72	2	1976	"
6809	1	59	2-3	1980	"
F8-3850	2	67	65	1974	Fairchild
Z80	1-1,6	158	14	1977	Zilog
8008	12,5-20	48	6	1971	Intel
8080	1,5-2	72	7	1974	"
8085	0,8-1,3	74	7	1977	"
8088	0,4	105	8	1979	"
SC/MP	2	46	2	1975	National Semic.
NSC800	1	158	14	1980	"
TMS9980	0,4	69	Register im RAM	1978	Texas Instruments

Abb. 7.1: Gegenüberstellung einiger 8bit-Mikroprozessoren. Die meisten werden in NMOS-Technik angefertigt, lediglich der 8008 wurde in PMOS-Technik und der CDP1802, der MD46802 (kompatibel mit dem 6802) und der NSC800 werden in CMOS-Technik gefertigt. Eine innere 16bit-Struktur besitzen die Mikroprozessoren MC6809, 8088, TMS9980.

Die meisten 8bit-Mikroprozessoren haben einen 16bit-Adreßbus, so daß ein Speicherraum von 64KB adressierbar ist. Bei den neuesten Bausteinen wird intern die 16bit-Struktur der 16bit-Mikroprozessoren benutzt (Abb.7.3).

Leider verfügen fast alle Mikroprozessoren über einen anderen Befehlssatz und eine differierende Anschlußbelegung, so daß sie untereinander nicht kompatibel sind. Auch im folgenden wird deshalb - wie bisher - der Mikroprozessor 8080 bzw. 8085 exemplarisch behandelt.

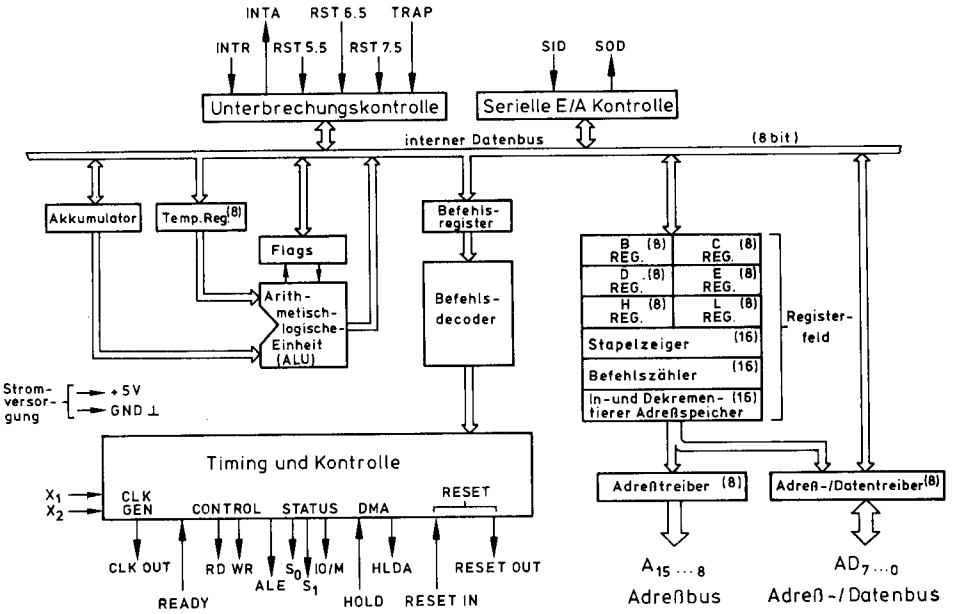


Abb. 7.2: Innerer Aufbau des MP8085. Die Ein- und Ausgangssignale werden im nächsten Kapitel beschrieben.

16bit-Mikroprozessoren

Der Begriff "16bit-Mikroprozessor" ist etwas diffus und eignet sich nicht sehr gut zur Klassifizierung von Mikroprozessoren. Da sich diese Bezeichnung jedoch eingebürgert hat, soll hier kurz darauf eingegangen werden.

Ein 16bit-Mikroprozessor verfügt im allgemeinen über einen 16bit Datenbus und die Möglichkeit intern mit 16bit-Worten zu arbeiten.

Beispiele: 8086 von Intel
MC68000 von Motorola
TMS9900 von Texas Instruments
Z8001 von Zilog

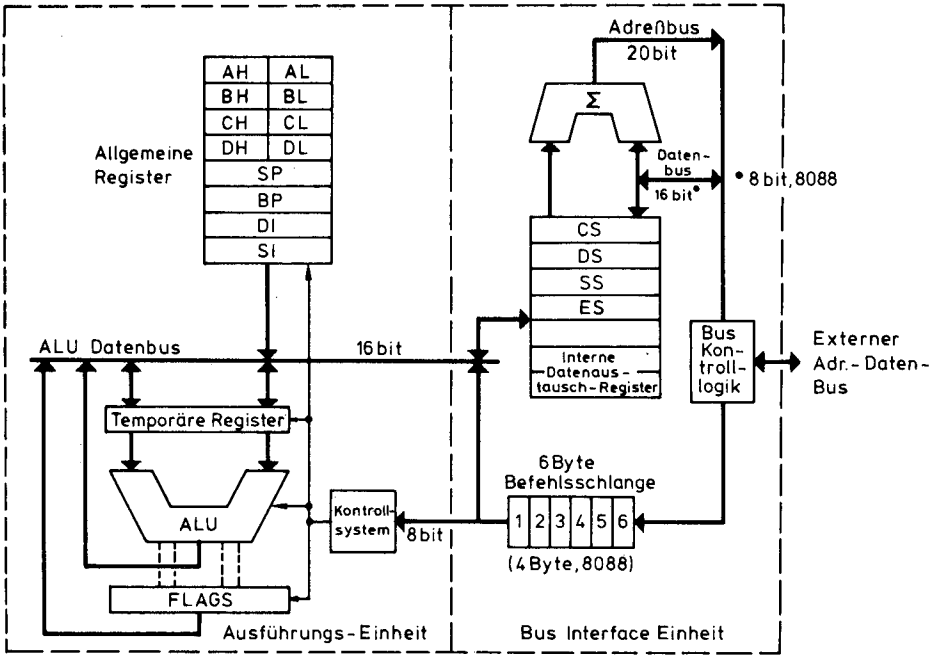


Abb. 7.3: Vereinfachtes Blockdiagramm des 16bit-Mikroprozessors 8086, bzw. des quasi 8bit-Mikroprozessors 8088.

Auch der Adreßbus weist meistens eine größere Breite auf. Hierdurch lassen sich dann Speicherräume bis zu einigen Megabytes adressieren.

7.1 Das Grundsystem des MP8080/85

Der MP8080 benötigt einen zweiphasigen Takt mit einer Frequenz zwischen 0,5 und 2,08MHz. Zur Erzeugung dieses Taktes ist ein zusätzlicher Baustein erforderlich. Der zum Grundsystem gehörende Baustein 8224 (Abb.7.4) übernimmt diese Aufgabe und bereitet gleichzeitig noch einige andere Signale auf. Die Taktfrequenz wird meistens durch einen Quarz festgelegt um eine genaue Zeitbasis zu erhalten, wobei die Quarzfrequenz den neunfachen Wert der gewünschten Taktfrequenz des MP8080 haben muß.

Der Systemsteuerbaustein und Bustreiber 8228 (Abb.7.4) übernimmt vom MP8080 die Zustandsinformationen und erzeugt hieraus Steuersignale. Beide Zusatzbausteine werden in bipolarer Schottky-Technik gefertigt.

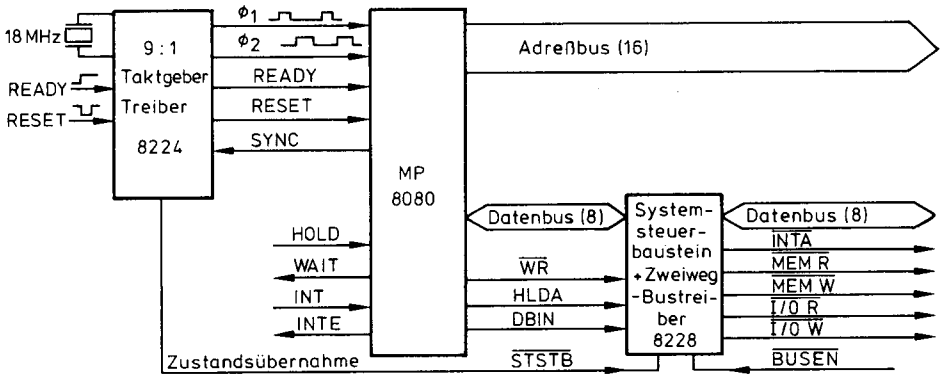


Abb. 7.4: Grundverschaltung des MP 8080

Die in Abb.7.4 angegebenen Signale bedeuten im einzelnen:

READY: H --→ MP arbeitet
L --→ MP bleibt nach der Aussendung einer Adresse stehen und bringt den Ausgang WAIT in den H-Zustand.

RESET: L --→ MP startet bei Adresse 0000 ⌐└⌐

SYNC : Signal zu Beginn eines neuen Maschinenzyklusses (Zustandsübernahme).

HOLD : H --→ MP anhalten, die Busse in den hochohmigen Zustand bringen und auf HLDA ein H ausgeben (Quittierung).

INT : H --→ Unterbrechungsanforderung (interrupt, Kap.7.4)

INTE : H --→ Es werden Unterbrechungen angenommen

WR : L --→ Datenausgabe (write) H --→ Dateneingabe

DBIN : H --→ Datenbus im Eingabezustand

STSTB: L --→ Zustandsübernahme (status strobe)

INTA : L --→ Unterbrechungsquittung

MEMR : L --→ Speicher lesen (memory read)

MEMW : L --→ Speicher schreiben (memory write)

I/OR : L --→ Ein-/Ausgabeport lesen (in/out read)

I/OW : L --→ Ein-/Ausgabeport schreiben (in/out write)

BUSEN: L --→ Baustein 8228 in den hochohmigen Zustand bringen (bus enable)

Wenn die Busse über die Eingänge HOLD und BUSEN in den hochohmigen Zustand gebracht worden sind, können sie von anderen Bausteinen oder Geräten benutzt werden. Diese Technik wird beispielweise angewandt, um auf die an den Bussen angeschlossenen Speicher direkt zugreifen zu können. Man sagt dann, der Mikrocomputer ist DMA fähig (direct memory access).

MP 8085

Während der MP 8080 noch in der alten NMOS-Technik gefertigt wird und deshalb drei Versorgungsspannungen (+12, ±5V) benötigt, erfolgt die Herstellung des MP 8085 in der neueren (depletion-load) NMOS-Technik mit nur +5V Versorgungsspannung.
Die Fortschritte in der Herstellungstechnologie führten zu weiteren Vorteilen des MP 8085 gegenüber dem MP 8080:
Der MP 8085 ist um 50% schneller (3MHz) als der MP 8080,
hat zusätzlich einen seriellen Eingang (SID) und einen seriellen Ausgang (SOD),
besitzt 4 weitere Unterbrechungseingänge (TRAP, RST5.5, RST6.5, RST7.5),
benötigt keinen Taktgeber- und keinen Systemsteuerbaustein,
hat darüberhinaus aber den gleichen Befehlssatz wie der MP 8080 und noch einige Befehle mehr.
Da der MP 8085 auch nur 40 Anschlußbeinchen (pins) wie der MP 8080 hat, können nicht alle Ein-/Ausgabekanäle gleichzeitig nach außen geführt werden. Dies führte zur Anwendung des 'Zeitmultiplex-Verfahrens' in Bezug auf den Datenbus und den niederwertigen Teil des Adreßbusses. Auf 8 Pins des MP 8085 erscheint deshalb kurzzeitig das niederwertige Adreßbyte und dann erst stehen sie für den Datenbus zur Verfügung.

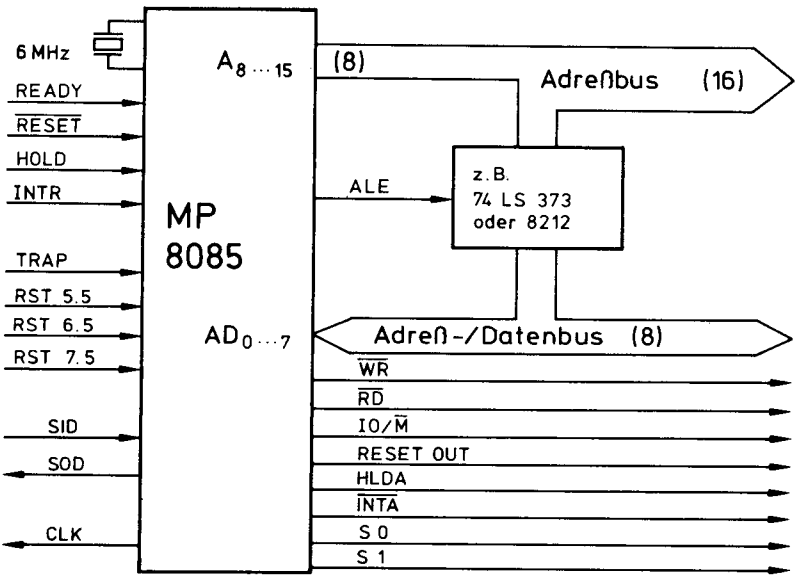


Abb. 7.5: Grundverschaltung des MP8085. Ein Taktgeber- und Systemsteuerbaustein ist nicht erforderlich. Der untere Teil des Adreßbusses wird durch das Demultiplexen des AD-Busses gewonnen.

Durch den Einsatz spezieller Speicher- und Ein-/Ausgabe-Bausteine, die den Adreß-/Datenbus intern demultiplexen, ist es möglich, mit nur drei Bausteinen einen kompletten Mikrocomputer aufzubauen (siehe Anhang).

Meistens werden jedoch an den MP 8085 Standardbausteine - wie auch an den MP 8080 - angeschlossen.

Durch das Demultiplexen des Adreß-Datenbusses (74LS373 in Abb.7.5) erhält man ein Bussystem ähnlich dem des MP 8080. Immer wenn auf dem AD-Bus gerade das niederwertige Adreßbyte erscheint, gibt der ALE-Ausgang (address latch enable) einen H-Impuls ab, der den Baustein 74LS373 dazu veranlaßt dieses Adreßbyte aufzunehmen, zu speichern und in den Adreßbus einfließen zu lassen.

Die 4 Schreib-/Lesesignale für Speicher und Ein-/Ausgabebausteine des MP 8080 Grundsystemes (MEMR...I/OW) sind beim MP 8085 durch 3 Ausgänge ersetzt worden:

RD : L --→ Lesen (read), der Datenbus ist auf Eingabe geschaltet.
WR : L --→ Schreiben (write), der Datenbus ist auf Ausgabe geschaltet.
IO/M : H --→ Ein-/Ausgabe (in/out) Operation. Zugriff zu peripheren Einheiten.
L --→ Speicherzugriff (memory).

Wenn über der, einen Kanal kennzeichnenden Buchstabenkombination, ein Querstrich steht, so bedeutet dies "low aktiv", d.h. ein L-Signal charakterisiert diesen Zustand. Zwei zusätzliche Ausgänge geben Aufschlüsse über den Status (SO und S1) des MP 8085, wobei diese Signale jedoch einen anderen zeitlichen Ablauf als das WR- und RD-Signal haben (Kap.7.2).

	IO/M	WR	RD	S1	SO
Speicher lesen (read)	L	H	L	H	L
Speicher schreiben (write)	L	L	H	L	H
Peripherie Eingabe	H	H	L	H	L
Peripherie Ausgabe	H	L	H	L	H
Operationsteil holen	L	H	L	H	H
Interrupt Quittung	H	H	H	H	H
HLT (Halt) - Befehl	.	.	.	L	L
MP anhalten (HOLD)	.	.	.	X	X
Reset	.	.	.	X	X

Abb. 7.6: Schreib-, Lese- und Statussignale des MP8085. X bedeutet undefiniert und . bedeutet hochohmiger Zustand (tristate).

Die durch den Quarz erzeugte Frequenz (typisch 6MHz) wird im Mikroprozessor halbiert und ergibt damit den internen Takt, der gleichzeitig auf dem Ausgang CLK (clock) verfügbar ist.

7.2 Signalverläufe am MP8085 (Timing)

Die im letzten Kapitel beschriebenen Ein-/Ausgabekanäle des MP 8085 können immer nur einen der beiden Zustände L oder H annehmen, dies jedoch zu den unterschiedlichsten Zeitpunkten. Eine Gegenüberstellung der zeitlichen Verläufe von L- und H-Zuständen mehrerer Kanäle nennt man Timing-Diagramm.

Befehlszyklus

Es dürfte klar sein, daß die Signale auf den 38 Kanälen (40-Stromversorgung) eines Mikroprozessors recht komplex sein können und der Mikrocomputer nur richtig arbeitet, wenn die angeschlossenen Bausteine das Timing des Mikroprozessors berücksichtigen. Deshalb sollen hier einige wichtige und grundlegende zeitliche Abläufe kurz skizziert werden.

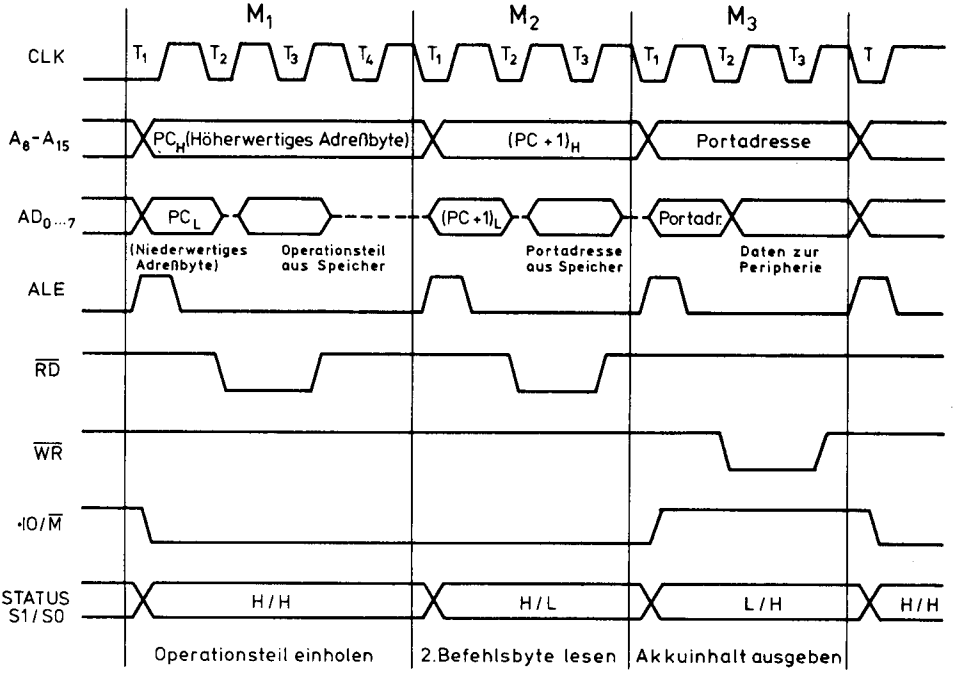


Abb. 7.7: Grundlegendes Timing des MP8085 für den Befehl OUT.

In Abb.7.7 ist als Beispiel der Signalverlauf für einen OUT-Befehl dargestellt. Für den gesamten Ablauf werden 3 Maschinenzyklen (M_1, M_2, M_3) und 10 Takte benötigt. Diese 10 Takte entsprechen dem in der Befehlsliste (Abb.3.4a) angegebenen Wert. Die Dauer einer Taktperiode T errechnet sich aus der Quarzfrequenz f_q wie folgt:

$$T = \frac{2}{f_q} \quad ; \quad \text{z.B. } T = \frac{2}{6\text{MHz}} = 0{,}33 \cdot 10^{-6}\text{s}$$

Ein Maschinenzyklus beginnt immer mit der Aussendung einer Adresse und ist normalerweise 3 Takte lang, nur M1 besteht meistens aus 4 oder 6 Takten. Ein gesamter Befehlszyklus kann aus bis zu 5 Maschinenzyklen und maximal insgesamt 18 Takten (CALL) bestehen. Es sind insgesamt sieben verschiedene Arten von Maschinenzyklen möglich. Welcher Zyklus gerade abläuft läßt sich anhand der Status- und Kontrollsignale (IO/M, S1, SO, RD, WR, INTA, s.a. Abb.7.6) feststellen.

Der erste Maschinenzyklus dient immer zur Einholung des Operationsteiles eines Befehles aus dem Speicher (opcode fetch). Hierzu wird der Inhalt des Befehlszählers (PC = program counter) auf den Adreßbus gebracht; das höherwertige Byte auf den oberen Halbbus (A8...15) und das niederwertige Byte auf den Adreß-Datenbus ($AD_{0...7}$). Das ALE-Signal sorgt dann dafür, daß das niederwertige Byte außerhalb des Mikroprozessors zwischengespeichert wird. Danach erfolgt eine Umschaltung des AD-Busses auf Dateneingabe und die Aussendung eines Lesesignales (RD = L). Da der Ausgang IO/M ebenfalls im L-Zustand ist, sendet der, über die Adresse angesprochene Speicherplatz den Operationscode des OUT-Befehles (D3) an den MP 8085. Die Interpretation des Operationsteiles (während T4) zeigt dem Mikroprozessor, daß noch ein weiteres Befehlsbyte, nämlich die Portadresse, eingeholt werden muß. Dies geschieht unter Erhöhung des Befehlszählers um 1 im zweiten Maschinenzyklus.

Im letzten Maschinenzyklus schaltet dann der Ausgang IO/M auf H (Peripheriezugriff) und der Schreibausgang (WR) auf L und es wird der Inhalt des Akkumulators auf den Datenbus gebracht. Die 1Byte-Portadresse wird dabei als höher- und niederwertiges Adreßbyte ausgegeben.

Wartezustand

Die Arbeitsgeschwindigkeit eines Mikroprozessors ist recht hoch. Aus Abb.7.7 geht beispielweise hervor, daß zum Einlesen eines Befehlsbytes aus dem Speicher nur ungefähr eine Taktperiode (ca. 400ns) zur Verfügung steht. Was ist zu tun, wenn der Speicher langsamer arbeitet, oder wenn ein langsames peripheres Gerät bedient werden soll? Hierzu gibt es die Möglichkeit eine beliebige Anzahl von Warteperioden in den Lese- oder Schreibablauf einzubauen. Das Einfügen von Warteperioden (T_{WAIT}) muß dem Mikroprozessor von außen, über den READY-Eingang, befohlen werden. Immer wenn innerhalb eines Maschinenzykluses die zweite Taktperiode (T_2) läuft, fragt der MP 8085 den READY-Eingang ab. Findet er ein L vor, fügt der MP 8085 soviele Wartezyklen ein, bis der READY-Eingang wieder den Zustand H annimmt. Auf diese Weise kann der Einlese- oder Ausgabe-Zustand um die Zeitdauer n•T (n = 1, 2, 3...) verlängert werden, wobei die Zahl n beliebig groß sein kann. Während der

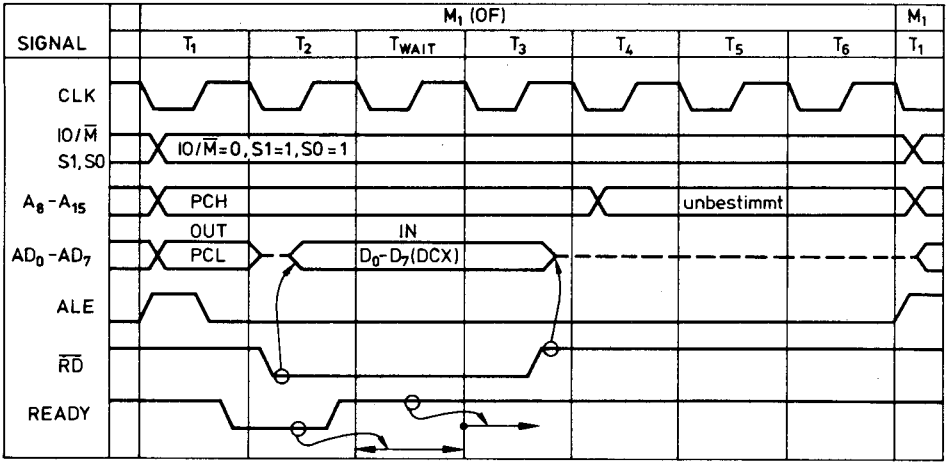


Abb. 7.8: Operationsteil-Zyklus für den Befehl DCX mit einer Warteperiode. PCH= Befehlszähler höherwertig, PCL= Befehlszähler niederwertig, ---= hochohmig, OF= opcode fetch, T_{WAIT}= Warteperiode.

gesamten Wartezeit bleiben alle vom Mikroprozessor ausgesandten Signale im gleichen Zustand.

Nachdem der READY-Eingang wieder den logischen Pegel H angenommen hat führt der MP 8085 den T_3-Takt aus.

In dem in Abb.7.8 dargestellten Beispiel handelt es sich um den Befehlszyklus des 1Byte-Befehles DCX (dekrementiere ein Registerpaar). Nachdem der Befehlsdecoder während T_4 festgestellt hat, um welchen Befehl es sich handelt, erfolgt während T_5 und T_6 die Ausführung des Befehles. Zur Zeit der Befehlsinterpretation und Abarbeitung ($T_{4, 5, 6}$) ist der Zustand des höherwertigen Adreßbytes unbestimmt, da der Adreßzwischenspeicher (Abb.7.2) auch zum In- und Dekrementieren benutzt wird.

7.3 Die Busse

Schon mehrfach wurde auf die Busse eines Mikrocomputers Bezug genommen. Sie stellen gewissermaßen das zentrale Nervensystem eines Computers dar.
Neben dem Adreß- und dem Datenbus werden häufig noch die Steuerleitungen zum sog. Steuerbus zusammengefaßt. Dieser Steuerbus sorgt vorallem dafür, daß die vielen, parallel am Datenbus angeschlossenen, Bausteine diesen immer nur einzeln und in der richtigen Richtung benutzen.

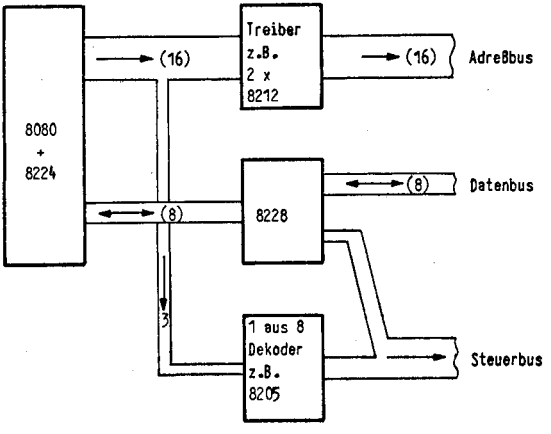


Abb. 7.9: Die drei Busse des MP8080. Eine detaillierte Schaltung enthält Kapitel 12.

In einem, aus dem MP 8080 aufgebauten, Mikrocomputer (Abb.7.9) besteht der Steuerbus im wesentlichen aus den Steuersignalen des Systemsteuerbausteines 8228 (s.a. Abb.7.4) und den Bausteinauswahlsignalen, die beispielweise durch einen '1 aus 8 Dekoder' aus dem Adreßbus erzeugt werden.
Während der Systemsteuerbaustein gleichzeitig für einen hohen Ausgangsstrom sorgt, muß der Adreßbus bei großen Systemen zusätzlich über entsprechende Treiber verstärkt werden. Dies führt uns zur Frage der Belastbarkeit von Bussen.

Belastbarkeit

Jeder Baustein, der an die Busse angeschlossen wird, stellt eine ohmsche und eine kapazitive Last dar. Diese Lasten und die entsprechenden Ausgangsströme der Mikroprozessoren und einiger Treiberbausteine sind in Abb.7.10 dargestellt.
Wenn die Bausteine im hochohmigen Zustand sind, fließt nur noch ein Strom von ca. ± 0,01mA.
Die Eingangsströme zu den Speichern (2114, 2716) und den Ein-/Ausgabebausteinen (8255, 8155) sind im allgemeinen so klein, daß sie kaum ins Gewicht fallen. Der Ausgang des MP 8085 (0,4mA) könnte beispielweise 40 Speicher (0,01mA) treiben. Diese Rechnung berücksichtigt jedoch nur die statischen Ströme.
Von größerer Bedeutung sind jedoch meistens die im dynamischen Fall fließenden Lade- oder Entladeströme, bei einem L/H-oder H/L-Wechsel, die durch die Kapazitäten der Eingänge - auch der gerade nicht benutzten - und der Leitungen hervorgerufen werden.

Baustein	Ströme					Kapazitäten in pF		Impulsverzögerung in ns
	Ausgang in mA			Eingang in uA				
	L-Pegel (kleiner 0;45 V)	H-Pegel		L-Pegel	H-Pegel	Ausg.	Eing.	
		U_{min} in V	I					
2114	9	2,4	-2,5	10	10	5	5	
2708	1,6	3,7	-0,1	10	10	12	6	
2716	2,1	2,4	-0,4	10	10	12	6	
2732	2,1	2,4	-0,4	10	10	12	6	
74LS245	24	3,2	-15	-400	20			8
74LS367	24	3,5	-2,6	-400	20			16
74LS373	24	3,5	-2,6	-400	20			10
8080	1,9	3,7	-0,15	-100	-2000	20	10	
8085	2	2,4	-0,4	10	10			
8155	2	2,4	-0,4	10	10			
8205	10	2,4	-1,5	-250	10		5	18
8212	15	3,6	-1	-250	10	12	9	15
8228	10	2,4	-1	750	20	15	12	30
8255	1,7	2,4	-0,4	10	10	20	10	

Abb. 7.10: Treiber- und Lastkenndaten einiger Mikrocomputer-Bausteine

Beispiel: An den MP 8085 sollen 4 Programmspeicher - (2716, 8KB), 16 Arbeitsspeicher - (2114,8KB) und 2 Ein-/Ausgabe - (8255, 48 Kanäle) Bausteine angeschlossen werden.

Baustein	4x2716	16x2114	2x8255	MP 8085	Leitungen	Summe
Kapazität	24	80	20	ca.20	100	244pF

Die Standard-Lastkapazität beträgt beim MP 8085 C_L = 150pF und wird damit um 94pF überschritten. Dies bedeutet eine Impulsverzögerungszeit von t_v = 94pF·0,30ns/pF = 28,2ns.

Wie das Beispiel zeigt, wird durch die Kapazitäten eine Impulsverzögerung hervorgerufen, wenn sie größer als 150pF (MP 8085) ist. Für den Bereich bis 300pF muß mit 0,30ns/pF gerechnet werden. Bei kleineren Kapazitäten, bis hinunter zu 25pF, wird das Timing dagegen um 0,10ns/pF beschleunigt ablaufen.

In großen Systemen, die auf mehreren Leiterplatten aufgebaut sind, wächst insbesondere die Leitungskapazität stark an und es müssen, zur Vermeidung großer Impulsverzögerungen, Treiber in die Busse geschaltet werden. Im einfachsten Fall reichen Treiber direkt hinter dem Mikroprozessor. Meistens ist es jedoch erforderlich auch auf den einzelnen Leiterplatten Treiber vorzusehen.

Adreßbus

Der Adreßbus (AB) besteht aus 16 Ausgangsleitungen, auf denen eine 16bit-Zahl parallel ausgegeben wird. Hierdurch lassen sich $2^{16} = 65536$

Adreßbus-Kanäle	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
Kanalwertigkeit	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Dezimale Wertigkeit der Kanäle	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Abb. 7.11: Adreßbus-Wertigkeiten

verschiedene Adressen darstellen, es ist also ein Speicherraum von 64KB adressierbar. Über den Adreßbus werden diejenigen Speicherplätze und Ein-/Ausgabe-Ports (mit je 8 Kanälen) angesprochen bzw. ausgewählt, die mit dem Datenbus in Kontakt treten sollen.

Während beim MP 8080 alle 16 Adreßbits für die Dauer eines normalen Maschinenzyklusses anstehen, müssen beim MP 8085 die niederwertigen 8bit zwischengespeichert werden, da sie immer nur zu Beginn eines neuen Maschinenzyklusses während T_1 erscheinen (Abb.7.7). Da für größere Systeme sowieso Bustreiber erforderlich sind, kann diese Zwischenspeicherung vom Treiberbaustein mit übernommen werden. In Abb.7.12 sind zwei Schaltungen dargestellt, die diese Funktion erfüllen können. Der Baustein 8212 ist schon älter und entstammt der 8080er Serie, während der Baustein 74LS373 neu entwickelt wurde. Letzterer hat ein kleineres Gehäuse, eine geringere Impulsverzögerung und einen höheren Ausgangsstrom (Abb.7.10).

Der ALE-Impuls (8085) sorgt für die Zwischenspeicherung des gesamten Adreßbusses. Unbedingt notwendig ist jedoch nur eine Speicherung des niederwertigen Adreßbytes. Ein Vorteil der Schaltungen in Abb.7.12 ist jedoch, daß hierdurch auch der höherwertige Teil des Adreßbusses die gleichen Eigenschaften wie der untere Teil bekommt und zu keiner Zeit unbestimmte Zustände - wie in Abb.7.8 - auftreten können.

Wenn der Mikroprozessor über den HOLD-Eingang angehalten wird, sorgt das Quittierungssignal HLDA dafür, daß die Treiber den hochohmigen Zustand annehmen.

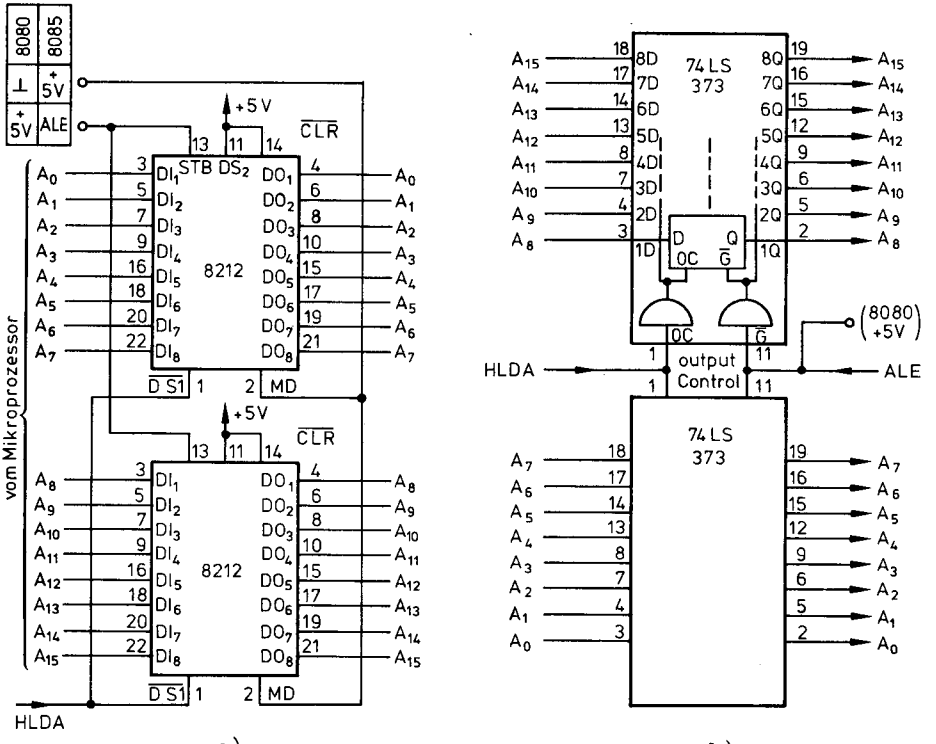


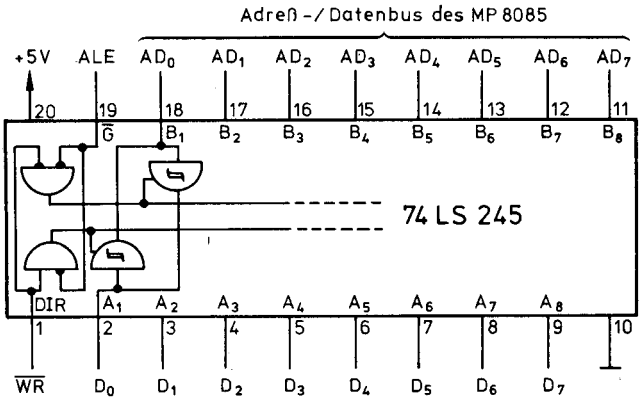
Abb.7.12: Adreßbus-Treiberschaltungen für den MP 8080/85. Für den MP 8085 erfolgt jeweils eine Adreßzwischenspeicherung.

Datenbus

Der Datenbus (DB) besteht aus 8 Leitungen und kann in beiden Richtungen (bidirectional) Daten transportieren. Die 8bit-parallelen-Daten repräsentieren jeweils eins von 256 (2^8) verschiedenen Datenworten die hier darstellbar sind.

Über den Datenbus läuft der Informationsaustausch des Mikroprozessors mit seinen Speichern und der Umwelt (Befehle, Rechenergebnisse, Meßdaten, periphere Steuersignale u.v.m.).

Beim MP 8080 sorgt der Systemsteuerbaustein für eine Verstärkung der Bussignale. In welcher Richtung der Datenbus arbeitet, wird durch den MP 8080 bestimmt. Ein H-Signal auf dem Ausgang DBIN zeigt an, daß sich der Datenbus im Eingabezustand befindet. Bei einem L-Signal auf dem Ausgang WR ist der Datenbus auf Ausgabe geschaltet. Ein Datenbustreiber für den MP 8085 ist in Abb.7.13 dargestellt.



G	DIR	Datenrichtung
L	L	B → A
L	H	A → B
H	X	hochohmig

Abb. 7.13: Datenbustreiber für den MP 8085

Je zwei Tristate-Gatter mit Schmitt-Trigger-Eingängen stellen wahlweise eine Verbindung zwischen dem AD-Bus des MP 8085 und dem äußeren Datenbus oder in Gegenrichtung her. Die Auswahl der Datenrichtung geschieht über den Eingang DIR (direction), er kann mit dem Ausgang WR des MP 8085 verbunden werden. Der Bausteinauswahleingang G (CS) - verbunden mit dem ALE-Ausgang - sorgt dafür, daß der Treiberbaustein hochohmig ist, wenn auf dem AD-Bus, während des Taktes T_1 (Abb.7.8), der niederwertige Adreßteil ausgegeben wird.

Steuerbus

Der Steuerbus besteht aus einer variablen Anzahl von Steuerleitungen. Die wichtigsten sind die Lese-, Schreib- und Bausteinauswahlsignale.

Abb.7.14 zeigt ein Beispiel für den Anschluß von bis zu 8 Speicherbausteinen an den MP 8085. Über das Steuersignal WR (write) wird die Arbeitsrichtung des Datenbusses festgelegt. Ob ein Speicherbaustein mit dem Datenbus in Verbindung tritt, wird jedoch durch das Bausteinauswahlsignal CS (chip select) bestimmt. Diese Auswahlsignale lassen sich beispielweise mit dem Baustein 8205 erzeugen. Der 8205 hat selbst drei Auswahleingänge E_1, E_2, E_3, nur wenn diese die Zustände L, L, H annehmen, wird auf einem der 8 Auswahlleitungen $O_{0...7}$ ein L Signal abgegeben.

Indem man das Steuersignal IO/M auf einen der L-aktiven Eingänge E_1 oder E_2 legt, kann verhindert werden, daß bei einem IN- oder OUT-Befehl (Peripheriezugriff) gleichzeitig ein Speicher angesprochen wird. Zur Erinnerung sei bemerkt, daß bei einem IN- oder OUT-Befehl die 1Byte-Adresse dieser Befehle auf dem nieder- und dem höherwertigen Teil des Adreßbusses ausgegeben wird.

Beispiel: Bei der Ausführung des Befehles OUT 03 sendet der MP 8080/85 die Adresse 0303_H aus. Wenn der 8205 hierbei nicht über das Signal IO/M = H gesperrt würde, brächte der Mikroprozessor den Inhalt des Akkumulators sowohl auf den Ausgang mit der Adresse 03 als auch in den Speicherplatz mit der Adresse 0303_H.

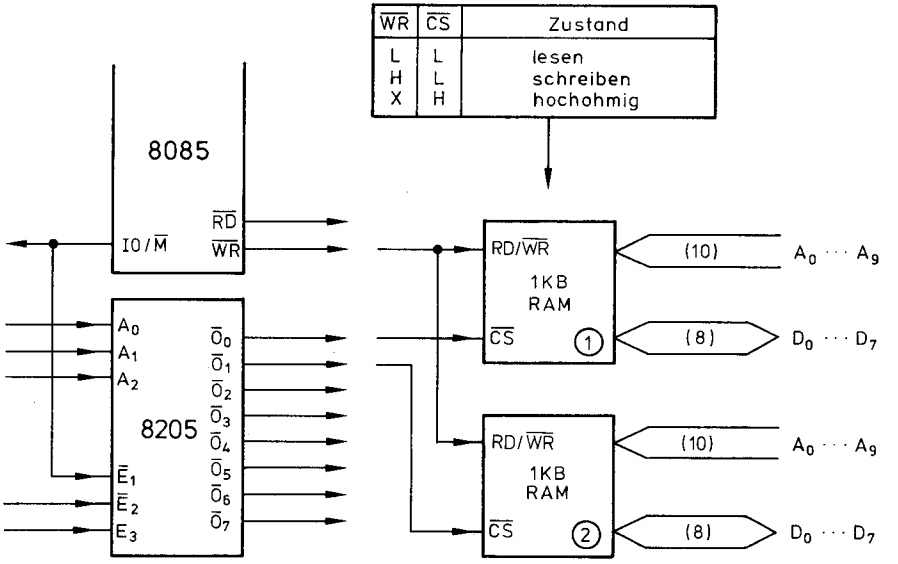


Abb. 7.14: Steuerbussignale zum Anschluß von Speicherbausteinen (RAM = Arbeitsspeicher) an den MP 8085

Da beide Speicher in Abb.7.14 parallel an den Datenbus und die ersten zehn Adreßbits angeschlossen werden, wird ihnen über die CS-Signale ein unterschiedlicher Adreßbereich zugewiesen. Jeder Speicher umfaßt einen relativen Adreßbereich von 0 bis 1023_D (1024 Speicherplätze), deshalb benötigt man 10 Adreßleitungen (2^{10} = 1024), um alle Plätze eindeutig adressieren zu können. Diese Speicherbereiche müssen nun in dem zur Verfügung stehenden Adreßraum des Mikroprozessors (0 bis 65535_D) eindeutig und überschneidungsfrei angeordnet werden. Hierzu bedient man sich häufig eines Decoders (8205) wie in Abb.7.14. Sobald der 8205 ausgewählt ist (E_1 = L, E_2 = L, E_3 = H) gibt einer der acht Ausgänge (O_i) ein L-Signal ab. Welcher Ausgang den L-Zustand annimmt, ist durch die parallel auf den drei Eingängen A_0, A_1, A_2 anstehende Dualzahl i festgelegt. Im oberen Teil der Abb.7.15 sind die acht (2^3) möglichen Zustände dargestellt.

Auswahllogik des 8205

A_0	L	H	L	H	L	H	L	H	$E_1 = L$
A_1	L	L	H	H	L	L	H	H	$E_2 = L$
A_2	L	L	L	L	H	H	H	H	$E_3 = H$
L auf	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	

Speicher-Baustein	Adreßbus																Adreßbereich	
	2^{15}	14	13	12	11	10	9	8	7	6	5	4	3	2	1_2	0	Hex.	Dez.
RAM 0	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	0000	0000
	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	03FF	1023
RAM 1	L	L	L	L	L	H	L	L	L	L	L	L	L	L	L	L	0400	1024
	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	07FF	2047

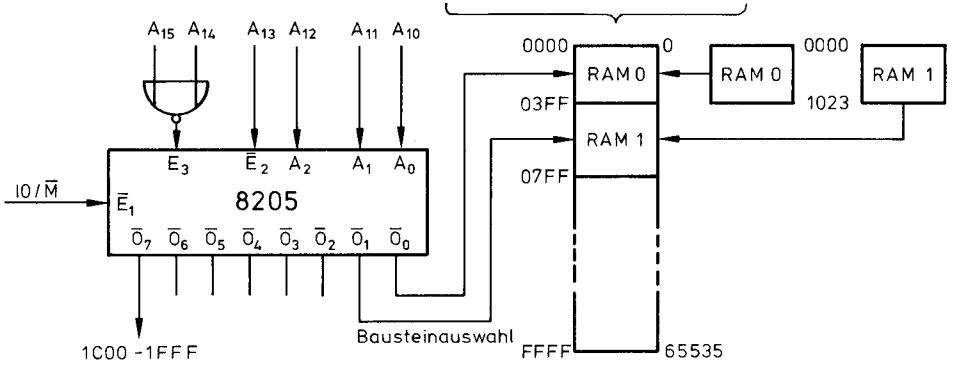


Abb.7.15: Speicherauswahlschaltung mit dem 1 aus 8 Decoder 8205 für den MP8085

An welche Adreßleitungen der 8205 angeschlossen werden muß, um zwei 1KB-Speicher, von der Adresse 0000 beginnend, hintereinander anzuordnen, zeigt Abb.7.15.

Die untersten 10 Adreßbits selektieren die einzelnen Speicherplätze in einem Speicherbaustein, falls er durch ein Bausteinauswahlsignal aktiviert wird. Für den ersten Speicher (RAM 0), dessen relative Adressen (0 - 3FF) mit den absoluten übereinstimmen, da er bei 0000 beginnt, ändern sich die oberen Adreßbits (ab A_{10}) nicht, d.h. sie müssen den Zustand L behalten. Der zweite Speicher dagegen beginnt mit der absoluten Adresse 400_H. Dies bedeutet ein H auf A_{10} und eine Null für den Speicher ($A_{0...9}$ = L).

Diese Systematik wird benutzt, um die aneinandergereihten Speicher durchzunummerieren und immer nur einen von ihnen aufzurufen. Wenn nämlich die auf die unteren 10 Adreßbits folgenden 3 Bits (A_{10}, A_{11}, A_{12}) an die Eingänge A_0, A_1, A_2 des 8205 angeschlossen werden, gibt der Index (0 bis 7) an den Ausgangsleitungen die Nummer des hierüber aus-

gewählten Speichers an. Auf diese Weise lassen sich dann mit den oberen 6 Adreßbits ($A_{15...10}$) 2^6 = 64KB Speicher adressieren. Der Baustein 8205 kann nur die Adreßleitungen A_{10}, A_{11}, A_{12} decodieren. Die drei höchsten Adreßbits müssen deshalb auf andere Weise festgelegt werden. In Abb.7.15 erfolgt die Festlegung dieser Bits durch die Verwendung eines NOR-Gatters und der Auswahleingänge E_2, E_3 des 8205.

Zur Auswahl weiterer Speicher können zusätzliche Decoder eingesetzt werden.
Wenn die anzuschließenden Speicher mehr Plätze haben - z.B. 2KB - verschieben sich die Anschlüsse des 8205 in Abb.7.15 um ein Bit nach links, da für den Anschluß der einzelnen Speicherbausteine die unteren 11 Bits ($A_{0...10}$, 2^{11} = 2048) des Adreßbusses nötig sind.

Beispielschaltung

Abschließend sollen, anhand einer Beispielschaltung für alle drei Busse, noch einige offene Fragen geklärt werden.

In Abb.7.16 ist diese Schaltung für ein komplettes Grundsystem, einschließlich aller Bustreiber und einer Bausteinauswahllogig für 1KB-Speicher, dargestellt.
An die auf diese Weise verstärkten Busse des MP 8085 lassen sich beliebig viele Bausteine anschließen, so daß ein maximaler Ausbau zu einem Mikrocomputer mit 64KB Speicher und vielen Ein-/Ausgabekanälen möglich ist.
Die H-aktiven Interrupt- (Unterbrechungs-) Eingänge sowie der HOLD-Eingang des MP 8085 sind über Widerstände (pull-down) mit der Masse verbunden, um ein unbeabsichtigtes Auslösen zu verhindern und weil bei MOS-Schaltkreisen alle Eingänge festgelegt sein müssen.

Der Reset-Eingang ist mit einer sog. Einschaltreset-Anordnung versehen. Sie besteht aus einer RC-Kombination und einer Freilaufdiode und hat die Aufgabe nach dem Einschalten den Reset-Eingang solange auf dem logischen Pegel L zu halten, bis der Mikroprozessor dieses Signal annimmt und der normale Rücksetzvorgang abläuft.

Zur Annahme des Reset-Signales nach dem Einschalten benötigt der MP 8085 ca. 10ms, da innerhalb dieser Zeit ein interner Ossillator anlaufen muß, der über eine Ladungspumpe eine negative Substratvorspannung erzeugt.

Durch das Reset werden im MP 8085 eine Reihe von Registern und Flip-Flops in einen definierten Zustand gebracht (z.B. der Befehlszähler auf 0000). Die allgemeinen Register, der Akkumulator, der Stapelzeiger und die Kennzeichenbits werden jedoch nicht verändert, sie bekommen beim Einschalten einen zufälligen Inhalt und behalten bei einem zwischenzeitlich erfolgenden Reset ihren alten Wert.

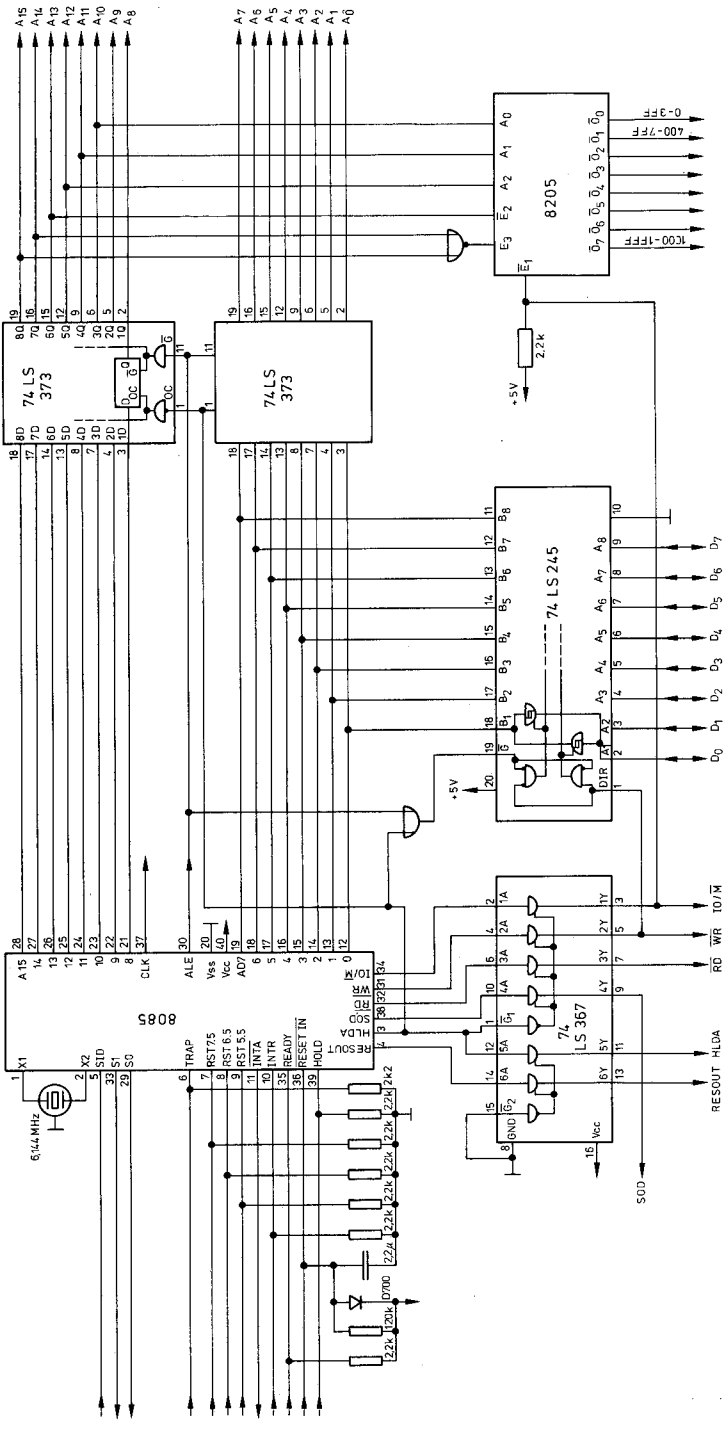


Abb. 7.16: Stromlaufplan für ein Grundsystem mit dem MP 8085 und Bustreibern

Wenn der MP 8085 über den HOLD-Eingang angehalten wird, bearbeitet er den momentanen Maschinenzyklus intern noch zu Ende, gibt aber ab T_3 schon das Quittierungssignal HLDA heraus. Dieses Signal sorgt dann in Abb.7.16 dafür, daß die Bustreiber den hochohmigen Zustand annehmen. In diesem Zustand können dann von außen Adressen und Steuersignale aufgeprägt und Informationen auf dem Datenbus ausgetauscht werden (DMA = direct memory access).

Die Steuerausgänge RESETOUT und HLDA verbleiben im nicht hochohmigen Zustand.

7.4 Unterbrechungen (Interrupts)

Während ein Mikroprozessor mit der Abarbeitung eines Programmes beschäftigt ist, kann über ein Signal von außen (Interrupt-Eingänge) bewirkt werden, daß dieses Programm unterbrochen wird und zunächst ein anderes Programm abläuft. Nach der Ausführung des Unterbrechungsprogrammes springt der Mikroprozessor in das alte Programm zurück.

Eine solche Unterbrechung wird Interrupt genannt.

Beispiele: Anwendung von Interrupts:

- Anschluß peripherer Geräte an den Mikrocomputer (MC), die nicht ständig bedient werden, sondern sich über einen Interrupteingang melden, wenn sie mit dem MC in Kontakt treten wollen.
- Eingabe eines Gefahrensignales auf das der MC reagieren muß.
- Echtzeit-Anwendungen. Hierbei wird der MC im Rhytmus einer äußeren 'Uhr' regelmäßig unterbrochen.

Der Interrupt stellt eine Alternative zur sog. Abfragemethode (polling) dar. Bei der Abfragemethode werden alle Geräte zyklisch auf eine Bedienanforderung hin abgefragt. Dies bedeutet, daß sich der Mikroprozessor ständig in einer Abfrageschleife befindet und keine anderen Aufgaben nebenher übernehmen kann, selbst wenn die Geräte nur in großen Zeitabständen zu bedienen sind.

Ein Interruptsignal bewirkt im Mikroprozessor folgendes:

1. Der gerade laufende Befehlszyklus wird zu Ende geführt.
2. Die Adresse des folgenden Befehles wird gespeichert (MP 8080/85: über den Stapelzeiger ins RAM gebracht).
3. Das Interrupt-Bedienprogramm wird abgearbeitet.
4. Rücksprung ins Ausgangsprogramm zur gespeicherten Adresse.

Der MP 8080/85 verfährt bei einem Interrupt so, als würde ein CALL-Befehl abgearbeitet. Durch ein von außen kommendes Signal kann demnach zu einem beliebigen Zeitpunkt ein Unterprogrammaufruf erzwungen werden. - Auf eine Sperrung der Interrupts wird später eingegangen. - Zunächst bleibt noch die Frage zu klären, wohin ein solcher Unterprogrammsprung erfolgt.

RST		0	1	2	3	4	TRAP	5	5.5	6	6.5	7	7.5
Pin	8080	14	14	14	14	14		14		14		14	
Nr.	8085	10	10	10	10	10	6	10	9	10	8	10	7
Datenbus		C7	CF	D7	DF	E7	XX	EF	XX	F7	XX	FF	XX
Ziel-adresse		0_H	8	10	18	20	24	28	2C	30	34	38	3C
Priorität		5.	5.	5.	5.	5.	1.	5.	4.	5.	3.	5.	2.

Abb.7.17: Die Interrupts des MP8080 und des MP8085 (RST= restart)

In Abb.7.17 sind - neben anderen Daten - die Zieladressen der Interruptsprünge dargestellt. Wenn beispielsweise am Pin Nr.9 des MP 8085 ein H-Signal erscheint, wird das laufende Programm unterbrochen und ein Unterprogrammsprung zur Adresse 002C durchgeführt.

INT/INTR:

Über den Eingang INT (Pin 14/8080) oder INTR (Pin 10/8085) - beide haben die gleiche Bedeutung - können acht Interrupts mit unterschiedlichen Zieladressen (vektorisierte Interrupts) erfolgen. Welche der Adressen gemeint ist, muß dem Mikroprozessor, nach der Annahme des Interrupts (INTA), über den Datenbus mitgeteilt werden. In der 4. Zeile der Abb.7.17 sind die acht Bytes angegeben, die den MP 8080/85 veranlassen zu den entsprechenden Adressen (Zeile 5) zu springen. Diese acht Bytes sind nichts anderes als die Objektcodes der Befehle RST 0 bis RST 7 (Abb.3.4c).

Wenn ein Gerät über den INT- oder INTR-Eingang eine Bedienung durch den Mikrocomputer anfordert, kann es beim Erscheinen des Signales INTA (Unterbrechungsannahme) auf den Datenbus einen der acht RST-Befehle geben und dadurch die Zieladresse des Unterprogrammaufrufes - mit dem entsprechenden Bedienprogramm - festlegen.

Darüber hinaus gibt es jedoch auch noch die Möglichkeit, anstelle eines RST-Befehlsbytes, den Operationsteil des CALL-Befehles (CD) auf den Datenbus zu senden. In diesem Fall gibt der MP 8080/85 zwei weitere INTA-Signale ab, mit denen die zwei, zu einem CALL-Befehl gehörenden, Adreßbytes eingeholt werden. Der Ort des Interrupt-Bedienprogrammes

wird dann über die Adresse des CALL-Befehles festgelegt. Zur Unterstützung dieses Verfahrens gibt es einen speziellen programmierbaren Interrupt-Steuerbaustein (8259).

Die in Abb.7.17 angegebenen Prioritäten gelten nur für den MP 8085, da nur dieser über mehr als einen Interrupteingang verfügt. Hierbei hat der INTR-Eingang die niedrigste Priorität. Dies bedeutet, daß der INTR-Interrupt nur angenommen wird, wenn nicht gleichzeitig auf einem der anderen Interrupteingänge ein Anforderungssignal anliegt. Im folgenden Beispiel werden weitere Möglichkeiten und Besonderheiten des INT-Einganges beschrieben.

Beispiel für den MP 8080

In einem Grundsystem gemäß Abb.7.4, gibt es eine einfache Möglichkeit der Interrupt-Verarbeitung. Hierzu wird der INTA-Ausgang über einen Widerstand an die + 12V Versorgungsspannung gelegt. Wenn das Interrupt-Quittierungssignal (INTA) erscheint, sendet der Systemsteuerbaustein (8228) einen RST7-Befehl auf den Datenbus. Der MP 8080 führt dann einen Unterprogrammsprung zur Adresse 0038 durch. Der Anschluß mehrerer Geräte, unter Ausnutzung der geschilderten Eigenschaft des Grundsystemes, ist in Abb.7.18 dargestellt.

Hierbei werden die Bedienanforderungssignale über eine ODER-Verknüpfung dem INT-Eingang des MP 8080 zugeleitet. Wenn ein oder mehrere Geräte bedient werden möchten, erhält der MP 8080 ein Interrupt-Signal.

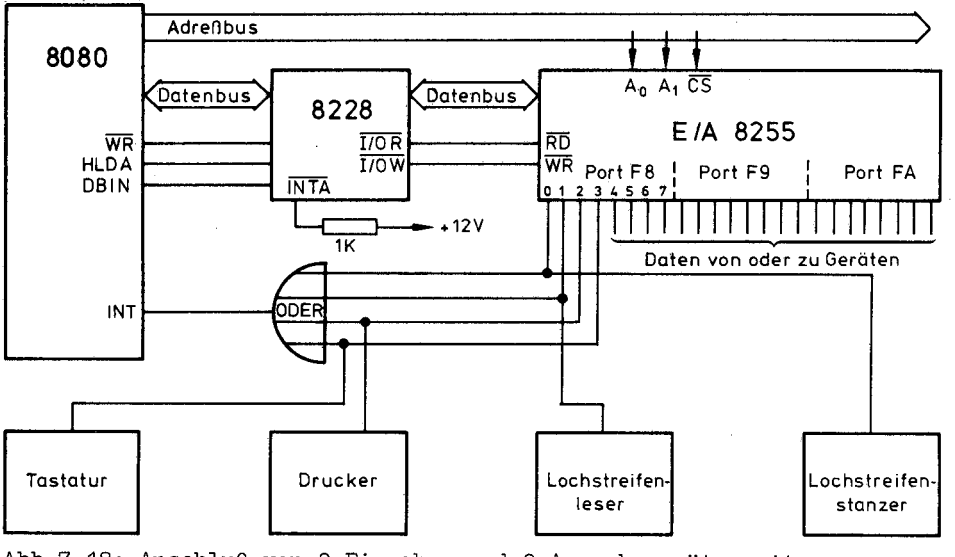


Abb.7.18: Anschluß von 2 Eingabe- und 2 Ausgabegeräten mit Bedienanforderung (Interrupt) an den MP 8080. Die Daten-Ein-/Ausgänge wurden nicht gezeichnet.

Zur Feststellung, welches Gerät den Interrupt ausgelöst hat, ist es erforderlich die vier Anforderungssignale mit den Eingängen eines Ein-/Ausgabebausteines zu verbinden.

```
0000                    ORG     0038H
0038 F5                 PUSH    PSW         ;Akku + Flags sichern
0039 C5                 PUSH    B           ;Register B+C sichern
003A D5                 PUSH    D           ;Register D+E sichern
003B E5                 PUSH    H           ;Register H+L sichern
003C DBF8               IN      0F8H        ;Port F8 einlesen
003E 0F                 RRC                 ;Bit 0 ins Carry-Flag schieben
003F DC0001             CC      LOST        ;Aufruf Lochstreifenstanzer-Prog.
0042 0F                 RRC                 ;Bit 1 ins Carry-Flag schieben
0043 DC8001             CC      LOLE        ;Lochstreifenleser Bedienprogramm
0046 0F                 RRC                 ;Bit 2 ins Carry-Flag schieben
0047 DC6300             CC      DRUC        ;Zum Druckerprogramm,wenn Carry=H
004A 0F                 RRC                 ;Bit 3 ins Carry-Flag schieben
004B DC5400             CC      TAST        ;Tastatur Bedienprogramm
004E E1                 POP     H           ;  Inhalte
004F D1                 POP     D           ;  der Register und Flags
0050 C1                 POP     B           ;  des unterbrochenen Programmes
0051 F1                 POP     PSW         ;  wiederherstellen
0052 FB                 EI                  ;INTE-FF setzen
0053 C9                 RET                 ;Zurück ins laufende Programm

                ;******** UP TASTATUR ********
0054 F5         TAST:   PUSH    PSW
0055 DBFA               IN      0FAH        ;Tastaturport FA einlesen
0057 E67F               ANI     7FH         ;8. Bit Null setzen (7bit ASCII)
0059 2A0012             LHLD    CURSO       ;Speicheradresse laden
005C 77                 MOV     M,A         ;Eingelesenes Zeichen -) Speicher
005D 23                 INX     H           ;Speicheradresse erhöhen
005E 220012             SHLD    CURSO       ;Neue Speicher-Adr. abspeichern
0061 F1                 POP     PSW
0062 C9                 RET                 ;Zurück ins Interrupt-Bedienprogr.
0100            LOST    EQU     0100H
0180            LOLE    EQU     0180H
0063            DRUC    EQU     0063H
1200            CURSO   EQU     1200H
```

Abb. 7.19: Interrupt-Bedienungsprogramm für das Beispiel in Abb. 7.18

Bei einem Interrupt liest der Mikroprozessor den Inhalt des Ports F8 ein und kann feststellen welche Geräte zu bedienen sind. Hierdurch ist es auch möglich beliebige Prioritäten festzulegen, für den Fall, daß mehrere Geräte gleichzeitig einen Interrupt auslösen.

Im Programmbeispiel in Abb.7.19 wurde folgende Priorität festgelegt: Lochstreifenstanzer, Lochstreifenleser, Drucker, Tastatur.

Wenn eine Interrupt-Anforderung vorliegt, springt der MP 8080 zur Adresse 0038. Dort steht dann entweder das Interrupt-Bedienprogramm, oder, falls an dieser Stelle nicht genügend Platz vorhanden ist, ein Sprungbefehl der zu diesem Programm führt.

Im Interrupt-Bedienprogramm (Abb.7.19) müssen zunächst die Inhalte aller Register und der Kennzeichenspeicher (flags) gesichert werden, damit beim Rücksprung ins unterbrochene Programm alles normal weiterlaufen kann. Dann wird der Port F8 eingelesen und damit der oder die

Unterbrecher festgehalten. Das folgende Programm lenkt den Mikroprozessor über bedingte Unterprogrammaufrufe zu den Programmteilen, die das jeweilige Gerät bedienen. Für ein Gerät, nämlich die Tastatur, wurde in Abb.7.19 ein entsprechendes Bedienprogramm angefügt.

Nach der Abarbeitung aller angeforderten Bedienprogramme, kehrt der Mikroprozessor über den RET-Befehl ins unterbrochene Programm zurück. Für den Fall, daß in einem Programm keine Unterbrechung erfolgen darf, besteht die Möglichkeit über den Befehl DI (disable interrupts) eine Interrupt-Annahme auszuschließen. Am Ende des unterbrechungsfreien Programmteiles kann über den Befehl EI (enable interrupts) die Annahme von Interrupts wieder erlaubt werden.

Der Ausgang INTE (interrupt enable) zeigt an, ob ein Interrupt angenommen wird. Intern ist diesem Ausgang ein RS-Flip-Flop zugeordnet.

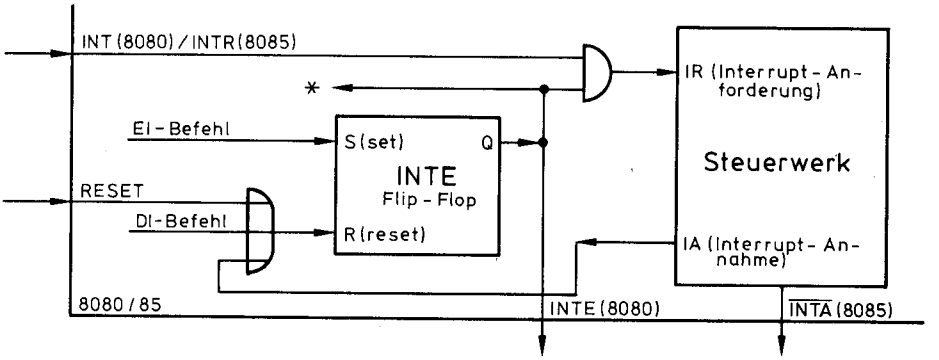


Abb.7.20: Vereinfachtes logisches Bild für die Bedingungen zur Annahme eines INT-/INTR- Interrupts.
* Zu weiteren Interrupts des MP 8085.

Ein Interrupt wird angenommen, wenn INTE = H. Dies ist der Fall, wenn kein Reset vorliegt, kein DI-Befehl gegeben, bzw. durch einen EI-Befehl aufgehoben wurde und wenn nicht schon vorher ein anderer Interrupt erfolgte, so daß eine Sperrung über IA (Abb.7.20) stattgefunden hat.

Hieraus geht hervor, daß während der Abarbeitung eines Interrupts kein weiterer angenommen wird, da INTE = L, falls kein EI-Befehl gegeben wurde.

Ein Interrupt wird auch angenommen, wenn sich der Mikroprozessor im HOLD-Zustand befindet oder durch einen HLT-Befehl (Halt) angehalten wurde.

Die Interrupts des MP 8085

Wie Abb.7.17 zeigt, verfügt der MP 8085, gegenüber dem MP 8080, über vier weitere Interrupts.

Der Interrupteingang INTR hat die gleichen Eigenschaften wie der INT-Eingang beim MP 8080. Lediglich der Ausgang des INTE-Flip-Flops (Abb.7.20) ist nicht nach außen geführt, dafür aber über den RIM-Befehl (Abb.7.22) abfragbar.

Alle Interrupts (INTR, RST5.5, RST6.5, RST7.5), außer dem TRAP, sind über die Befehle DI und EI maskierbar (abschaltbar). Während eines HOLD- oder Halt- (HLT-Befehl) Zustandes werden alle Interrupts angenommen.

Eingangsbedingungen:
Die Bedingungen am Eingang, für die Annahme eines Interrupts, sind folgende:

INTR, RST5.5, RST6.5 : H-Pegel bis zur Annahme des Interrupts.
RST7.5 : Ansteigende Flanke mit Speicherung.
TRAP : Ansteigende Flanke und H-Pegel bis zur Annahme des Interrupts.

Außer beim RST7.5, der zwischengespeichert wird, müssen alle anderen Interruptsignale wenigstens für die Dauer des längsten Befehlszyklusses anliegen, um sicher angenommen zu werden. Dies ist dadurch bedingt, daß das Interruptsignal asynchron ankommt und der gerade laufende Befehlszyklus erst zu Ende bearbeitet wird, bevor die Unterbrechung erfolgen kann.

Beispiel: 3MHz Taktfrequenz; längster Befehl: CALL mit 18 Takten.
Die minimale Interrupt-Impulslänge t_I beträgt dann:
$t_I = 18/3MHz = 6\mu s$

Prioritäten:
Die fünf Interrupteingänge haben untereinander eine fest vorgegebene Priorität (Abb.7.17), wobei der TRAP-Interrupt die höchste und der INTR-Interrupt die niedrigste Priorität hat.

Liegen mehrere Interrupts vor, so wird der mit der höchsten Priorität angenommen. Dies bezieht sich jedoch nur auf das gleichzeitige Vorliegen mehrerer Interrupt-Anforderungen. Wenn dagegen gerade ein Interrupt-Unterprogramm hoher Priorität abgearbeitet wird und es erfolgt ein Interrupt niedriger Priorität, so wird dieser angenommen, falls das INTE-Flip-Flop, das bei jeder Interrupt-Annahme zur Sperrung weiterer Interrupts rückgesetzt wird, durch einen EI-Befehl gesetzt wird und außerdem über den SIM-Befehl (siehe unten) keine Sperrung erfolgt ist.

Interrupt Maske:
Der MP 8085 hat gegenüber dem MP 8080 zwei weitere Befehle, die zur seriellen Ein- und Ausgabe (Kap.9.2) und zur Steuerung der zusätzlichen Interrupts dienen.

Mittels des Befehles SIM (set interrupt mask) lassen sich die Interrupts RST5.5, RST6.5 und RST7.5 maskieren. Dieses Maskieren dient der Sperrung oder Freigabe von Interrupts.

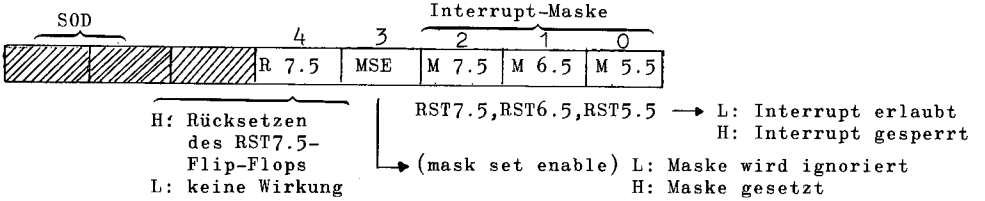


Abb. 7.21: Setzen der Interrupt-Maske mittels des SIM-Befehles (und serielle Ausgabe auf SOD)

Hierzu wird in den Akkumulator ein Byte eingeschrieben, dessen unterste 5 Bits diese Steuerung übernehmen, wenn danach der Befehl SIM gegeben wird (Abb.7.21).
Über das Bit 4 (R7.5) läßt sich das zur Speicherung einer RST7.5 Anforderung benutzte Flip-Flop zurück setzen (Abb.7.23)

```
Beispiel:   MVI A,00011001B ; Interrupt-Steuerwort in den Akku
                            ; bringen
            SIM             ; RST7.5 FF rücksetzen, Interruptmaske
                            ; setzen: RST7.5 und RST6.5 erlaubt,
                            ; RST5.5 nicht erlaubt.
```

Der Befehl RIM (read interrupt mask) bringt 7 Bits in den Akkumulator, die Aufschluß über den Zustand des Interrupt-Systemes geben.

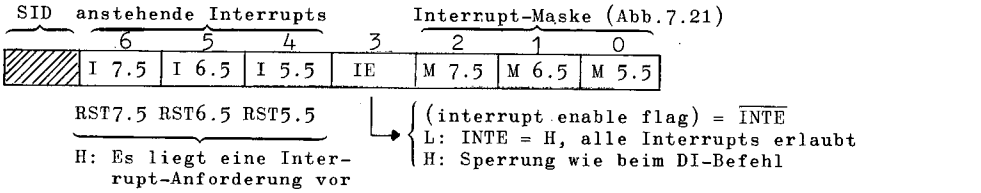


Abb. 7.22: Zustand des Interruptsystemes (und des seriellen Einganges SID), abfragbar über den RIM-Befehl

Anhand der Bits 0, 1, 2 (Abb.7.22) läßt sich feststellen, wie die Interrupt-Eingänge RST5.5, RST6.5 und RST7.5 maskiert worden sind und Bits 4, 5, 6 lassen erkennen, ob gerade eine Interrupt-Anforderung vorliegt. Die Bits 4, 5, 6 (pending interrupts) geben den momentanen Zustand dieser Eingänge an, unabhängig von jeder Maskierung. Das Bit 3 zeigt an, ob eine generelle Sperrung der Interrupts (TRAP läßt sich nie sperren) oder eine Freigabe vorliegt (INTE). Bei einem Reset des MP 8085 werden die Flip-Flops INTE, RST7.5 und TRAP zurückgesetzt und die Interrupt-Maske mit H,H,H gefüllt.

RST5.5 und RST6.5:
Die Interrupt-Vektoren von RST5.5 und RST6.5 sind unveränderbar und zeigen auf die Adressen 002C und 0034_H.

Zur Anforderung ist ein H-Pegel am Eingang erforderlich. Es erfolgt keine Speicherung, so daß die Anforderung wenigstens über eine Befehlslänge bestehenbleiben muß. Liegt der H-Pegel noch nach Beendigung des Interrupt-Unterprogrammes an, so wird das Unterprogramm erneut aufgerufen.

Die Annahme des Interrupts kann über den Befehl DI und durch entsprechende Maskierung (SIM-Befehl) verhindert werden. Auch wenn der Interrupt nicht angenommen wird, kann über den RIM-Befehl der Zustand der Interrupt-Eingänge abgefragt (polling) und damit ebenfalls auf die Anforderung reagiert werden.

RST7.5:

Dieser Interrupt-Vektor zeigt unveränderbar auf die Adresse 003C.

Zur Anforderung genügt ein kurzer H-Impuls, da eine Zwischenspeicherung erfolgt.

Das RST7.5-Eingangs-Flip-Flop wird durch die Interruptmaske M7.5 und durch das INTE-FF nicht beeinflußt (Abb.7.23), so daß eine Interrupt-Anforderung auch später, nach einer Freigabe, angenommen werden kann.

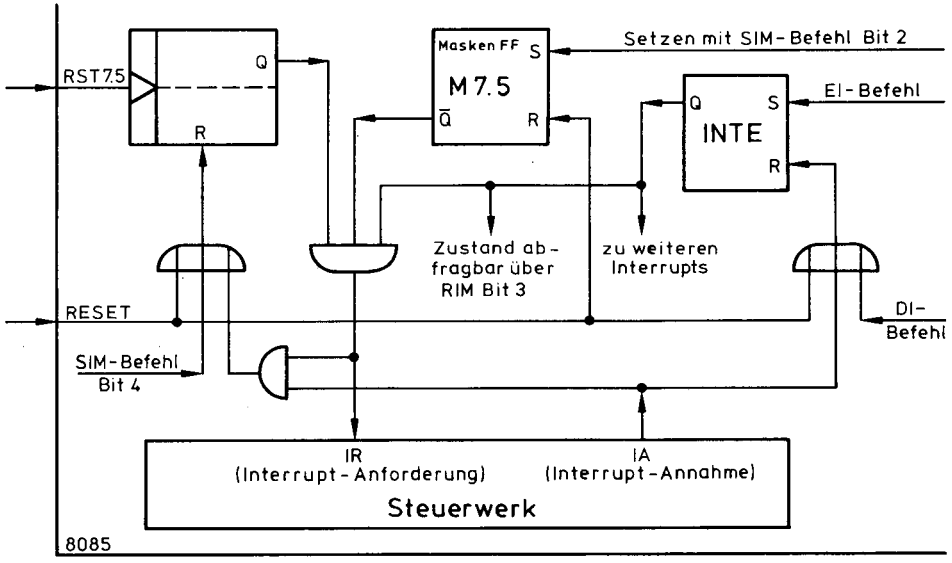


Abb.7.23: Logische Verknüpfungen beim RST 7.5- Interrupt

Darüber hinaus besteht die Möglichkeit den Zustand des Eingangs-Flip-Flops durch den RIM-Befehl abzufragen und auch bei gesperrtem Interrupt-System eine entsprechende Interrupt-Routine abarbeiten zu lassen.

TRAP:

Der TRAP-Vektor (trap = Falle) ist auf die Adresse 0024_H gerichtet.

Dieser Interrupt hat die höchste Priorität und ist nicht maskierbar. Er wird vorallem zur Aktivierung nicht aufschiebbarer Routinen benutzt.

Beispiele:
- Rettung aktueller Daten bei Stromausfall.
- Behebung schwerwiegender Fehler bei einer Prozeß- oder Maschinensteuerung.
- Abwendung einer akuten Gefahr.

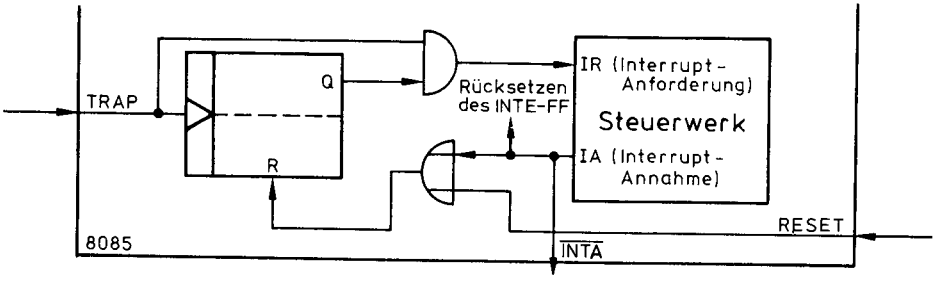


Abb.7.24: Logische Bedingungen beim TRAP- Interrupt

Die steigende Flanke des TRAP-Signales setzt das Eingangs-Flip-Flop. Danach muß noch für eine Befehlslänge H-Pegel anliegen, damit der Mikroprozessor bei IR (Abb.7.24), nach der Bearbeitung des gerade aktuellen Befehles, noch eine Anforderung vorfindet.

Wichtig ist, daß durch das Quittierungssignal IA nicht nur das Eingangs-Flip-Flop zurück gesetzt wird, sondern auch das INTE-Flip-Flop.

Dies hat die selbe Wirkung wie ein DI-Befehl, d.h. alle anderen Interrupts sind gesperrt.
Durch das Rücksetzen des Eingangs-Flip-Flops wird verhindert, daß bei weiter anliegendem H-Pegel am TRAP-Eingang, der Interrupt nochmals angenommen wird; hierzu ist ein erneuter L/H-Wechsel erforderlich.

Wenn eine TRAP-Interrupt-Routine abgearbeitet worden ist, wird es im allgemeinen notwendig sein das Interrupt-System wieder in den alten Zustand zu bringen. Hierzu ist es erforderlich, daß das durch den TRAP rückgesetzte INTE-Flip-Flop den vorherigen Zustand wieder annimmt. Zunächst muß herausgefunden werden, ob das INTE-F-F gesetzt (H) oder rückgesetzt (L) war, bevor die Unterbrechung erfolgte. Diese

Information kann über den ersten RIM-Befehl, der nach dem TRAP-Interrupt gegeben wird, in den Akkumulator geholt werden. Bit 3 (IE, Abb.7.22) enthält dann den invertierten INTE-Zustand vor dem TRAP-Ereignis. Jede weitere Abfrage über den RIM-Befehl gibt den momentanen Zustand an.

War das INTE-FF vor dem Interrupt gesetzt, so kann es über den EI-Befehl erneut gesetzt und damit die Interrupts freigegeben werden.

7.5 Übungsaufgaben

7.1 Wozu ist es erforderlich die Busse in den hochohmigen Zustand zu bringen?

7.2 Was verstehen Sie unter einem gemultiplexten Bus?

7.3 Wozu wird demultiplext?

7.4 Was verstehen Sie unter einem Schreib- (Lese-) Signal?

7.5 Was ist ein Maschinenzyklus und ein Befehlszyklus?

7.6 Wozu dient immer der erste Teil des ersten Maschinenzyklusses?

7.7 Wie kann der MP 8080/85 zur Anpassung an langsame Speicher verlangsamt werden?

7.8 Welche Zeit wird benötigt, um bei 3MHz-Takt (MP 8085)

a) einen 1 Byte-Befehl ins Befehlsregister zu holen,

b) ihn zu interpretieren,

c) ein Registerpaar zu dekrementieren?

7.9 Welche Impulsverzögerung tritt in einem Mikrocomputer auf, wenn an den MP 8085 6 Programmspeicher (2716), 8 Arbeitsspeicher (2114) und 4 Ein-/Ausgabebausteine (8255) angeschlossen werden? Die Verdrahtungskapazität betrage 120pF.

7.10 Wieviele Adressen lassen sich mit einem 20bit Adreßbus darstellen?

7.11 Wieviel Zeit steht ungefähr für die Übernahme des unteren Adreßbytes beim MP 8085 zur Verfügung?

7.12 Mit wieviel Adreßleitungen muß ein Speicherbaustein der Kapazität 1/2KB angeschlossen werden?

7.13 Es wird der Befehl IN 07 abgearbeitet. Welche Zustände nehmen die Eingänge A_0, A_1, A_2, E_1, E_2, E_3 des 8205 in Abb.7.15 während des 3. Maschinenzyklusses an?

7.14 Wie könnte in Abb.7.15 ein weiterer 8205 angeschlossen werden, um 8 Bausteinauswahlsignale für den Adreßbereich 2000 bis 3FFF zu erzeugen?

7.15 Wozu dient das ODER-Gatter in Abb.7.16?

7.16 Wozu dient der Widerstand an E_1/8205 Abb.7.16?

7.17 Welchen Vorteil hat ein Interrupt gegenüber der Abfragemethode (polling)?

8. Die Speicher

8.1 Überblick

Der Speicher eines Mikrocomputers soll digitale Informationen aufnehmen und behalten, d.h. es handelt sich hierbei ausschließlich um digitale Speicher. Die Größe und Schnelligkeit der Speicher sind wesentliche Kriterien für die Leistungsfähigkeit eines Mikrocomputers.

Doch zunächst ist es erforderlich, einige wichtige Begriffe zu klären:

Speicherkapazität:	Menge der Bits oder Bytes die ein Speicher aufnehmen kann. Z.B. 1KB = 1024x8bit.
Zugriffszeit:	Zeit die zwischen der Anforderung und der Verfügbarkeit eines Speicherplatzinhaltes vergeht.
Einschreiben:	Einbringen von Informationen in eine Speicherzelle unter Löschung der alten Information.
Lesen:	Entnahme eines Speicherplatzinhaltes ohne Löschung.
Wahlfreier Zugriff:	Alle Speicherplätze sind in beliebiger, ungebundener Reihenfolge erreichbar. (random access) Z.B. RAM, ROM
Reihenfolge Zugriff:	Z.B. Bindung an die Reihenfolge in der die Daten eingeschrieben wurden (Magnetband).
Permanenz:	Nach einem Energie- (Strom-) Ausfall bleibt die Information entweder erhalten (permanente Speicher: ROM, Magnetspeicher), oder ist gelöscht (temporäre Speicher: RAM, Register).

Anstelle der Begriffe permanenter- oder temporärer-Speicher werden häufig auch die Begriffe nichtflüchtiger- (nonvolatile) oder flüchtiger Speicher benutzt.

Ein Vergleich verschiedener Speichermedien (Abb.8.1) zeigt, daß die Kapazität und die Zugriffzeit von Speichern, in einem viele Zehnerpotenzen überstreichenden Bereich, variieren kann.

Das menschliche Gedächtnis ist zwar relativ langsam, dafür aber von großer Kapazität. Die in Abb.8.1 angegebene Kapazität entspricht etwa der Anzahl der Nervenzellen im Gehirn. Da der Speichermechanismus

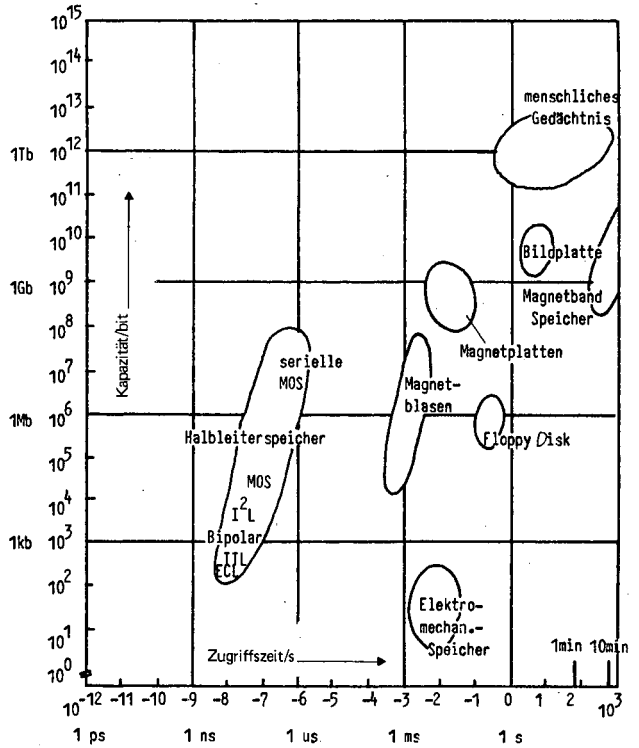


Abb. 8.1: Vergleich unterschiedlicher Speichermedien

jedoch noch nicht genau bekannt ist, könnte die Kapazität auch wesentlich größer sein (bis zu 10^{21} bit).

Die gebräuchlichsten internen Speicher der Mikrocomputer sind die Halbleiterspeicher. Ihre Zugriffszeit liegt etwa in dem Bereich, der für eine Anwendung ohne Wartezyklen des Mikroprozessors (Kap.7.2) nötig ist.

Als externe oder auch Massenspeicher, werden ausschließlich Magnetspeicher angewandt. Diese Speicher haben eine große Kapazität - wie schon der Name sagt -, jedoch eine lange bis mäßige Zugriffszeit.

Eine Zwischenstellung nehmen die neu entwickelten Magnetblasenspeicher ein. Diese permanenten Speicher besitzen eine große Kapazität und auch eine große Speicherdichte. Die Zugriffszeit ist kürzer als bei den anderen Magnetspeichern. Ihre Einsatzgebiete sind vorallem die Bereiche, wo Daten auch bei Stromausfall erhalten bleiben müssen.

Dieses Kapitel ist hauptsächlich den internen - und damit den Halbleiterspeichern gewidmet.

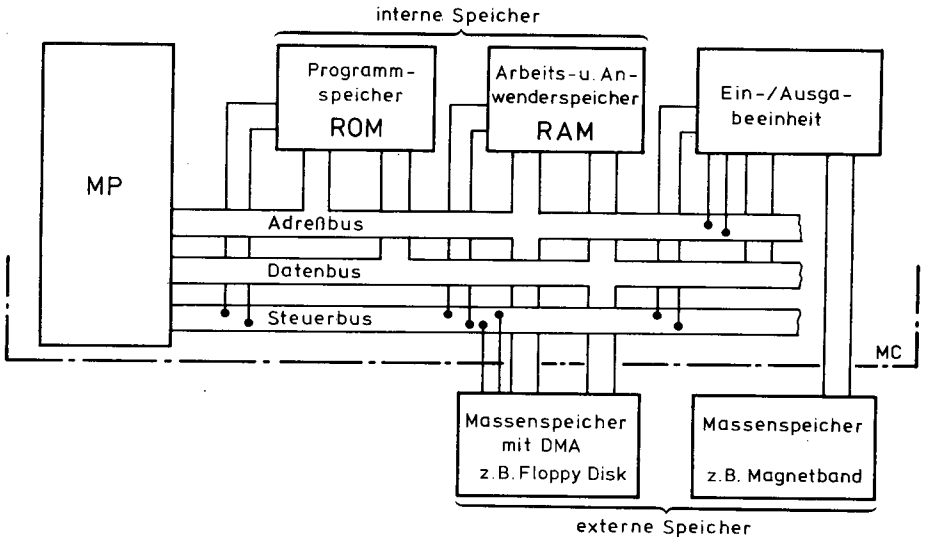


Abb. 8.2: Die Speicher eines Mikrocomputers (MC)

Wie Abb.8.2 zeigt, können die Massenspeicher entweder über eine Ein-/Ausgabe-Schnittstelle, oder unter direktem Zugriff auf die Busse (DMA) betrieben werden. Bei relativ langsamen Speichern (z.B. normale Magnetbandkassette) kann dieser an einen Ein-/Ausgabebaustein angeschlossen werden. Bei schnellen Speichern reicht die Datenübertragungsrate eines E/A-Bausteines nicht aus. Ein direkter Zugriff auf die Busse und damit auch auf die internen Speicher, erlaubt dagegen eine Übertragungsrate die nur durch die Zugriffszeit der internen Speicher begrenzt ist. In diesem Fall wird dann der Inhalt eines internen Speicherbereiches direkt in den Massenspeicher übertragen (kopiert). Der Mikroprozessor befindet sich während dieser Zeit im HOLD-Zustand und hat die Kontrolle über die Busse an die Steuerlogik des Massenspeichers abgegeben.

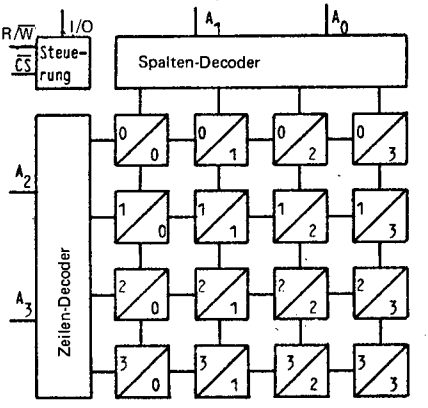
8.2 Speicherorganisation

Die internen Halbleiterspeicher eines Mikrocomputers besitzen einen wahlfreien Zugriff, d.h. jeder einzelne Speicherplatz ist direkt adressierbar.

Zur Bezeichnung der Größe und der Organisation eines solchen Speichers werden meistens die Anzahl der Speicherplätze und die Anzahl der Zellen pro Speicherplatz angegeben. Wobei die Anzahl der pro Speicherplatz parallel verarbeitbaren Bits (Speicherzellen) auch als Wortlänge bezeichnet wird.

Beispiele: 256x1 : 256 Speicherplätze mit jeweils einer Zelle.
1024x4 : 1024 Speicherplätze mit einer Wortlänge von 4 bit.
2048x8 : 2KB-Speicher mit einer Kapazität von 16384 bit = 16Kbit

In Abb.8.3 ist die innere Organisation eines 16x1 Speichers dargestellt.



Adresse A_3 A_2 A_1 A_0	HEX	Speicher Nr.
L L L L	0	0/0
L L L H	1	0/1
L L H L	2	0/2
L L H H	3	0/3
L H L L	4	1/0
L H L H	5	1/1
L H H L	6	1/2
L H H H	7	1/3
H L L L	8	2/0
H L L H	9	2/1
H L H L	A	2/2
H L H H	B	2/3
H H L L	C	3/0
H H L H	D	3/1
H H H L	E	3/2
H H H H	F	3/3

Abb. 8.3: Aufbau einer bitorientierten 16x1 Speichermatrix (R/W = Lesen oder Schreiben, I/O = Ein- oder Ausgabe, CS = Bausteinauswahl, A = Adreßleitungen)

Alle Speicherzellen sind in einer 4x4-Matrix angeordnet und über Spalten- (column) und Zeilen- (rown) Leitungen auswählbar. Es ist immer nur die Speicherzelle mit der Ein-/Ausgabeleitung (I/O) verbunden, die gleichzeitig über den Spalten- und den Zeilen-Decoder ausgewählt wird.

Die '1 aus 4 Decoder' für die Spalten- und Zeilen-Auswahl arbeiten in gleicher Weise wie der früher besprochenen '1 aus 8 Decoder' für die Bausteinauswahl (Abb.7.15). Zur Auswahl aller 16 Zellen sind 4 Adreßleitungen ($2^4 = 16$) erforderlich.

Ein Speicher, der 16 Plätze mit einer Wortlänge von 4 bit haben soll und damit 64 Zellen benötigt, kann ebenfalls als quadratische Matrix mit der Organisation 8x8 hergestellt werden. Wie Abb.8.4 zeigt, sind dann jeweils 4 Zellen einer Zeile zu einem Speicherplatz zusammenzufassen. Insbesondere die großen Speicher (1 - 2KB) haben häufig diese Organisation.

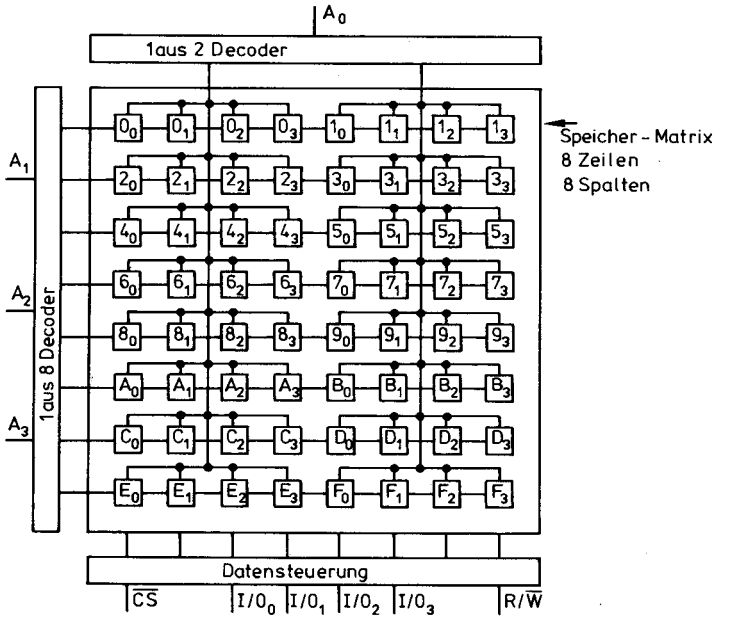


Abb. 8.4: Prinzipieller Aufbau eines 16x4 Schreib-/Lesespeichers mit quadratischer Matrix

Für die Spaltenauswahl sind entsprechend weniger Adreßleitungen als für die Zeilenauswahl erforderlich, da jeweils eine der Wortlänge entsprechende Anzahl von Zellen in einer Zeile gleichzeitig angesprochen und mit den I/O-Leitungen verbunden wird.

Beispiel: Bei 4 bit Wortlänge, wie in Abb.8.4, werden für die Spaltenauswahl 2 Adreßleitungen weniger benötigt, da zur Darstellung von 4 Adressen 2 bit ($4 = 2^2$) erforderlich sind.

Der Eingang CS dient zur Steuerung der Tristate-Gatter (Kap.6.3) in den Ein-/Ausgabeleitungen. Hierdurch ist ein direkter Anschluß an die Busse des Mikrocomputers möglich.

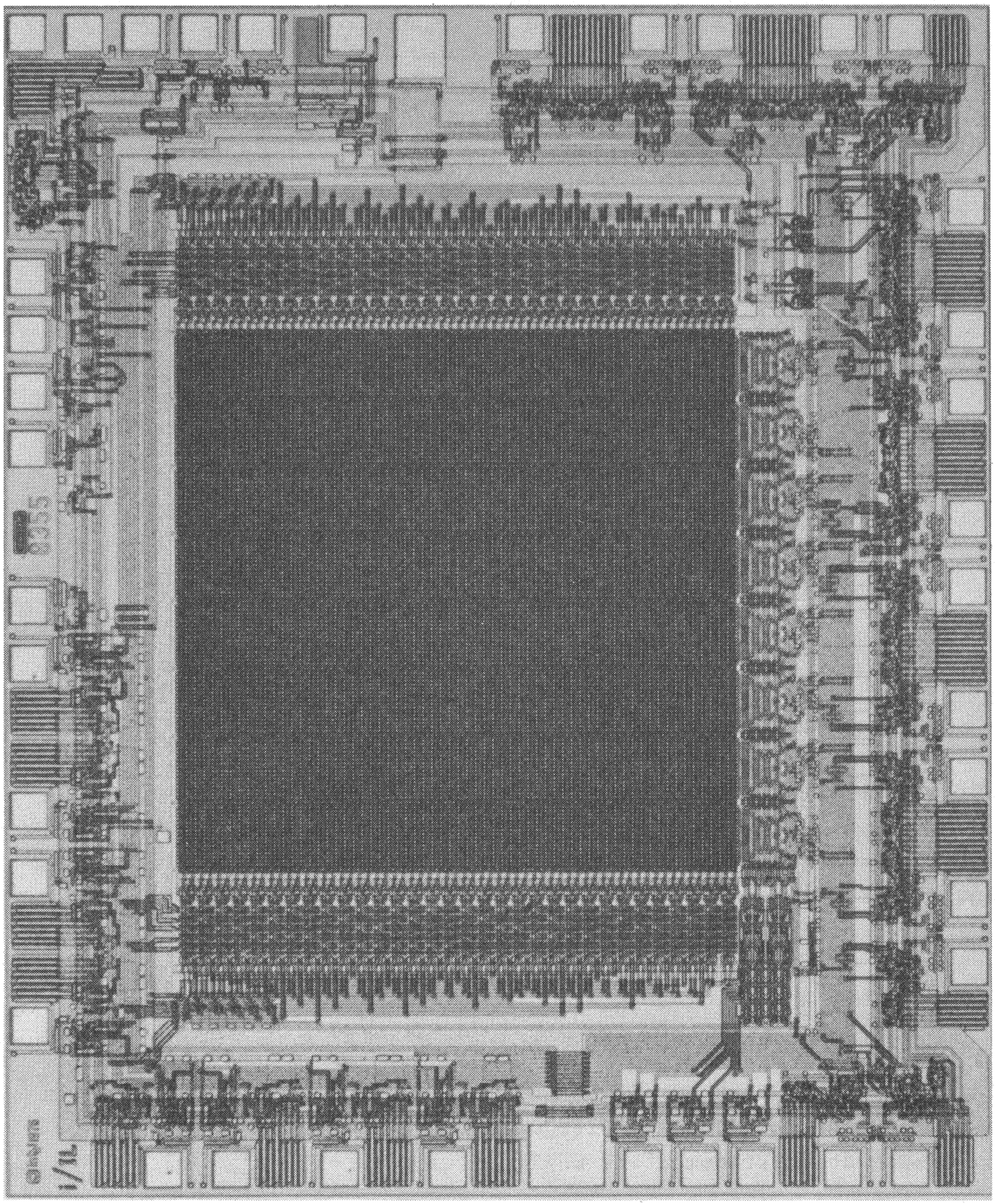


Abb.8.5: Dies Mikrofoto des Chips eines 8355-Bausteines zeigt deutlich den symmetrischen Aufbau der Speichermatrix.
(8355: 2048x8bit ROM + 2 x 8bit I/O Ports, direkt anschließbar an den AD-Bus des MP8085)

8.3 Schreib-Lese-Speicher (RAM)

RAM = Random Access Memory

Schreib-Lese-Speicher mit wahlfreiem Zugriff, für den Einsatz als interne Speicher eines Mikrocomputers, werden in den verschiedenen Variationen der MOS-Technik und auch in bipolarer Technik gefertigt. Die bipolaren Speicher mit ihren hohen Verlustleistungen werden jedoch kaum im Mikrocomputer eingesetzt, da für die hiermit erreichbaren kurzen Zugriffszeiten von unter 100ns zur Zeit kein Bedarf ist.

Typ	Bezeichnung	Organisation	Zugriffszeit/ns	Verlustleistung/mW standby/aktiv	Versorgungsspannung/V
statische TTL	74S189	16x4	35		+ 5
	82S16/17	256x1	40		+ 5
	2510	1024x1	35-70	/650	+ 5
statische ECL	10155	8x2	12		- 8
	10414	256x1	8-10	/800	- 5,2
	10470	4096x1	25		- 5,2
	100475	1024x4	25		- 5,2
statische MOS	2111	256x4	250-450	/150	+ 5
	2114	1024x4	100-450	/320	+ 5
	2141	4096x1	120-250	/320	+ 5
	2185	1024x8	200-300	/500	+ 5
	4118	1024x8	220	/400	+ 5
	4016	2048x8	150-450	/325	+ 5
statische CMOS	5101	256x4	450-800	0,01/110	+ 5
	74C929	1024x1	250		+ 5
	6508	1024x1	95-460	0,003/	+ 4 bis 11
	5115	1024x4	200-300	0,2/33	+ 4 bis 6
	6148	1024x4	70-85	/200	+ 5
	6116	2048x8	120-200	/160	+ 5
dynamische MOS	2104	4Kx1bit	150-300		+12,+5,-5
	2109	8Kx1bit	200-250	20/462	+12,+5,-5
	2118	16Kx1	100-150	11/150	+ 5
	4816	16Kx1	100	/250	+ 5
	4864	64Kx1	150-200	/170	+ 5
	4164	64Kx1	100-150		+ 5

Abb.8.6: Vergleich einiger Schreib-/Lese-Speicher mit wahlfreiem Zugriff (RAM). Standby bedeutet: Sparschaltung, ohne Zugriffe.

Statische Speicher

Bei statischen MOS-Speichern bestehen die einzelnen Speicherzellen aus zwei rückgekoppelten Invertern (bistabile Kippstufe, Flip-Flop). Den beiden stabilen Zuständen wird jeweils die logische Information L oder H zugeordnet. Die für diese Schaltung erforderlichen Lastwiderstände werden aus technologischen Gründen durch zwei Transistoren (T_3, T_4 in Abb.8.7) mit fester Vorspannung ersetzt. Ein Signal auf der Zeilenauswahlleitung (Abb.8.7) führt dazu, daß alle Speicherzellen, über die jeweiligen Transistoren T_5 und T_6, an die beiden

zugehörigen Spaltenleitungen angeschlossen werden. Die Transistoren T_7 und T_8 dienen der Spaltenauswahl.

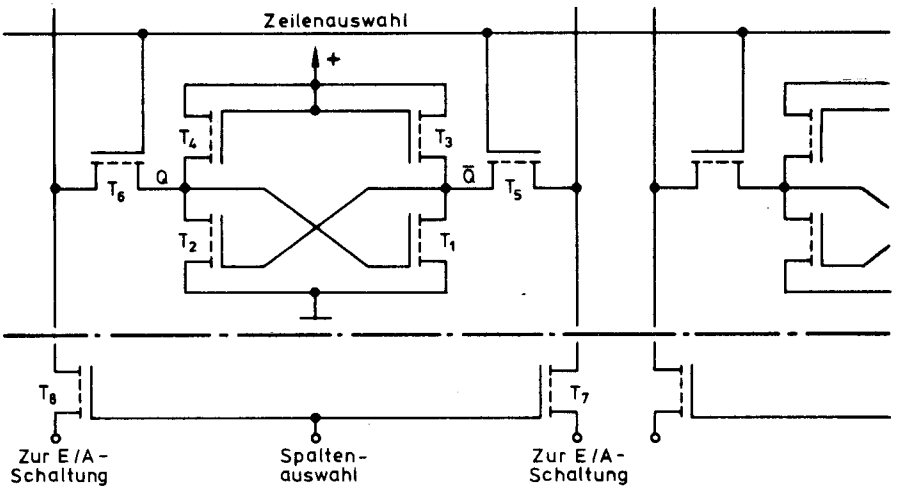


Abb. 8.7: Statische MOS-Speicherzelle mit Zeilen- und Spaltenauswahl

Die Schalttransistoren T_7 und T_8 sind nur einmal pro Spalte einer Speichermatrix vorhanden. Hierdurch werden aus der aktivierten Zeile die Zellen mit der Ein-/Ausgabeschaltung verbunden, die zu einem Speicherplatz bzw. einem Wort gehören.

Bei einer Speicherorganisation wie in Abb.8.4 (16x4) schalten immer 8 der 16 T_7/T_8-Transistoren durch und verbinden damit die 4 Zellen eines Speicherplatzes mit der E/A-Schaltung.

Zum Lesen des Inhaltes der ausgewählten Speicherzellen wird der Potentialunterschied zwischen den rechten und linken Spaltenleitung der einzelnen Zellen ausgewertet. Beim Schreibvorgang wird das Flip-Flop über das Spaltenleitungspaar gesetzt. Ein H auf der linken Spaltenleitung und ein L auf der rechten übergibt der Speicherzelle die Information H, falls das Flip-Flop über die Transistoren T_5 und T_6 an die beiden Spaltenleitungen angekoppelt wurde. Der einmal eingestellte Zustand bleibt solange erhalten, bis das Flip-Flop über einen erneuten Schreibzugriff verändert wird, oder die Stromversorgung aufhört.

Dies ist der Grund, weshalb Flip-Flop-Speicher 'statisch' genannt werden; sie benötigen keine Auffrischzyklen wie die dynamischen Speicher.

Als typisches Beispiel für einen RAM-Speicherbaustein, der häufig in Mikrocomputern eingesetzt wird, soll hier kurz auf den 2114 eingegangen werden. Wie Abb. 8.8 zeigt, besteht der 4096bit große Speicher aus einer 64x64 Matrix und hat die Organisation 1024 Worte x 4bit.

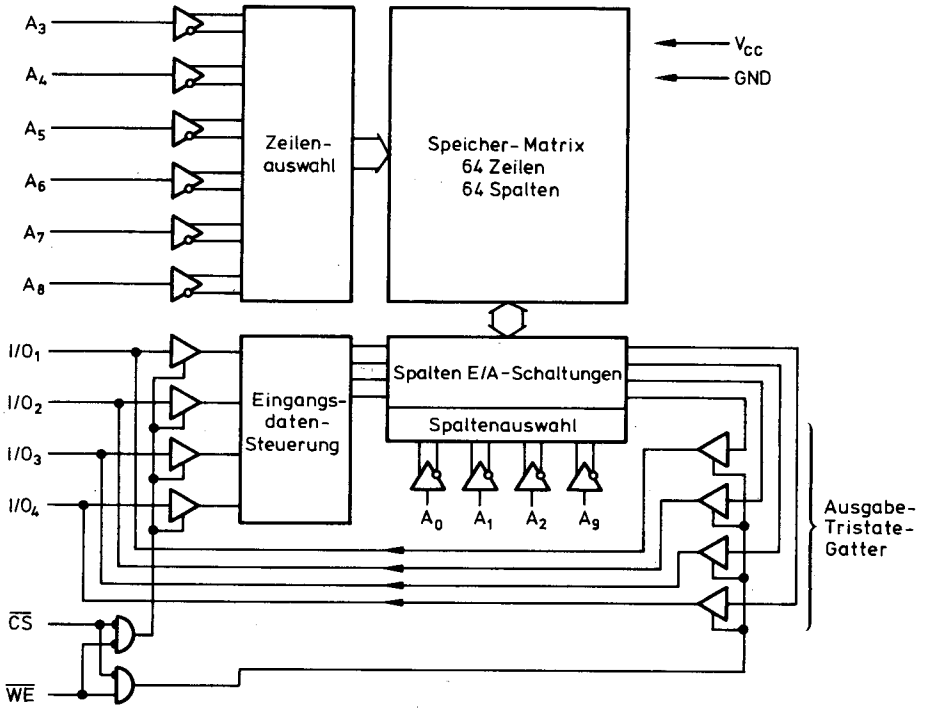


Abb. 8.8: Blockschaltbild des 1K x 4bit RAMs 2114

Durch die logische Verknüpfung der Signale CS und WE werden entweder die Eingangs- oder die Ausgangstreiber oder auch keiner von beiden (hochohmiger Zustand) aktiviert.

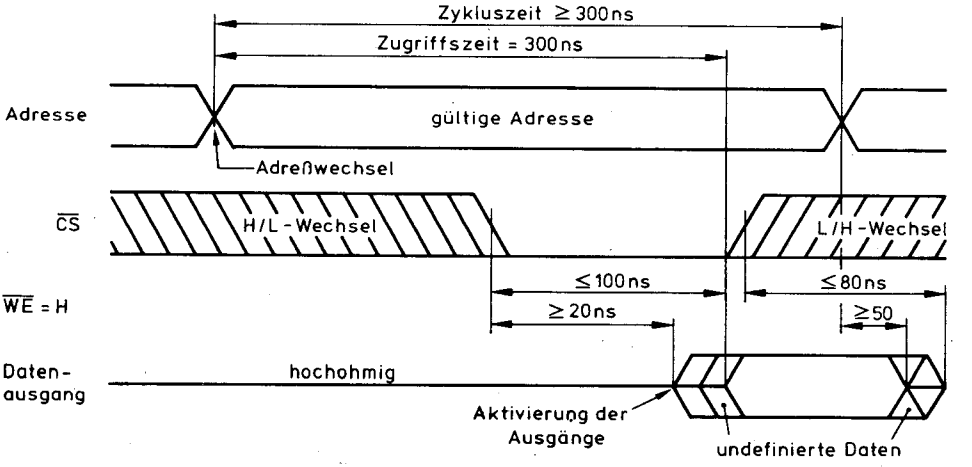


Abb. 8.9: Lesezyklus des RAM 2114 mit 300ns Zugriffszeit

So wie bereits beim MP 8085 geschildert, ist auch dieser Baustein erst ca. 0,5ms nach dem Einschalten der Versorgungsspannung ansprechbar, da zunächst über eine Ladungspumpe die Substratvorspannung erzeugt werden muß.

Wenn ein Lesezugriff auf den Speicher erfolgen soll, wird zunächst der Schreib-/Leseauswahleingang WE auf H gesetzt und die Adresse angelegt (Abb.8.9). Nach spätestens 200ns sollte dann das Bausteinauswahlsignal CS den L-Zustand annehmen. Daraufhin werden dann, 300ns nach dem Anlegen der Adresse, die Daten der vier adressierten Speicherzellen auf den Datenausgangsleitungen $I/O_{1...4}$ zur Verfügung stehen.

Wenn die Adresse geändert wird, verbleiben die Ausgabedaten noch wenigstens 50ns auf den I/O-Leitungen. In den hochohmigen Zustand geht der Ausgang, nachdem das CS-Signal einen L/H-Wechsel vorgenommen hat und zwar innerhalb von maximal 80ns.

Der Abb.8.9 kann man entnehmen, daß die meiste Zeit darauf verwandt wird die Speicherzellen über die Adresse auszuwählen und ihren Inhalt über die internen Datenleitungen zu den E/A-Schaltungen zu bringen (s.a. Abb.8.7).

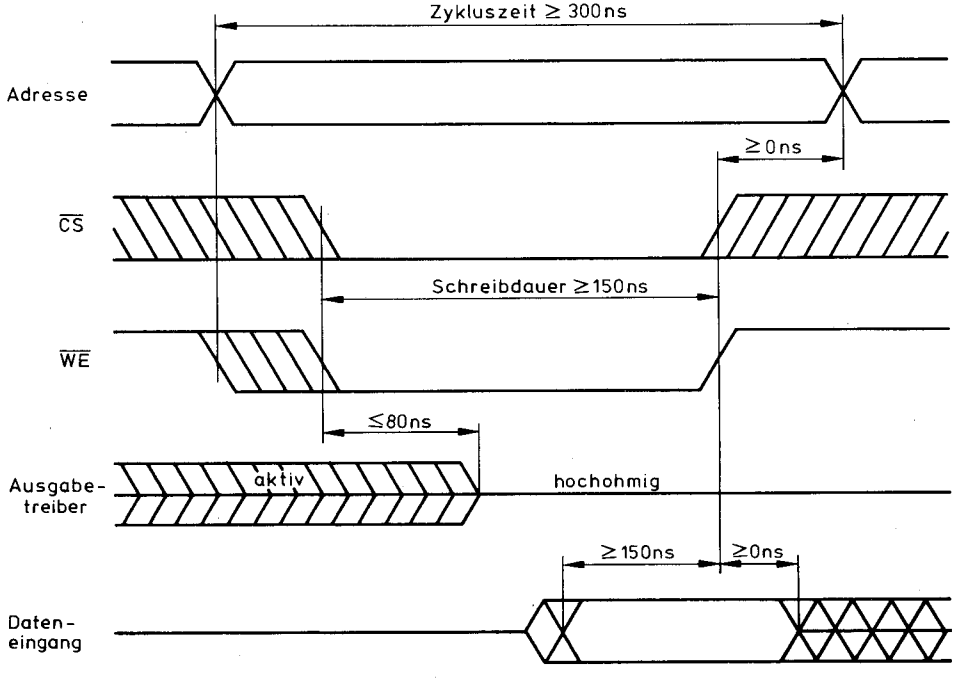


Abb. 8.10: Schreibzyklus des RAM 2114 mit 300ns Zugriffszeit

Die zeitliche Impulsfolge (timing) an den Eingängen, bei einem Schreibvorgang, ist in Abb.8.10 dargestellt.

Zum Einschreiben der Daten müssen diese wenigstens 150ns lang an den Eingängen anstehen. Während dieser Zeit muß der Baustein ausgewählt (CS) sein und der Schreib-/Leseauswahleingang (WE) den L-Zustand angenommen haben.

Bei einer Adressenänderung muß der Eingang WE immer ein H-Signal führen, da hierdurch ein unbeabsichtigtes Schreiben vermieden wird. Das Sperren der Ausgabetreiber durch WE = L dauert maximal 80ns (Abb.8.8 u. 8.10), wenn das Signal CS = L früher angelegen hat. Für den Fall, daß CS und WE gleichzeitig den L-Pegel annehmen, verbleiben die Ausgabetreiber im hochohmigen Zustand.

Beispielschaltung mit dem RAM 2114

In diesem Beispiel soll an das MP 8085-Grundsystem, gemäß Abb.7.16, eine Schreib-/Lese-Speicherplatine mit 16KB Kapazität angeschlossen werden. Hierzu benötigt man 32 RAM 2114-Bausteine, eine entsprechende Auswahllogik und Treiberbausteine. Alle Bausteine haben auf einer sog. Europakarte (100 x 160mm^2) Platz.

Bei der in Abb.8.11 dargestellten Schaltung wurden nur zwei der 32 RAM-Bausteine eingezeichnet. Die weiteren Speicherbausteine werden an die 15 noch freien Bausteinauswahlleitungen ($CS_{2...16}$) paarweise angeschlossen. Jeweils ein 2114 wird für die niederwertige und einer für die höherwertige Datentetrade benötigt; wobei jedoch beide an die selbe CS-Leitung anzuschließen sind, da sie gleichzeitig und zusammen den Datenbus bedienen.

Zur Verstärkung des Datenbusses wurde, in gleicher Weise wie im Grundsystem (Abb.7.16), der Baustein 74LS245 eingesetzt. Als Adreßbustreiber kommen 2 Bausteine 74LS367 zur Anwendung.

Die Erzeugung der 16 Bausteinauswahlsignale, wird durch einen '1 aus 16 Decoder' vorgenommen. Dieser Baustein arbeitet ähnlich wie der bereits früher beschriebene 8205 (Abb.7.15).

Der Auswahl- und die Treiberbausteine benötigen ihrerseits Signale, durch die sie aktiviert werden. Diese Aktivierung wird durch den '1 aus 4 Decoder' 74LS139 vorgenommen. Es ist hierdurch möglich, den 16KB-Speicherraum der Karte auf 4 verschiedene Adreßbereiche zu legen. Der Adreßbereich 0000 - 3FFF wird festgelegt, indem über den Schalter 1 die entsprechende Auswahlleitung mit den Auswahleingängen der Treiber und des '1 aus 16 Decoder' verbunden wird. Mit den Schaltern 2 bis 4 kann der Speicherkarte die Adreßbereiche 4000 - 7FFF bis C000 - FFFF zugewiesen werden.

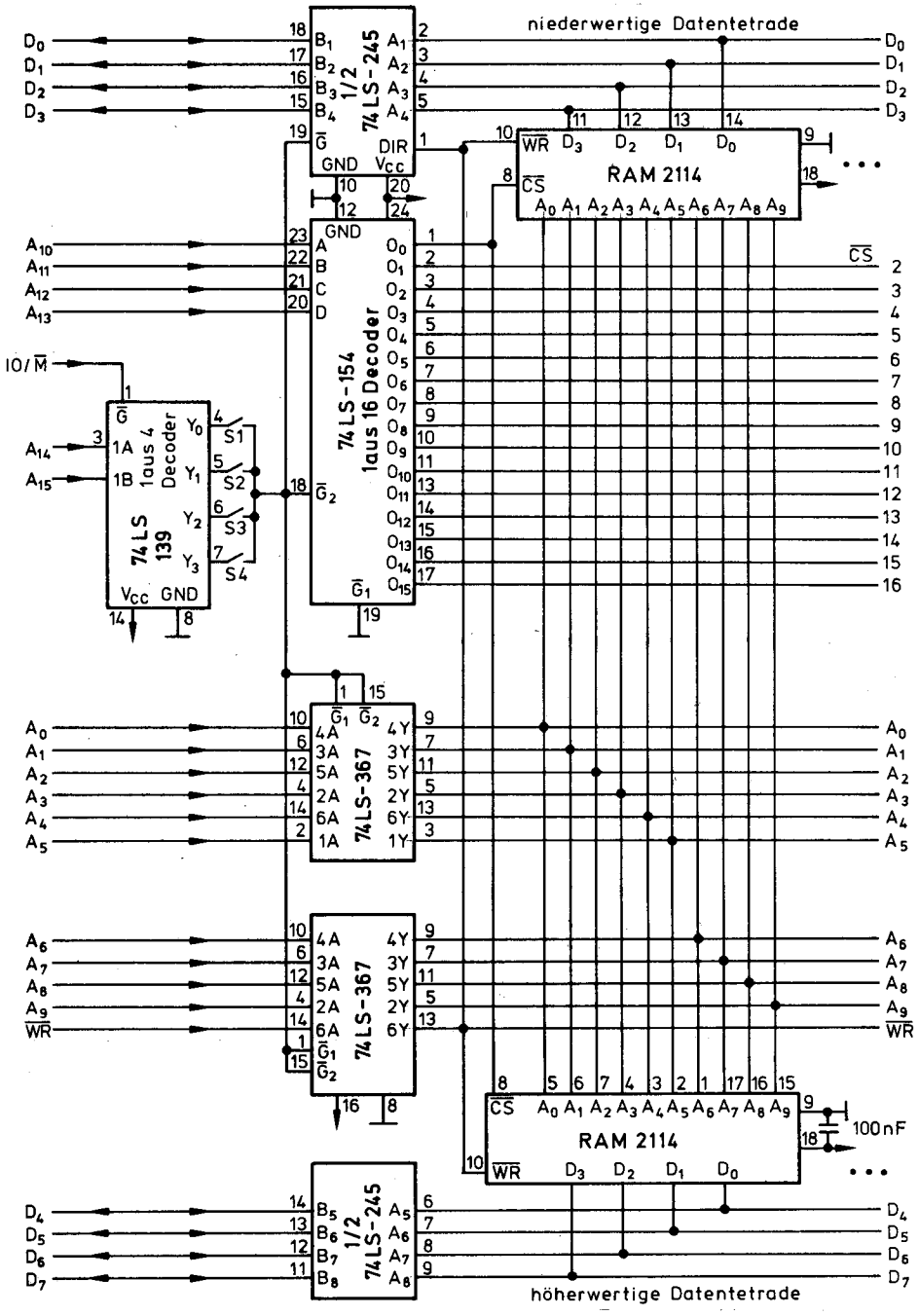


Abb. 8.11: 16KB-RAM-Speicher für folgende Adreßbereiche: S1: 0-3FFF, S2: 4000-7FFF, S3: 8000-BFFF, S4: C000-FFFF. ... bedeutet, zu weiteren Bausteinen.

CMOS-Speicher

Die Bedeutung der CMOS-Speicher liegt vorallem in ihrem geringen Leistungsbedarf begründet. Darüber hinaus behalten sie ihre Informationen auch noch, wenn die Versorgungsspannung drastisch reduziert wird (ca. 2V). In diesem sog. Standby-Betrieb, der häufig über eine Pufferbatterie bei Spannungsausfall aufgenommen wird, beträgt der Leistungsbedarf manchmal nur noch einige Mikrowatt.

Während früher die CMOS-RAMs eine geringere Kapazität pro Baustein und eine längere Zugriffszeit hatten (z.B. 5101 in Abb.8.6) wurde inzwischen die Herstellungstechnologie soweit verbessert, daß sie den MOS-Speichern hierin nicht mehr nachstehen.

Der Aufbau einer Speicherzelle gleicht der in Abb.8.7 für eine statische MOS-Zelle dargestellten Form. Im Unterschied zur MOS-Zelle werden bei der CMOS-Zelle jedoch immer zwei complementäre MOS-Transistoren (P- und N-Typ) gepaart benutzt.

Auch der organisatorische Aufbau des gesamten Speicherbausteines gleicht häufig dem der MOS-Speicher. So ist beispielsweise der Baustein 5115 (Ab..8.6) pinkompatibel zum 2114 und besitzt die gleiche innere Organisation, wie in Abb.8.8 dargestellt.

Die Sicherung der Daten in einem CMOS-Speicher über eine Pufferbatterie, wird im folgenden Abschnitt beschrieben.

Datensicherung bei Stromausfall

Ein großer Nachteil der Halbleiter-RAMs, ist der Datenverlust bei Stromausfall. Nach dem Wiedereinschalten der Versorgungsspannung nehmen alle Speicherzellen einen unterschiedlichen Zustand an, dieser ist bedingt durch zufällige Variationen in der Zellenstruktur, herrührend vom Herstellungsprozess.

Zur Milderung dieses Nachteiles werden im wesentlichen zwei verschiedene Methoden angewandt.

Datenverlagerung:
Bei der Methode der Datenverlagerung werden die zu sichernden Speicherinhalte in einen permanenten (nichtflüchtigen) Speicher gebracht wenn die Versorgungsspannung ausfällt.

Zur Realisierung dieser Methode ist vorallem ein Spannungsüberwacher und ein Energiespeicher erforderlich. Der Spannungsüberwacher muß bei einem Absinken der Versorgungsspannung ein Signal aussenden und der Energiespeicher die Versorgungsspannung noch so lange oberhalb des unteren Grenzwertes halten, bis die Daten gesichert sind. Bei dem Beispiel in Abb.8.12 bewirkt das Signal des Spannungsüberwachers einen TRAP-Interrupt des MP 8085.

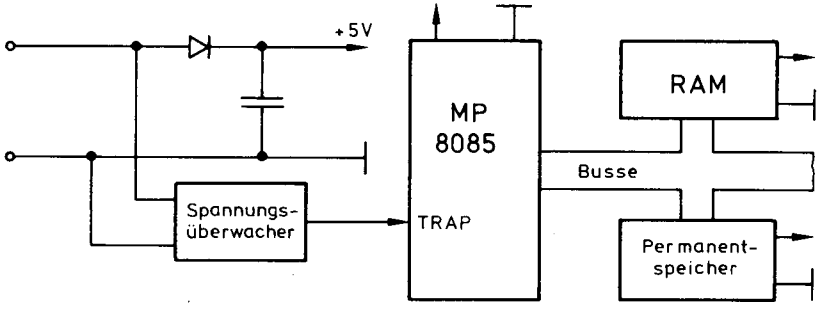


Abb.8.12: Beispiel für die Datensicherung nach der Verlagerungsmethode

Da dieser Interrupt nicht abschaltbar ist, wird er unbedingt durchgeführt (Kap.7.4). Das Interruptbedienprogramm hat dann die Aufgabe, innerhalb der Zeit in der die Versorgungsspannung noch anliegt, die zu sichernden Daten (RAM, Register, Flags) in den permanenten Speicher zu schreiben.

Als Energiespeicher kann beispielsweise ein Kondensator großer Kapazität herangezogen werden.

Der Permanentspeicher kann aus einem Magnetspeicher (Magnetblasenspeicher, Magnetplatte etc.) oder auch aus einem elektrisch programmierbaren ROM (Kap.8.4) bestehen.

Batteriepufferung:
Durch eine Batteriepufferung bei Stromausfall kann aus einem temporären Speicher ein quasi statischer Speicher gemacht werden. Insbesondere bei der Verwendung von CMOS-Speichern - mit ihrem geringen Leistungsbedarf (Abb.8.6) - lassen sich schon mit kleinen Batterien die Daten über lange Zeiten sichern (bis zu einigen Jahren; s.a. Übungsaufg.8.9).

Wie bereits in Kap.6.2 ausgeführt wurde, hängt der Leistungsbedarf von CMOS-Schaltkreisen stark von der Frequenz ab. Hieraus und aus der Möglichkeit die Versorgungsspannung im Standby-Betrieb stark zu reduzieren (auf ca. 2V) erklärt sich der geringe Leistungsbedarf der CMOS-Speicher.

Die Schaltung in Abb.8.13 stellt ein Beispiel für die Auslegung einer Stromversorgung mit Batteriepufferung dar.

Im Normalbetrieb erfolgt die Stromversorgung über ein nicht gezeichnetes Netzteil; beispielsweise das Netzteil des Mikrocomputers. Die Speicherbausteine und die Batterie sind über den Schalttransistor T1 mit diesem Netzteil verbunden. Als Verbindungselement wurde ein Transistor gewählt, damit ein möglichst geringer Spannungsabfall eintritt.

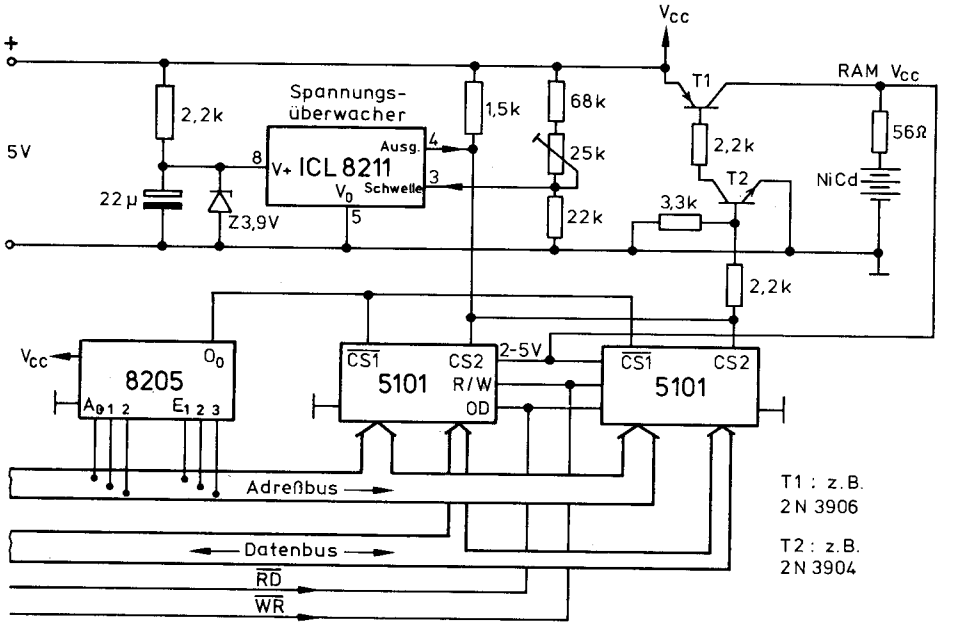


Abb. 8.13: Beispiel für die Notstromversorgung von RAM-Bausteinen

Dies ist wichtig, da sonst vom Mikroprozessor Signale kommen können, deren H-Pegel höher ist als die Versorgungsspannung der RAMs. In einem solchen Fall muß mit einer dauerhaften Schädigung der Bausteine gerechnet werden.

Eine Alternative hierzu besteht darin, die MP-Signale (Busse) über Treiber mit offenen Kollektor-Ausgängen an die Speicher weiterzugeben und dann die pull-up-Widerstände an die Versorgungsspannung der RAMs anzuschließen. In diesem Fall kann die Versorgungsspannung der RAMs reduziert werden, da eine Anpassung der den logischen Zuständen zugeordneten Spannungspegel erfolgt.

Die Batterie - hier werden häufig NiCd-Zellen eingesetzt - wird während des normalen Betriebes über einen Vorwiderstand dauernd geladen. Dieser Vorwiderstand muß so bemessen sein, daß der maximale Dauerladestrom der Batterie nicht überschritten wird (z.B. 1/10 der Batteriekapazität x 1 Stunde).

Der Spannungswächter-Baustein ICL 8211 kann über ein Potentiometer so eingestellt werden, daß bei einem Absinken der Versorgungsspannung unter beispielsweise 4,8V der Zustand des Ausganges 4 von H nach L wechselt. Dieser Wechsel wird dazu benutzt, die Bausteinauswahl 2 der RAMs zurück zu nehmen und den Transistor 1 zu sperren.

Der gesperrte Transistor verhindert ein Abfließen der Batterieenergie in die normale Stromversorgung.

Die sofortige Sperrung aller RAMs über CS2 verhindert unkontrollierte Schreibzugriffe beim Zusammenbrechen der normalen Versorgungsspannung.

Dynamische Speicher

Die Forderung nach Speicherbausteinen großer Kapazität, bei geringem Raum- und Leistungsbedarf, führte zur Entwicklung der dynamischen Speicher.

Eine dynamische Speicherzelle benötigt weniger Transistoren und kommt damit den genannten Forderungen entgegen. Nachteilig ist jedoch, daß die in einem Kondensator gespeicherte Information nur kurzzeitig erhalten bleibt und deshalb immer neu aufgefrischt werden muß (etwa alle 2ms).

Die einfachste Form einer dynamischen Speicherzelle ist die in Abb. 8.14 dargestellte 1-Transistor-Zelle.

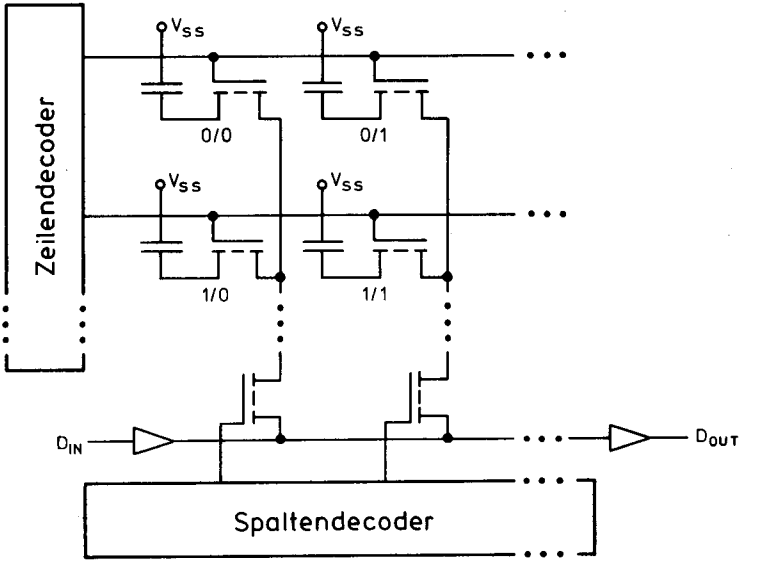


Abb.8.14: Bitorientierter dynamischer Speicher mit 1-Transistor-Speicherzellen

Zum Lesen einer Information werden die Transistoren einer Zeile durchgeschaltet und über den Spaltendecoder die Ladung des Kondensators der gewünschten Zelle auf den Ausgabeverstärker gebracht. Da infolge der geringen Kapazität des Kondensators einer Zelle nur ein geringer

Spannungshub auftritt, sind hier sehr empfindliche Leseverstärker erforderlich.

Das Lesen wirkt datenzerstörend, weil die Ladung des Kondensators abgezogen wird. Zur Auffrischung der Zelle (refresh) muß deshalb nach einem Lesevorgang der Inhalt zurückgeschrieben werden. Hierzu sind in dem Baustein entsprechende Schaltungen vorgesehen, die immer den gesamten Speicher auffrischen.

Wegen der unvermeidlichen Leckströme (Kondensatorentladung) muß die Speicherinformation periodisch regeneriert werden. Da das Abfließen der Kondensatorladung bei höheren Temperaturen beschleunigt wird, ist die Einhaltung der minimalen Refreshzeit (ca. 2ms) von großer Bedeutung. Der geringe Schaltungsaufwand für eine Speicherzelle ermöglicht den Aufbau großer Speicher pro Baustein.

Bausteine mit vielen Speicherplätzen benötigen eine große Adreßlänge. Dies würde zu großflächigen Bausteinen mit einer hohen Anzahl von Anschlußstiften führen. Bei dynamischen Speicherbausteinen, so wie in Abb.8.15, wird deshalb die Adresse meistens in zwei Hälften nacheinander übergeben.

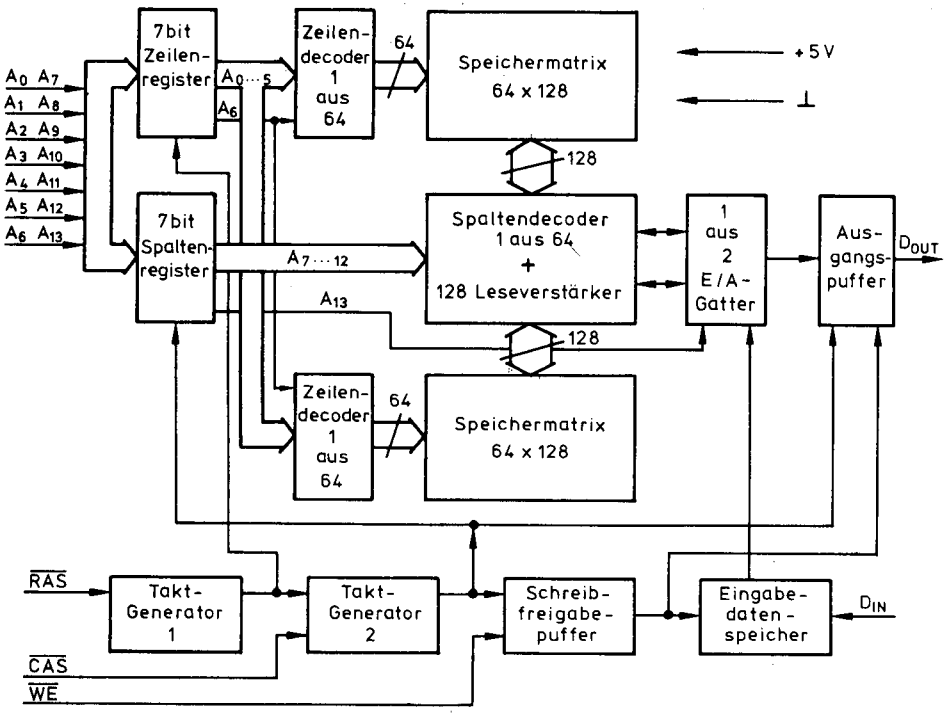


Abb.8.15: Blockschaltbild eines dynamischen Speichers mit der Organisation 16K x 1bit (2118)

Durch dieses Adreßmultiplexverfahren können bei einem 16K-Baustein 7 Anschlußstifte eingespart werden.

Wie Abb.8.15 zeigt, wird durch die Signale RAS (row address strobe) und CAS (column address strobe) die Übernahme der beiden Teiladressen, die zur Adressierung der Zeilen und Spalten erforderlich sind, gesteuert.

Beim Einsatz dynamischer Speicher in Mikrocomputern kann zwischen die Speicherbausteine und den Mikroprozessor ein spezieller Baustein eingefügt werden, der die erforderlichen Signale erzeugt.

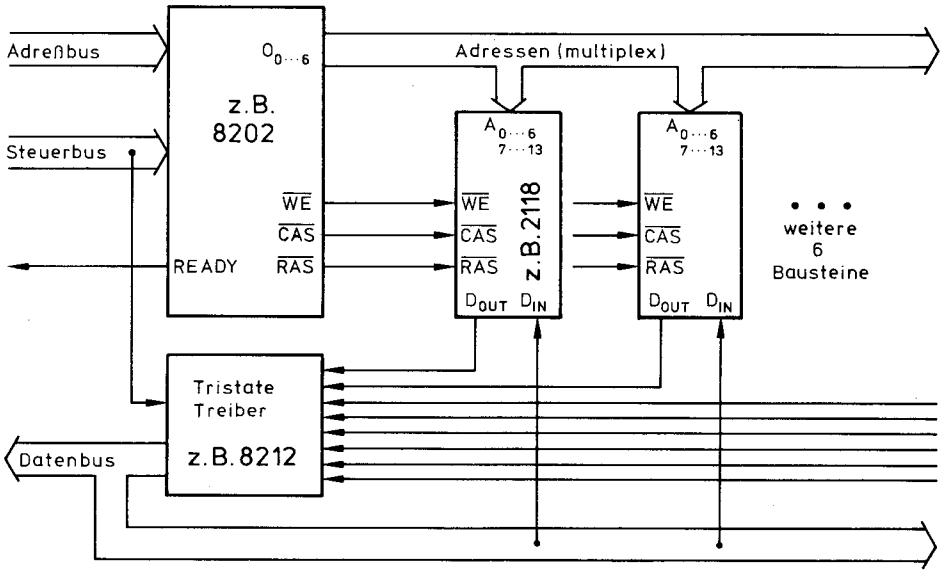


Abb. 8.16: Anschluß dynamischer Speicher an die Busse eines Mikroprozessors

8.4 Festwertspeicher (ROM)

ROM = Read Only Memory
Halbleiterspeicher, die nur einen wahlfreien Lesezugriff erlauben, werden als interne Programmspeicher für Mikrocomputer und zur Aufnahme von Tabellen benutzt. Hierbei ist von besonderer Bedeutung, daß der Speicherinhalt auch beim Ausfall der Versorgungsspannung nicht verloren geht (nonvolatile).

Da der Speicherinhalt vom Mikroprozessor nur ausgelesen werden kann, muß vorher - meist außerhalb des Mikrocomputers - der erforderliche Inhalt eingeschrieben werden. Dieses Einschreiben nennt man Programmieren.

Das Programmieren von ROMs geschieht je nach Typ (ROM, PROM, EPROM, EEPROM) auf sehr unterschiedliche Weise.

ROM: Programmierung durch entsprechende Masken bei der Herstellung.

PROM: Programmierbares ROM. Durch den Anwender einmal programmierbar.

EPROM: Lösch- (erasable) und programmierbares ROM. Sehr häufig programmierbar und durch UV-Licht löschbar.

EEPROM: Elektrisch lösch- und programmierbares ROM (= E^2PROM oder EEROM oder EAROM). Ungefähr 1000 mal lösch- und wieder programmierbar.

Die Organisationsform der Speicher entspricht denen der RAMs und wurde bereits in Kap.8.2 beschrieben. Lediglich die Eingabeschaltung entfällt hier und wird bei einigen Typen durch eine Programmierlogik ersetzt.

Die Speicherzellen bestehen aus einfachen Koppelelementen an den Kreuzungspunkten der Speichermatrix.

Maskenprogrammierbare ROMs

Bei einem maskenprogrammierbaren ROM wird der Inhalt als einer der letzten Produktionsschritte durch eine spezielle Maske erzeugt. Die Herstellung dieser Maske ist recht teuer (2500-5000DM, Stand 1980), wodurch diese Technik nur bei hohen Stückzahlen Anwendung findet.

Beispiel: 2Kx8bit-MOS ROM zur Aufnahme eines MP-Programmes für eine Maschinensteuerung:

1. Erstellung und Entwicklung des Programmes unter Zuhilfenahme löschbarer ROMs (z.B. EPROMs).
2. Das entgültige, für die Serienproduktion bestimmte Programm wird auf ein Massenspeichermedium (Lochstreifen oder Magnetspeicher) gebracht und an den Hersteller der ROMs gesandt.
3. Nach ca. 10 Wochen liefert der Hersteller ein Muster-ROM zur Prüfung des Programmes.
4. Wenn der Musterspeicher richtig ist, werden die restlichen ROMs nach weiteren ca. 10 Wochen geliefert, wobei von einer Mindestmenge von ca. 1000 Stück auszugehen ist.

Die Abb.8.17 zeigt eine in MOS-Technik hergestellte Transistormatrix. Durch die Maskenprogrammierung wird die Dicke der Isolationsschicht zwischen dem Kanal und dem Gate des MOS-Transistors variiert.

Ein Zeilenauswahlsignal gibt auf die Gates der Transistoren einer Zeile eine positive Spannung. Dies führt bei einer dünnen Isolations-

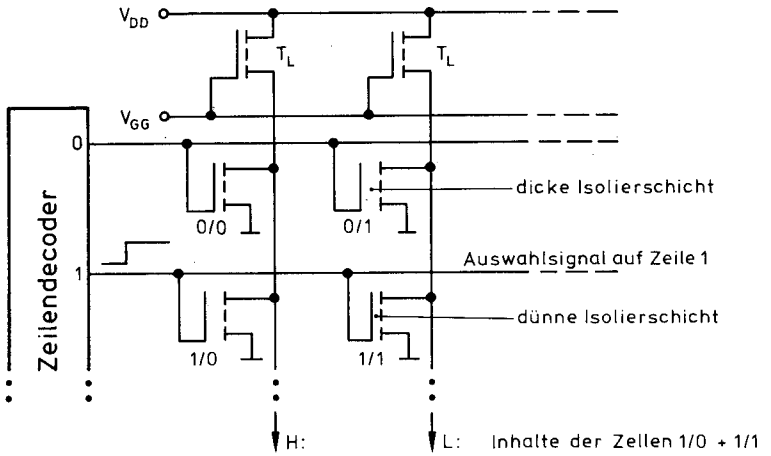


Abb. 8.17: MOS ROM-Speichermatrix mit Transistoren als Koppelelemente. Die Dicke der Gate-Isolierschicht wird durch eine spezielle Maske festgelegt (Programmierung).

schicht zu einem Durchschalten des Transistors und damit wird ein L auf der entsprechenden Datenleitung erzeugt.

Transistoren mit dicker Isolationsschicht verbleiben dagegen immer im gesperrten Zustand wodurch der H-Zustand auf der Datenleitung erhalten bleibt.

Die Erzeugung dieses H-Zustandes geschieht über Transistoren (T_L) mit fester Gate-Spannung, sie ersetzen hier die erforderlichen Lastwiderstände (pull-up-Widerstand).

Technologie	Typenbezeichn.	Organisation/bit	Zugriffszeit/ns	Verlustleistung/mW standby/aktiv	Versorgungsspannung/V	Pinkompatibel zu EPROM
Bipolar	6200 6250 81S290	256x4 1024x4 2048x8	45 60 80	 /900	+5 +5 +5	
MOS	2308 2616 2332 36000	1024x8 2048x8 4096x8 8192x8	450 450 450 350	/460 132/525 150/750 /225	+5;+12;-5 +5 +5 +5	2708 2716 2732 2764
CMOS	6312 65516 5332	1024x8 2048x8 4096x8	200 280 750	/2,5 0,015/0,6 4/	+5 +5 +5	 2732

Abb.8.18: Vergleich einiger maskenprogrammierbarer ROMs

Anwenderprogrammierbare ROMs (PROM)

Ein durch den Anwender programmierbarer Festwertspeicher wird als 'leerer' Baustein vom Hersteller bezogen. Leer bedeutet, daß alle Speicherzellen den selben Inhalt (H oder L) haben. Die Programmierung erfolgt beim Anwender mit Hilfe eines speziellen Gerätes und stellt einen nicht umkehrbaren (irreversiblen) Prozeß dar.

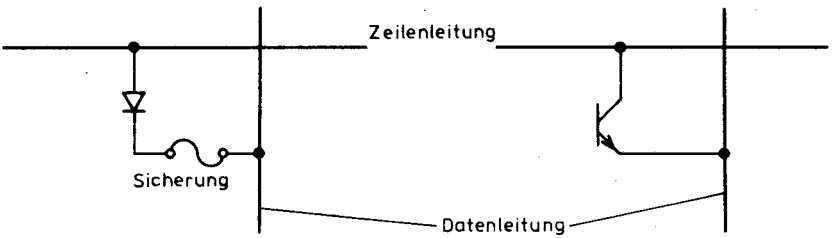


Abb.8.19: Zwei Beispiele für Koppelelemente in einem anwenderprogrammierbaren ROM

Bei der Programmierung wird das Koppelelement zwischen der Zeilen- und Spaltenleitung (ähnlich wie in Abb.8.17) entweder getrennt oder durchverbunden. Als Koppelelement kann beispielsweise eine Diode mit einer in Reihe geschalteten Sicherung (NiCr-fuse) oder ein Transistor benutzt werden.

Technologie	Typenbezeichn.	Organisation/bit	Zugriffszeit/ns	Verlustleistung/mW standby/aktiv
Bipolar	6300	256x4	55	
	5605	512x8	40	/750
	25089	1024x8	60	/600
	7616	2048x8	70	/500
CMOS	6611	256x4	450	05/50

Abb.8.20: Vergleich einiger anwenderprogrammierbarer ROMs (PROMs). Die Versorgungsspannung beträgt +5V.

Mit einem entsprechenden Stromimpuls wird beim Programmiervorgang entweder das Sicherungselement zerstört und damit die Verbindung zwischen der Zeilen- und Spaltenleitung unterbrochen, oder der Transistor wird dauerhaft leitend gemacht.
Ein Umprogrammieren ist hierdurch nicht mehr möglich.

Löschbare Festwertspeicher (EPROM, EEPROM)

Während der Entwicklung eines Programmes und einer Schaltung zur Serienreife, ist es von großem Nutzen Speicherbausteine zu haben, die lösch- und neuprogrammierbar sind. Darüberhinaus werden diese Speicher

auch bei Kleinserien und wenn eine Anpassungsflexibilität erhalten bleiben soll, eingesetzt.

Technologie	Typenbezeichnung	Organisation/bit	Zugriffszeit/ns	Verlustleistung/mW standby/aktiv	Versorgungsspannung/V	Pinkompatibel zu ROM
Löschung mit UV-Licht, EPROM						
MOS	2708	1024x8	350-450	/800	+5;+12;-5	2308
	+MS2716	2048x8	450	/525	+5;+12;-5	
	2716	2048x8	350-450	132/525	+5	2316/2616
	2732	4096x8	450-550	150/750	+5	2332/5332
	2764	8192x8	200	175/	+5	2364/36000
CMOS	6653	1024x4	450	07/30	+4,5÷10,5	
	6654	512x8	450	0,7/30	+4,5÷10,5	
Löschung durch elektrischen Strom, EEPROM						
MOS	1711	256x4	900		+5;-12	
	458	1024x8	500	/800		
	2808	1024x8	500		+5;+12V;-5V	2308
	48016	2048x8	350	/300	+5	2316/2616
	2816	2048x8	250-350	75/500	+5	2316/2616
CMOS	3008P	1024x8	500	0,5/12,5	+5	2308

Abb.8.21: Beispiele für löschbare PROMs (EPROM+EEPROM)

Zur Zeit gibt es zwei Arten von programmierbaren Halbleiterspeichern mit wahlfreiem Zugriff, die wiederholt lösch- und programmierbar sind. Sie unterscheiden sich in der Art des Löschverfahrens. Schon seit längerem im Einsatz sind die durch ultraviolettes Licht löschbaren EPROMs.

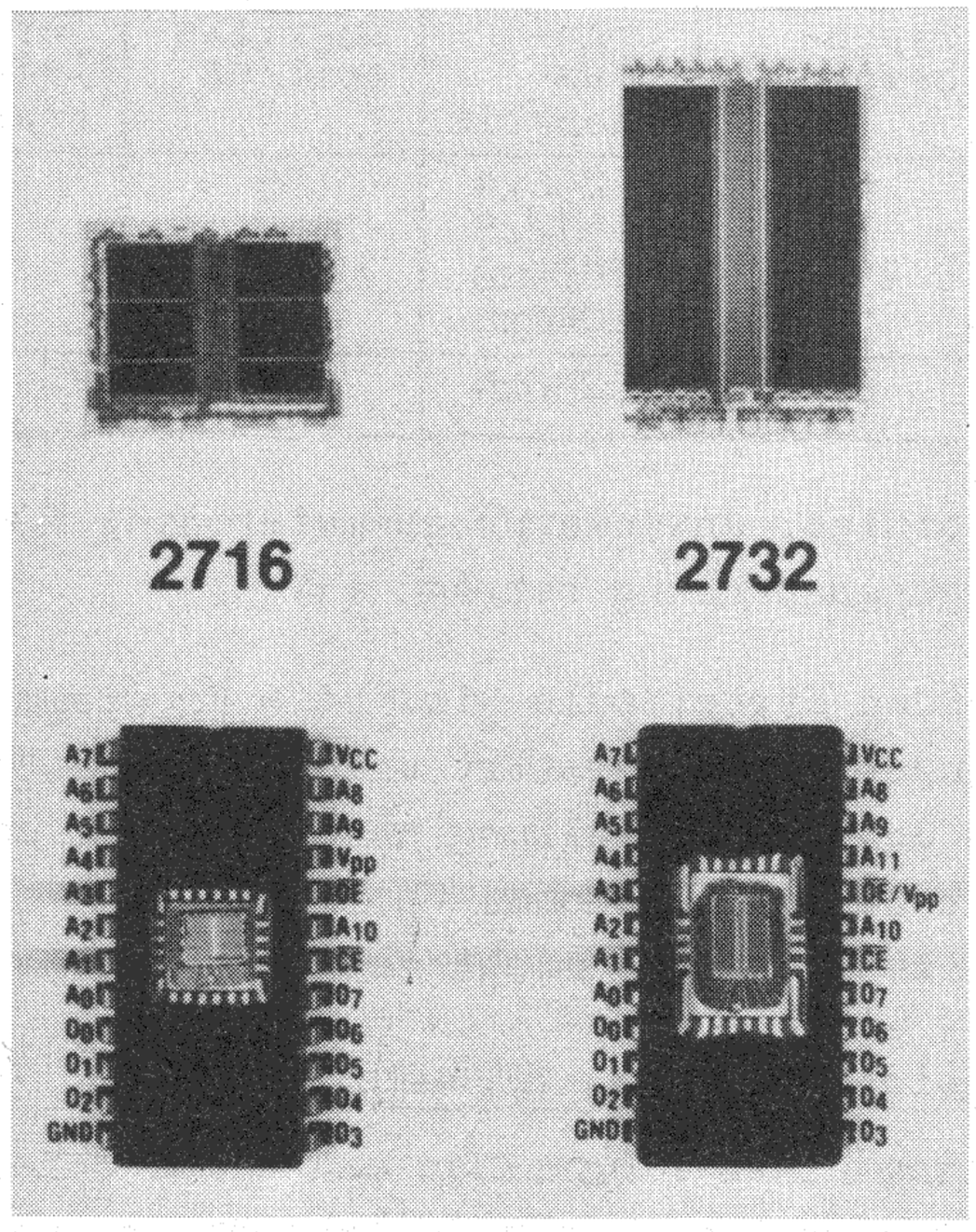


Abb. 8.22: Aufbau und Anschlußbelegung der EPROMs 2716 und 2732.

Die elektrisch löschbaren EEPROMs sind noch etwas jünger und haben bisher noch keine so große Verbreitung gefunden wie die EPROMs. Dies kann sich jedoch in naher Zukunft ändern, da die Entwicklung besserer EEPROMs rasche Fortschritte gemacht hat.

Durch UV-Licht löschbare Speicher (EPROM):
Ein durch UV-Licht löschbarer Speicher ist äußerlich erkennbar an einem Quarzfenster das in der Mitte des Bausteines den Chip abdeckt (Abb.8.22). Durch dieses Fenster wird beim Löschen (erase) ultraviolettes Licht auf die integrierte Schaltung gestrahlt. Da normales Glas nicht UV-durchlässig ist, wird für das Fenster Quarzglas (SiO_2) benutzt.

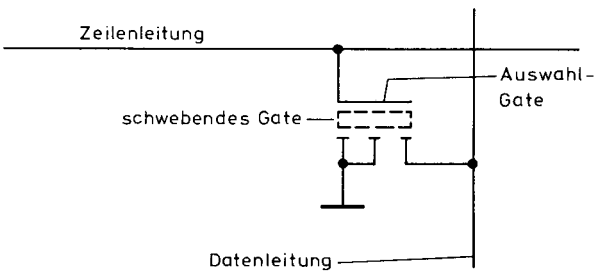


Abb.8.23: Durch UV-Licht löschbares FAMOS-Speicherelement (2716, NMOS-Technik)

Beim Programmieren wird der Speichertransistor mit einer großen Sperrspannung (z.B. 26V) bis zum Avalanchedurchbruch (Lawinendurchbruch) belastet. Hierbei gelingt es einigen Elektronen die Gate-Isolationsschicht zu durchdringen und diese aufzuladen. Dieses Gate ist ohne Anschluß nach außen und völlig isoliert eingebettet. Es schwebt in seinem Potential (floating gate, Abb.8.24).

Das elektrische Feld, des durch die Programmierung aufgeladenen Gates, erzeugt zwischen der Source- und der Draindiffusion einen leitenden Kanal (MOS-Feldeffekttransistor). Die Abkürzung FAMOS für diese Speicherzelle bedeutet: Floating Gate Avalanche Injection MOS.

Wenn der Halbleiter mit den energiereichen Quanten des UV-Lichtes bestrahlt wird, werden in der Isolationsschicht der Gates Ladungsträger freigesetzt und es kann ein Ausgleichsstrom fließen der die Gates in den Ausgangszustand versetzt. Auf diese Weise läßt sich der gesamte Baustein löschen.

Die Isolierung des Gates ist so gut, daß bei einer Umgebungstemperatur von 125°C, innerhalb von 10 Jahren, nicht mehr als 30% der Ladung abfließt.

Die technische Realisierung eines Gerätes zum Programmieren von EPROMs und die genauen Löschbedingungen werden in Kap. 11. beschrieben.

Das Kapitel 11 enthält die Beschreibung eines Mikrocomputers bei dem das EPROM 2716 als Programmspeicher benutzt wird.

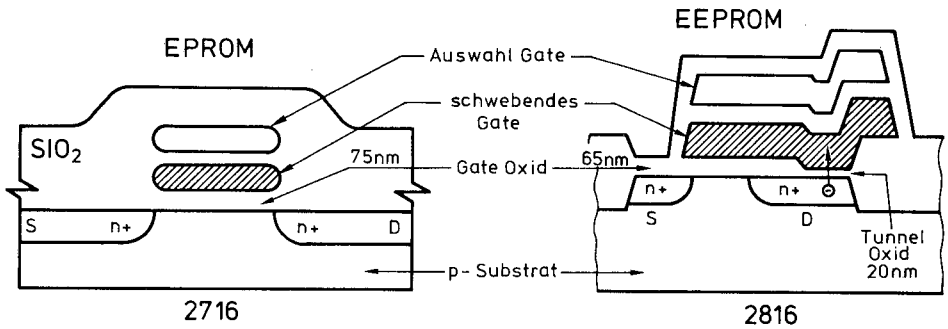


Abb. 8.24: Eine mit UV-Licht löschbare (EPROM) und eine elektrisch löschbare (EEPROM) Speicherzelle im Vergleich.

Elektrisch löschbare Speicher (EEPROM):

Eine elektrisch lösch- und programmierbare Speicherzelle ist ähnlich aufgebaut wie die eines EPROMs (Abb.8.24). Das schwebende Gate hat hierbei jedoch eine Stelle mit einer besonders dünnen Isolierschicht gegenüber der Drain-Diffusion. Durch diese dünne Isolierschicht können Elektronen in beiden Richtungen hindurch tunneln, jenachdem welche Richtung das elektrische Feld hat. Hierdurch ist das schwebende Gate auf- und entladbar; dies entspricht dem Vorgang der Programmierung und der Löschung.

Der bei diesem Vorgang ausgenutzte Tunneleffekt bedeutet, daß einige Elektronen die dünne Isolierschicht durchdringen indem sie, bildhaft ausgedrückt, einen Tunnel durch sie hindurch bohren. Die Quantenphysik erklärt diesen Vorgang mit dem Hinweis darauf, daß die von den Elektronen gebildete Materiewelle beim Auftreffen auf die Isolierschicht etwas in diese hineindringen kann und deshalb bei einer sehr dünnen Schicht diese zum Teil auch noch durchdringt.

In Abb.8.25 sind am Beispiel des EEPROMs 2816 die zum Löschen und Programmieren erforderlichen Spannungen bei einem 2-Transistor-Element angegeben.

Beim Löschen werden die Zeilenauswahl- und die Programmierleitung auf +21V gelegt, während die Datenleitung OV hat. Hierdurch können Elektronen in das schwebende Gate gelangen und es aufladen, denn das Programmier-Gate hat ein positives Potential.

Zum Programmieren wird die Feldrichtung umgekehrt und damit die Elektronen wieder aus dem schwebenden Gate herausgedrückt. Dieser Effekt wird durch ein Umpolen der Daten- und der Programmierleitung erreicht (Abb.8.25).

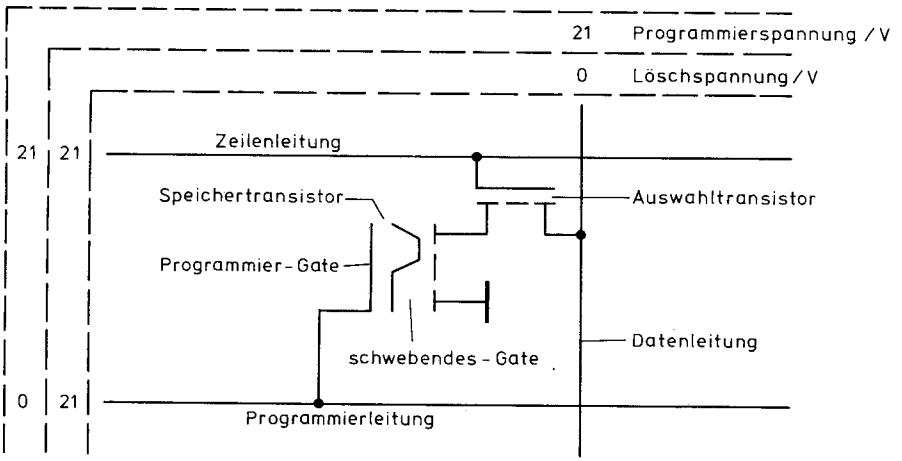


Abb.8.25: Elektrisch programmier- und löschbare Speicherzelle in 2-Transistor-Ausführung (2816)

Es werden immer gleichzeitig ganze Speicherplätze - entsprechend der Wortlänge (8bit beim 2816) - programmiert oder gelöscht. Insbesondere die Möglichkeit des selektiven Löschens und Programmierens durch einen elektrischen Strom, ist ein großer Vorteil gegenüber den EPROMs, so daß, bei sinkenden Preisen, für diese relativ neuen Bausteine mit einem großen Einsatzfeld zu rechnen ist.

8.5 Übungsaufgaben

8.1 Welche anderen Bezeichnungen werden für permanente und temporäre Speicher noch benutzt?

8.2 Wie nennt man große externe Speicher?

8.3 Welche Speicher werden als interne Speicher eines Mikrocomputers benutzt?

8.4 Wieviel Speicherzellen hat ein 2Kx8bit Speicher?

8.5 Wieviel Anschlußbeinchen braucht ein 1KB RAM-Speicher wenigstens?

8.6 Weshalb benötigt der RAM-Baustein in Abb.8.8 nur 4 Spaltenadressen obwohl die Speichermatrix 64 = 2^6 Spalten hat?

8.7 Aus welchem Speicherelement besteht eine statische und eine dynamische Zelle?

8.8 Ab wann und wielange ist der Inhalt eines Speicherplatzes auf dem Datenbus verfügbar, wenn der Mikroprozessor bei einem Speicherlesevorgang über ein READY-Signal 'angehalten' wird?

8.9 Wie lange können 4 RAM-Bausteine 5101 im standby-Betrieb durch eine Batterie mit 450mAh bei 2V betrieben werden?

8.10 Welcher praktische Unterschied besteht zwischen einem ROM und einem PROM und welche gemeinsamen Eigenschaften haben sie?

8.11 Wie wird bei einem löschbaren PROM die Information gespeichert?

9. Ein-/Ausgabe-Einheiten

Der Mikroprozessor - ergänzt durch Speicher - stellt eine autonome Rechen - und Steuereinheit dar. Diese Einheit ist jedoch praktisch solange wertlos, wie es keine Möglichkeit gibt einen Kontakt mit der Umwelt herzustellen.
Ein minimaler Kontakt würde beispielsweise darin bestehen, daß diese Einheit ständig Informationen an die Umwelt abgibt (z.B. Computer-Uhr). Darüberhinaus ist es jedoch meistens auch erforderlich, Informationen an den Mikroprozessor zu geben, damit auf seinen Programmablauf Einfluß genommen werden kann (z.B. Eingabe einer Weckzeit in eine Computer-Uhr).
Zur Kontaktaufnahme mit dem Mikroprozessor sind spezielle Ein-/Ausgabekanäle erforderlich. Fortschrittliche Mikroprozessoren verfügen manchmal selbst schon über einige solcher Kanäle (z.B. SID und SOD beim MP 8085).
Im allgemeinen werden jedoch weit mehr Kanäle benötigt und es ist deshalb erforderlich, spezielle Ein-/Ausgabe-Bausteine, als Bindeglieder zwischen dem Mikroprozessor und der Umwelt, einzufügen. Diese Bausteine haben die Aufgabe, Vermittler (interface) zwischen dem für den Datenaustausch zuständigen Datenbus und einer peripheren Einheit (Drucker, Anzeige, Tastatur, Maschine etc.) zu sein.

Damit der Datenbus einwandfrei arbeiten kann, müssen auch die Eingabe- und Ausgabe-Kanäle - ebenso wie die Speicher - über entsprechende Schalter (Tristate-Gatter) zum richtigen Zeitpunkt an den Datenbus angekoppelt werden.

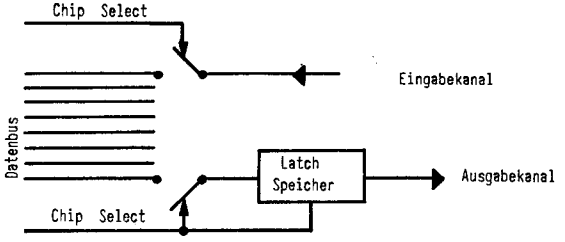


Abb.9.1: Ein-/Ausgabekanäle am Datenbus

Bei einem Ausgabe-Kanal wird meistens ein Zwischenspeicher vorgesehen, da die auszugebende Information auf dem Datenbus nur für sehr kurze Zeit (weniger als eine Mikrosekunde, Abb.7.7) erscheint. Am Ausgabe-Kanal bleibt dann die Information solange erhalten, bis sie durch einen erneuten Ausgabevorgang überschrieben wird.

Für eine serielle Datenübertragung benötigt man nur einzelne Kanäle, da die Information Bit für Bit übertragen wird. Diese Form des Informationsaustausches ist relativ langsam und findet vorallem bei langen Übertragungswegen Anwendung, denn dies führt zur Einsparung von Leitungen. Eingesetzt wird sie in der allgemeinen Datenverarbeitung, zum Anschluß von Druckern, langsamen Massenspeichern, Bildschirmen, zur Kopplung von Rechnern über das Telefonnetz etc.

Bei der parallelen Datenübertragung wird auf einer Anzahl von Leitungen - die der Wortlänge entspricht - immer ein gesamtes Wort gleichzeitig übertragen. Der Datenbus eines Mikrocomputers stellt beispielsweise ein solches Übertragungssystem dar, denn dort übertragen die Bausteine immer ein Wort (z.B. 1 Byte) gleichzeitig (parallel). Bei nicht zu weit entfernten Geräten ist, zur Erreichung einer hohen Datenübertragungsrate, ein paralleler Datentransport vorteilhafter.

Da bei einer längeren Information Wort für Wort hintereinander übertragen wird, spricht man in diesem Fall auch von einer wort- (oder auch byte-) seriellen-Datenübertragung. Demgegenüber wird dann die zuerst dagestellte serielle Datenübertragung als bitserielle-Datenübertragung bezeichnet.

Die weiteren Kapitel enthalten Beispiele für beide Formen der Datenübertragung.

Neben der reinen Datenübertragung sind jedoch - und dies insbesondere in der Mikrocomputertechnik - Eingabe- und Ausgabe-Kanäle zu Meßzwecken und für die Steuerung von Maschinen und Geräten erforderlich.

9.1 Parallele Ein-/Ausgabe

Zur Ein- oder Ausgabe eines 8bit-Wortes reicht es, zwischen das periphere Gerät und den Datenbus, einen Baustein einzufügen, der elektronische Schalter (Tristate) und evtl. Speicher für 8bit, gemäß Abb.9.1, enthält.

Eingabe-Schaltung

In Abb.9.2 ist ein Beispiel für die Ankopplung einer Tastatur an den Datenbus dargestellt. Es sind lediglich Tristate-Gatter zwischen die Tastaturausgänge und den Datenbus zu schalten. Eine Bausteinauswahllogik läßt sich sehr leicht über ein NAND-Gatter realisieren. Immer wenn A_0, IO/M und WR den Zustand H annehmen, wird die Tastatur an den Datenbus geschaltet und der ASCII-Code der gedrückten Taste übertragen.

```
Beispiel:   IN   01H ; A0=H,IO/M=H,RD=L,WR=H
            CPI  41H ; Wurde die Taste 'A' gedrückt?
```

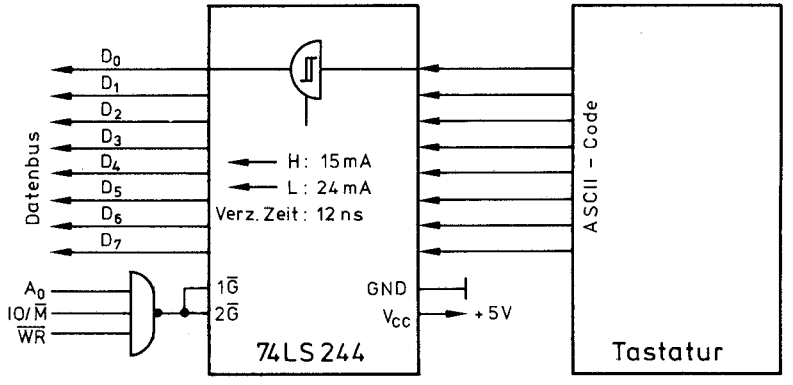


Abb.9.2: Eingabe-Schaltung für eine Tastatur mit einer Bausteinauswahllogik für den MP8085

Nur beim IN- oder OUT-Befehl nimmt der Ausgang IO/M des MP 8085 den Zustand H an. Die 1 Byte-Adresse erscheint auf dem unteren und dem oberen Byte des Adreßbusses gleichzeitig. Der Befehl IN 01 erzeugt somit die Adresse 0101H.
Für die Bausteinauswahl in Abb.9.2 kann deshalb auch anstelle der Adreßleitung A_0 die Leitung A_8 benutzt werden.

Ausgabe-Schaltung

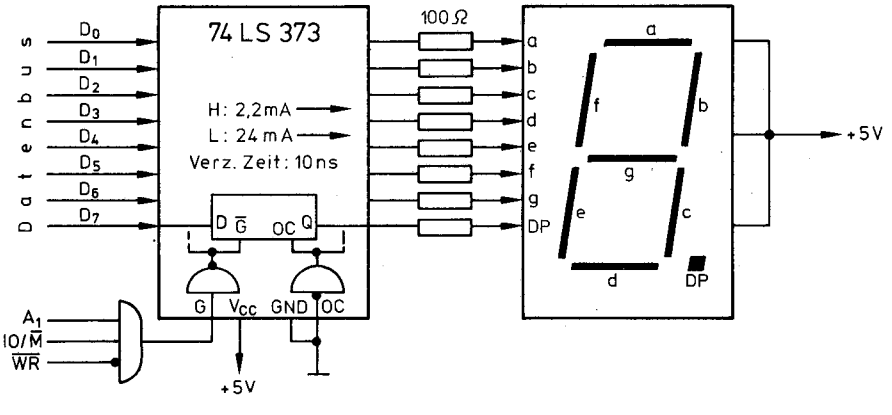


Abb.9.3: Ausgabe-Schaltung für eine Siebensegment-Anzeige zum Anschluß an den Datenbus des MP8085

Wie in Abb.9.1 angedeutet, enthält eine Ausgabe-Schaltung meistens einen Zwischenspeicher, da die logischen Zustände auf dem Datenbus sehr schnell wechseln.

Der in Abb.9.3 benutzte Baustein wurde bereits früher als Adreßbustreiber mit Zwischenspeicherung (Abb.7.12 u. 7.16) eingesetzt. Das

Beispiel in Abb.9.3 zeigt eine Schaltung zur Ausgabe von Zahlen auf einer sog. Siebensegment-Anzeige. Die zwischengeschalteten Widerstände begrenzen die Leuchtdiodenströme. Immer wenn ein L ausgegeben wird, leuchtet das entsprechende Segment auf und es lassen sich auf diese Weise verschiedene Zeichen darstellen (Kap.10.2).

Die Bausteinauswahllogik ist hier besonders einfach, da keine Verriegelung gegen ein unbeabsichtigtes Lesen zu erfolgen braucht. Wenn anstelle eines OUT- ein IN-Befehl gegeben wird, liest der Mikroprozessor nur einen unbestimmten Zustand oder bei TTL-Bausteinen den Wert FF ein. Gleichzeitig gehen jedoch auf der Anzeige alle Leuchtdioden aus, da der Wert FF auch in den Zwischenspeicher des Ausgabebausteines gelangt.

```
Beispiel:   MVI  A,49H ; LHLLHLLH≙49H
            OUT  2     ; A1=H,IO/M=H, Auf der Anzeige erscheint eine 5
            IN   2     ; Fehler, die Anzeige erlischt
```

Ein-/Ausgabe-Baustein 8255

Eine größere Flexibilität und Leistungsfähigkeit läßt sich durch die Verwendung spezieller, hochintegrierter und allgemein verwendbarer Ein-/Ausgabe-Bausteine erreichen.

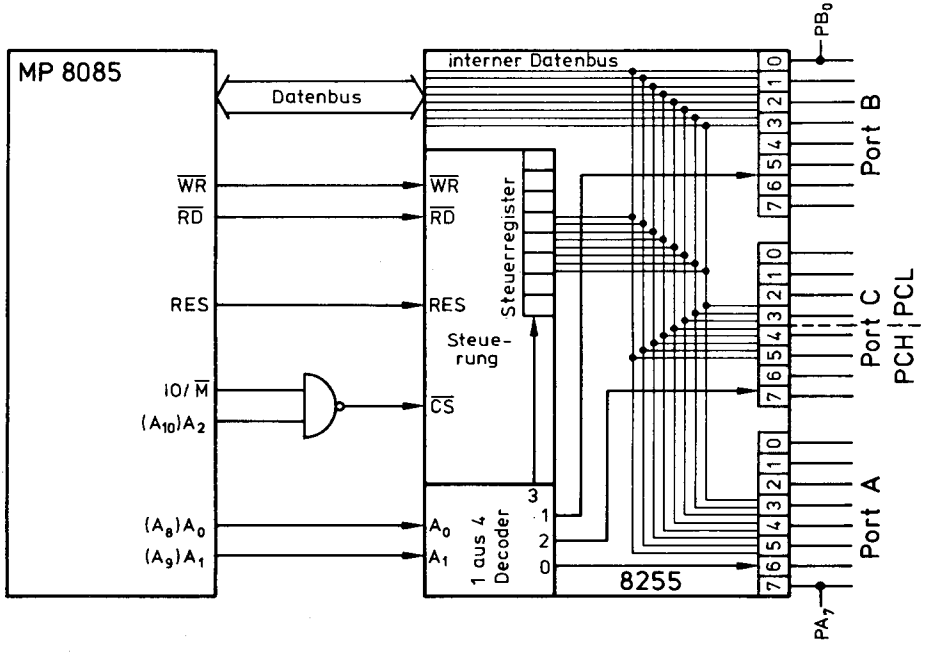


Abb.9.4: Anschluß des programmierbaren Port-Bausteines 8255 an den MP8085

Solche Bausteine enthalten meistens mehrere, frei programmierbare, Schnittstellen zum Datenbus. Jede Schnittstelle besteht aus 8 Kanälen - entsprechend den 8bits des Datenbusses - und wird häufig als 'Port' bezeichnet.
In Abb.9.4 ist ein, zur MP 8080-Familie gehörender, Port-Baustein dargestellt. Zum Anschluß an den MP 8085 sind neben dem Datenbus noch 4 Steuerleitungen und im einfachsten Fall 3 Adreßleitungen erforderlich.
Die Steuersignale WR und RD (write/read) dienen zur Festlegung der Datenflußrichtung auf dem internen Datenbus.
Das Reset-Signal bewirkt eine Programmierung der drei Ports A, B, C auf Eingabe. Hierdurch wird erreicht, daß kurz nach dem Einschalten des Mikroprozessors - bevor der Port-Baustein programmiert werden kann - keine undefinierten Ausgaben auf einem Port erfolgen können. Dies könnte sonst zu schweren Störungen oder gar zu Zerstörungen an peripheren Geräten führen. Die Bausteinauswahl (CS) kann in ähnlicher Weise wie beim Ausgabe-Baustein in Abb.9.3 erfolgen. Beachtet werden muß hierbei jedoch, daß die Adreßbits A_0 und A_1 für eine andere Aufgabe reserviert sind.

CS	WR	RD	RES	
X	X	X	H	Programmierung auf Eingabe
H	X	X	L	Baustein vom
L	H	H	L	Datenbus getrennt
L	H	L	L	Lesen, Port ---▸ MP, Eingabe
L	L	H	L	Schreiben, MP ---▸ Port, Ausgabe

A_0	L	H	L	H
A_1	L	L	H	H
Auswahl	PA	PB	PC	SR

Abb. 9.5: Steuersignale des 8255 und Auswahl der internen Register. PA= Port A, PB= Port B, PC= Port C, SR= Steuerregister.

Die Adreßbits A_0 und A_1 dienen - über einen internen '1 aus 4 Decoder' - zur Auswahl der im Baustein 8255 vorhandenen Register. Jenachdem, welche zweistellige Binärzahl über A_0, A_1 anliegt, wird eines der 4 Register mit dem internen Datenbus verbunden. Die rechte Tabelle in Abb.9.5 zeigt die entsprechende Zuordnung.

Das Steuerregister (SR) nimmt ein Datenwort auf, dessen einzelne Bits den Betriebszustand des 8255 festlegen. Die drei Port (PA, PB, PC) enthalten jeweils einen Speicher (Register) zur Sicherung von Ausgabedaten.

Wie schon erwähnt, bringt ein Reset-Signal alle drei Ports in den Eingabe-Zustand. Nach dem Einschalten des Mikroprozessors erzeugt im allgemeinen eine spezielle Schaltung automatisch ein Reset-Signal. Hierdurch sind alle Ports zunächst auf Eingabe programmiert.

Die Festlegung der Aufgabe einzelner Kanäle oder ganzer Ports - bedingt durch den konkreten Einsatz des Mikrocomputers - erfolgt durch die ersten Befehle des Mikrocomputerprogrammes, welches ja nach einem Reset, mit der Adresse 0 (MP 8080/85) beginnend, abgearbeitet wird.

```
Beispiel:   MVI  A,8AH ; Steuerwort in den Akku bringen
            OUT  7     ; Steuerwort ins Steuerregister schreiben
```

Die Aussendung der Adresse 0707H, im obigen Beispiel, erzeugt ein Bausteinauswahlsignal für den 8255 in Abb.9.4, denn IO/M und A_2 haben den Zustand H (7≙LHHH). Da auch die Adreßbits A_0 und A_1 H sind, wird im 8255 das Steuerregister mit dem Datenbus verbunden und es nimmt den Wert 8A auf. Nun ist der Port-Baustein 8255 programmiert.

Das im obigen Beispiel benutzte Steuerwort 8A programmiert Port A, sowie die niederwertige Tetrade des Ports C (PCL), auf Ausgabe und Port B, sowie die höherwertige Tetrade des Ports C (PCH), auf Eingabe.

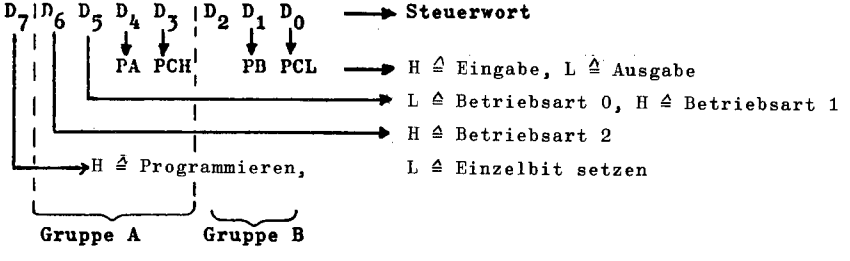


Abb. 9.6: Steuerwort für den 8255

Die Bedeutung der einzelnen Bits des Steuerwortes wird in Abb.9.6 erklärt. Das im letzten Beispiel angewandte Steuerwort 8A entspricht der Binärzahl HLLLHLHL. Dies sagt aus, daß eine Programmierung erfolgen soll (D_7=H), daß die Betriebsart 0 gewünscht wird und legt weiterhin die schon beschriebenen Datenflußrichtungen der einzelnen Ports fest.

Die Gruppen A und B können auf verschiedene Betriebsarten programmiert werden (siehe Datenbücher). Bei der schon beschriebenen Betriebsart 0 sind die Ports als einfache Ein-/Ausgabe-Schnittstellen zu benutzen. Die Betriebsart 1 gestattet eine getastete Ein-/Ausgabe, hierbei werden die Ports A oder B in Verbindung mit Abtastimpulsen oder Quittungs-Signalen des Ports C benutzt. Eine getastete Zweiweg-Bus-Ein-/Ausgabe ist schließlich in der Betriebsart 2 möglich.

```
Beispiele:  MVI  A,8AH ; Steuerwort (siehe oben)
            OUT  7     ; Steuerregister gemäß Abb.9.4 u. 9.5
            IN   5     ; Die momentan an Port B anstehenden 8 Signale
                       ; PB0...7 werden in den Akku gelesen.

            MVI  A,0   ; Ausgabewort
            OUT  4     ; Der Ausgang Nr. 4 (PA0...7) wird auf 8xL
                       ; gesetzt.
            IN   4     ; Es wird das Register des Ausganges 4 gelesen
                       ; und es erscheint im Akku das zuletzt ausge-
                       ; gebene Wort (hier 00).
```

Die Adressen der Ports des Bausteines in Abb.9.4 - sie wurden im obigen Beispiel benutzt - ergeben sich aus Abb.9.5 zu: Port A=04, Port B=05, Port C=06.

Diese Adressen sind allerdings nicht eindeutig, denn auch jede andere Adresse eines IN- oder OUT-Befehles, bei der A_2=H ist, selektiert den Baustein in Abb.9.4. Wenn mehrere 8255 an den Datenbus anzuschließen sind, kann jeweils eines der Adreßbits A_2 bis A_7 die Bausteinauswahl bewirken; auf diese Weise können 6 Port-Bausteine 8255 gleichzeitig benutzt werden. In diesem Fall ist jedoch darauf zu achten, daß immer nur eines der Adreßbits $A_{2...7}$ den Zustand H annimmt, da sich sonst mehrere Ports um den Datenbus 'streiten'.

Unter Verwendung von Adreßdecodern (z.B. 1 aus 8 Decoder, Abb.9.8), wie bei den Speichern, ist auch eine eindeutige Adreßzuordnung möglich. Außerdem erhöht sich dann die Anzahl der anschließbaren Port-Bausteine auf 64 ($=2^6$).

Ein-/Ausgabe mit Optokopplern

Über Ein-/Ausgabe-Schnittstellen sollen periphere Geräte und Einheiten an einen Mikroprozessor angeschlossen werden. Hierbei kann es zu Störungen des Mikroprozessors durch die peripheren Geräte kommen. Insbesondere, wenn diese Geräte oder Maschinen Magnete oder Motore enthalten. Gerade induktive Lasten können hier, infolge der Selbstinduktion, hohe Störspannungen erzeugen.

Ein anderes Problem stellen die Störungen dar, welche über die Masseleitung des Netzes in den Mikrocomputer gelangen. Hier hat es sich bewährt, die OV-Leitung des Mikrocomputers nicht mit der Netzmasse zu verbinden. Diese Maßnahme wird jedoch gestört, wenn periphere Geräte mit eigener Stromversorgung an den Mikrocomputer angeschlossen werden. In diesem Fall erfolgt unter Umständen eine Kopplung zwischen der OV-Leitung des MC und der Netzmasse über das Netzteil des peripheren Gerätes.

Zur Beseitigung der geschilderten Schwierigkeiten ist es häufig erforderlich, eine galvanische Trennung zwischen den Stromkreisen des Mikrocomputers und der peripheren Geräte vorzunehmen. Eine solche galvanische Trennung kann durch die Übertragung der Signale - die mit der peripheren Einheit ausgetauscht werden - mittels Licht erfolgen. Hierzu gibt es heute fertige Bausteine die man Optokoppler nennt.

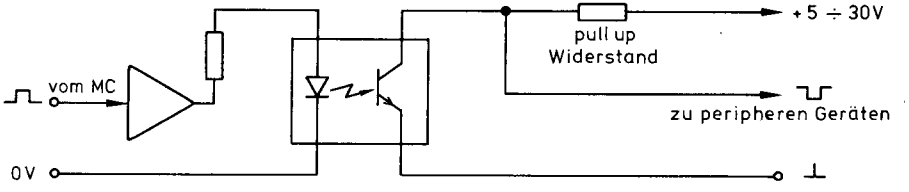


Abb.9.7: Galvanische Trennung durch Optokoppler

Die Leuchtdiode (LED) in dem Optokoppler (Abb.9.7) wird vom Mikrocomputer über eine Treiberstufe mit Strom versorgt. Wenn die LED leuchtet, trifft das Licht auf die Basis eines isoliert angeordneten Fototransistors und schaltet diesen durch. Dies erzeugt einen L-Pegel am Ausgang zum peripheren Gerät. Die Signale werden also invertiert. Durch den Ausgangstransistor mit offenem Kollektor ist darüber hinaus eine Anpassung an einen anderen logischen Pegel möglich. Die Isolationsspannung zwischen dem Ein- und Ausgang liegt typisch im kV-Bereich.

In der Abb.9.8 ist eine Ein-/Ausgabe-Schaltung - basierend auf dem Port-Baustein 8255 - mit vollständiger galvanischer Trennung aller Kanäle dargestellt. Zwei dieser Schaltungen finden gemeinsam auf einer Europakarte (160x100mm^2) Platz. Hierdurch erhält man insgesamt 24 Ein- und 24 Ausgabe-Kanäle.

Da diese Schaltung an einen mit 3MHz Takt arbeitenden MP 8085 angeschlossen werden sollte, kam der Baustein 8255A-5 zum Einsatz. Der Buchstabe A kennzeichnet eine weiterentwickelte Form des 8255, bei der jedoch die beschriebenen Eigenschaften unverändert erhalten geblieben sind. Die angehängte 5 weist diesen Baustein als besonders 'schnell' aus, um dem Timing des MP 8085 gerecht zu werden.

Als Treiber für die LEDs der Optokoppler enthält die Schaltung die 4 TTL-Bausteine 3 bis 6, mit jeweils 6 Treibern. In jedem der 6 Bausteine 7 bis 12 sind 4 einzelne Optokoppler vorhanden. Die 150 Ohm-Widerstände dienen zur Strombegrenzung der Leuchtdioden in den Optokopplern und die 4,7 k-Widerstände stellen pull-up-Widerstände dar.

Wenn anstelle der TTL-Treiber-Bausteine 5 und 6 CMOS-Bausteine (z.B. 4010) eingesetzt werden, ist die externe + 5V-Versorgungsspannung bis auf + 15V steigerbar. Hierdurch ist dann eine Anpassung an andere Logig-Pegel möglich.

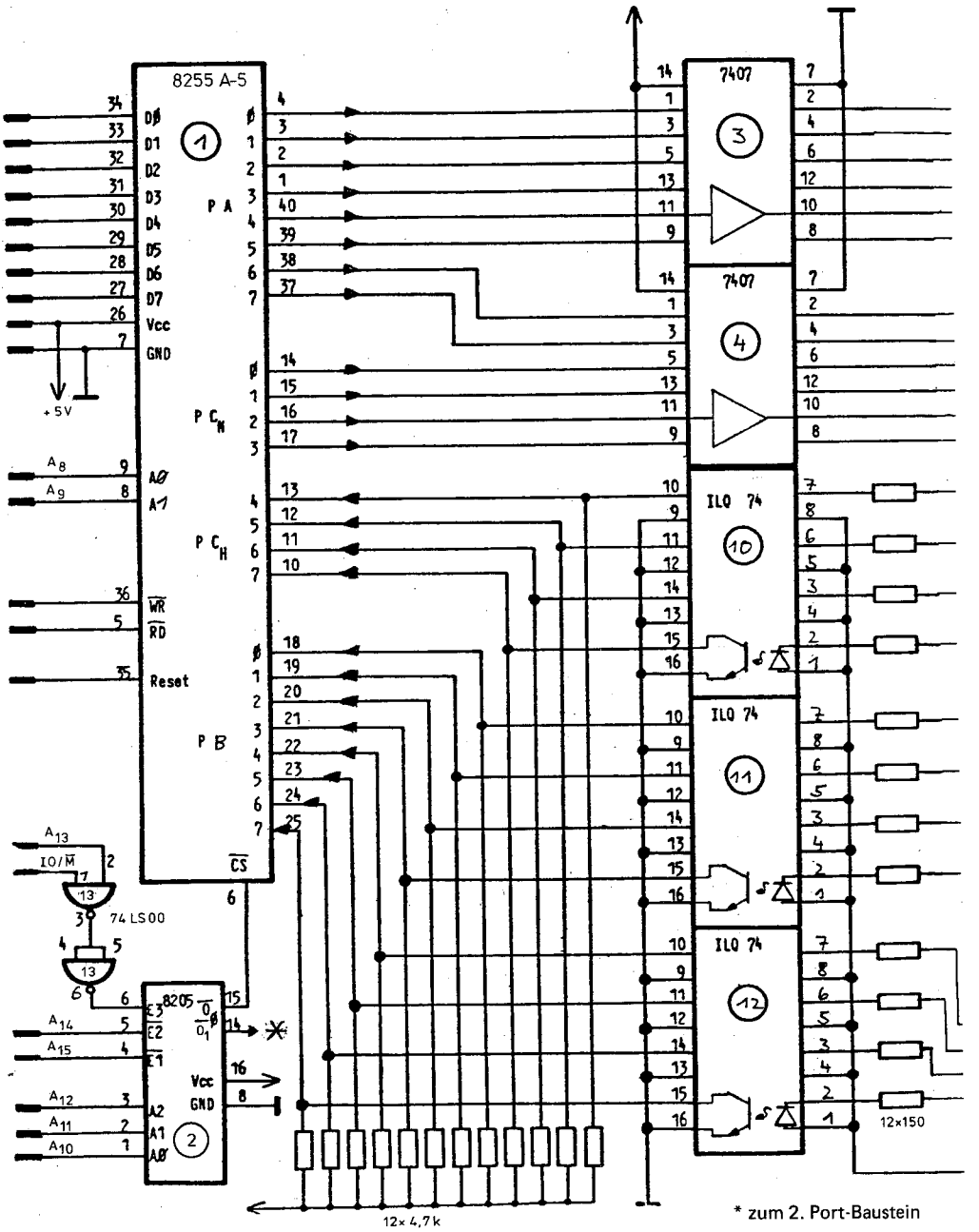
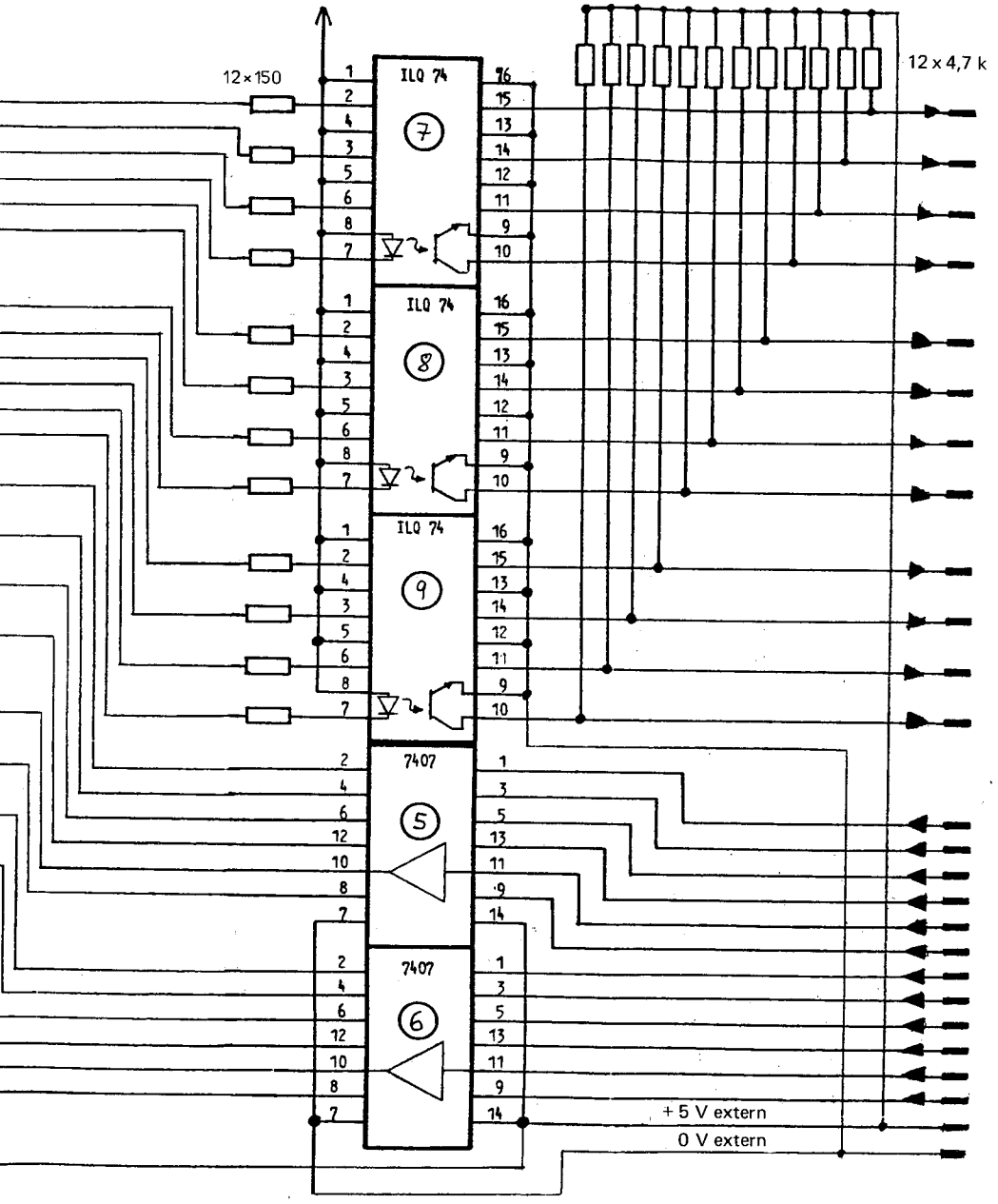


Abb.9.8: Ein-/Ausgabeschaltung zum Anschluß an den MP8085 mit 12 galvanisch getrennten Eingängen und 12 galvanisch getrennten Ausgängen. (Siehe auch Übungsaufgabe 9.3)



Je nach der gewählten Spannung müssen allerdings eventuell die 150 Ohm-Widerstände vor den Optokopplern 10 bis 12 durch höhere Widerstandswerte ersetzt werden.

9.2 Serielle Ein-/Ausgabe

Eine bitserielle Ein-/Ausgabe kann auf verschiedene Weise vorgenommen werden:

- Benutzung spezieller Kanäle des Mikroprozessors (z.B. SID und SOD des MP 8085),
- Verwendung einzelner Kanäle eines Parallel-Port-Bausteines (z.B. 8255, Kap.9.1),
- Einsatz spezieller Bausteine die eine parallel/seriell-Wandlung durchführen (Schieberegister, USART, etc.)

SID und SOD des MP 8085

Mit den Anschlüssen SID (serial input data) und SOD (serial output data) des MP 8085 (Abb.7.2) steht jeweils ein Eingabe- und ein Ausgabe-Kanal zur Verfügung. Insbesondere in kleinen Systemen ist hierdurch eine Beschränkung auf wenige Bausteine möglich.

Der Eingang SID wird durch den Befehl RIM gelesen und in Bit 7 des Akkumulators geschrieben. Darüber hinaus wird durch den RIM-Befehl die Interrupt-Maske gelesen (Abb.7.22).

```
Beispiel:     ; Einlesen einer 8bit-Zahl über SID
                MVI  B,8   ; Schleifenzähler
                MVI  C,0   ; Register zur Aufnahme der 8bit-Zahl
           BIT: RIM        ; Ein Bit einlesen (in Akkubit 7)
                ANI  80H   ; Akkubits 0 bis 6 auf L setzen
                MOV  D,A   ; Akku-Inhalt sichern
                MOV  A,C   ; Bisher eingelesene Bits in den Akku bringen
                RRC        ; Um ein Bit nach rechts verschieben
                ADD  D     ; Neu eingelesenes Bit hinzufügen
                MOV  C,A   ; Die Zahl in C sichern
                CALL WART  ; Zeitschleife zur Anpassung an die
                           ; Übertragungsrate
                DCR  B     ; Schleifenzähler um 1 erniedrigen
                JNZ  BIT   ; Sprung, wenn noch nicht alle Bits gelesen
              ; Die einzulesende Zahl steht im Register C
```

Der Ausgang SOD läßt sich durch den Befehl SIM setzen. Hierzu wird in Bit 7 des Akkumulators die auszugebende Information geschrieben, Bit 6 auf H gesetzt und dann der SIM-Befehl gegeben. Zu beachten ist jedoch, daß mit diesem Befehl gleichzeitig die Interrupt-Maske gesetzt wird (Abb.7.21).

Sollen an der Interrupt-Maske keine Veränderungen vorgenommen werden, so müssen die Bits 3 und 4 im Akkumulator vorher auf L gesetzt werden.

```
Beispiel:   MVI  A,11000000B ; Ausgabe-Bit in den Akku
            SIM              ; Am Ausgang SOD erscheint ein H und die
                             ; Interrupt-Maske wird nicht verändert
            MVI  A,40H
            SIM              ; Der Ausgang SOD wird auf L gesetzt
```

Ein weiteres Beispiel enthält Kap.10.4, dort wird über den Ausgang SOD ein Fernschreiber an den Mikroprozessor angeschlossen.

Parallel-Port 8255

Eine serielle Datenübertragung ist auch mit Hilfe eines Parallel-Ports möglich. Hierbei wird einer der Kanäle zur seriellen Ein- oder Ausgabe benutzt, während mit den anderen Kanälen des Ports beispielsweise Steuerungsaufgaben gelöst werden.

```
;Das auszugebende Byte steht im Akku
        PUSH    B
        PUSH    D
        MVI     C,8         ;Schleifenzaehler
        IN      4           ;Lesen des Zustandes von Port A
        ANI     7FH         ;UND-Verknüpfung: PA7=Akkubit7=L
        MOV     D,A         ;Zustandsinformation des Ports A sichern
EINBIT: MOV     A,B         ;Ausgabebyte in den Akku
        ANI     80H         ;Ausgabebit separieren,Bit 0-6 = L
        ADD     D           ;Zustand des Ports A zum Akku addieren
        OUT     4           ;Neues Bit und alten Zustand ausgeben
        CALL    WARTE       ;Festlegung der Übertragungsrate
        MOV     A,B         ;Ausgabebyte in den Akku
        RLC                 ;naechstes Ausgabebit auf höchste Stelle
        MOV     B,A         ;den Rest des Ausgabebytes sichern
        DCR     C           ;Schleifenzaehler - 1
        JNZ     EINBIT
        POP     D
        POP     B
        RET
WARTE   EQU     12A0H
```

Abb. 9.9: Programm zur seriellen Ausgabe eines Bytes auf einem Kanal eines 8255

Ein Beispiel hierfür enthält Abb.9.9. Es wird vorausgesetzt, daß der benutzte Mikrocomputer über einen Port-Baustein 8255 verfügt, daß Port A auf Ausgabe programmiert worden ist und die Adresse 04 hat (Abb.9.4). Die Ausgabe soll auf PA_7, d.h. auf dem 8. Kanal (mit der Bit-Wertigkeit 7) des Ports A erfolgen. Das Programm in Abb.9.9 zeigt die serielle Ausgabe eines Bytes, welches im Akkumulator an das Unterprogramm übergeben wird. Wenn die restlichen Kanäle des Ports A anderweitig benutzt werden, ist es wichtig, den Zustand des Ausgabe-Registers von Port A vorher zu lesen und dann unverändert bei der Ausgabe des seriellen Bits wieder zu übergeben.

Ein weiterer wichtiger Gesichtspunkt, bei einer seriellen Datenübertragung, stellt die Übertragungsgeschwindigkeit dar. Diese wird meistens als Baud-Rate bezeichnet, worunter man die Anzahl der übertragenen Bits pro Sekunde versteht.

Zur Einstellung der richtigen Baud-Rate wird hier eine Warte-Schleife benutzt.

Spezialbausteine

Eine Erhöhung der Datenübertragungsrate und eine Verringerung des Programmieraufwandes ist erreichbar, wenn die parallel/seriell Wandlung außerhalb des Mikroprozessors, durch einen speziellen Baustein, erfolgt.

Im einfachsten Fall kann hierzu ein Schieberegister benutzt werden.

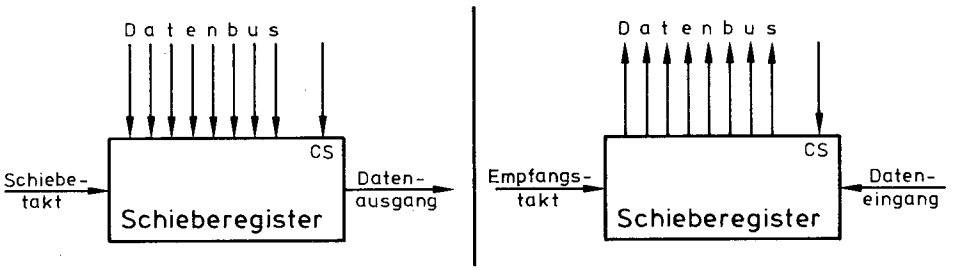


Abb. 9.10: Parallel/seriell und seriell/parallel Wandlung eines Datenstromes durch Schieberegister

Das parallel eingeschriebene Byte wird dann seriell und autonom weiter gegeben, oder der seriell ankommende Datenstrom wird in Parallelworte umgewandelt.

USART:

Zur Lösung dieser Aufgaben gibt es jedoch auch verschiedene programmierbare Bausteine. Einer dieser Bausteine ist der zur MP 8080-Familie gehörende 8251. Er wird häufig abkürzend USART genannt (universal synchronous/asynchronous receiver/transmitter). Dieser USART kann zur synchronen, wie auch zur asynchronen, seriellen Datenübertragung in beiden Richtungen benutzt werden.

Wie Abb.9.11 zeigt, enthält der USART einen seriellen Sender und Empfänger. Beide können gleichzeitig betrieben werden und somit beispielsweise den Zweirichtungs-Datenaustausch mit einer Datenstation (Terminal) realisieren.

Der Mikroprozessor übergibt dem USART Byte für Byte die auszusendende Information und dieser sendet sie bitseriell, gesteuert durch einen externen Takt, aus. Gleichzeitig kann der Baustein eine seriell einlaufende Information zu Bytes zusammenstellen.

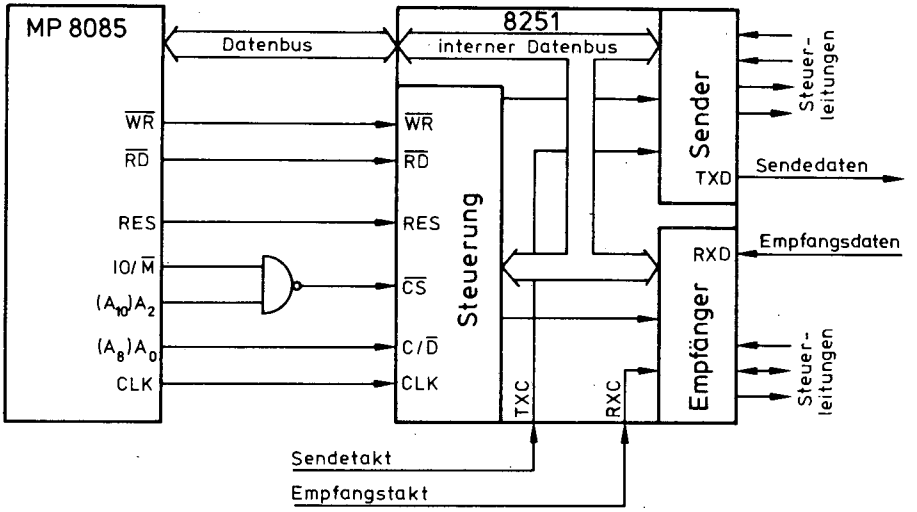


Abb. 9.11: Aufbau des USART8251 in vereinfachter Darstellung mit Anschluß an den MP8085.
C/D= Kennzeichnung der Parallel-Daten als Steuer- bzw. Status-Wort oder Zeichen (control/data). CLK= Systemtakt (clock). TxD= transmit data. TxX= transmitter clock. RxC = receiver clock.

Wenn das Senderegister leer, oder das Empfangsregister voll ist, wird an den Mikroprozessor ein Signal abgegeben, damit ein Datenaustausch erfolgen kann. Außerdem kann der Mikroprozessor jederzeit den Zustand des USART abfragen und erhält dann zugleich Aussagen über evtl. Datenübertragungs-Fehler und den Zustand von Steueranschlüssen geliefert.

Nach einem RESET-Impuls muß der 8251 zunächst programmiert werden. Je nach Betriebsart sind hierzu 2 bis 4 Bytes - unter Einhaltung einer genau definierten Reihenfolge - vom Mikroprozessor an den USART zu senden.

Das erste Steuerwort ist das Mode-Wort. Hiermit wird die Betriebsart (Synchron/Asynchron), die Länge eines Wortes (5 bis 8bit), ob und welche Parität (Datensicherung), sowie Angaben zur Art der Synchronisation oder zur Länge der Sperrschritte festgelegt. Danach können ein bis zwei spezielle SYNC-Worte folgen. Diese werden dann beim Senden automatisch in den Datenstrom eingefügt, oder beim Empfang aus ihm eliminiert.

Als letztes Steuerwort wird das Kommando-Wort übertragen. Hierdurch können der Sender und Empfänger freigegeben und gesperrt und verschiedene Hilfsfunktionen gesteuert werden (siehe Datenbücher).

Mit einem solchen Baustein läßt sich fast jedes heute gebräuchliche serielle Datenübertragungsverfahren realisieren.

9.3 Übungsaufgaben

9.1 Über eine Schnittstelle gemäß Abb.9.2 soll eine byteserielle Datenübertragung realisiert werden. Zur Bausteinauswahl wird A_5 benutzt. Es ist eine Übertragung von 100 Worten (Bytes) vorgesehen. Dieser Informationsblock soll in den Speicherbereich ab 4000H abgelegt werden. Wie müßte ein entsprechnedes Programm aussehen und wie groß ist dann die maximale Datenübertragungsgeschwindigkeit (Byte/s) bei einem MP 8085 mit 6,144MHz Quarz?

9.2 Ändern Sie das Programm aus Aufgabe 9.1 so ab, daß eine Übertragungsrate von 48000 Byte/s eingehalten wird.

9.3 Schreiben Sie ein Programm zur Programmierung des 8255 in Abb.9.8; geben Sie auf Port A die Zahl 55H heraus und lesen Sie Port B ein.

10. Peripherie-Geräte

Zur Ausführung sinnvoller Tätigkeiten benötigt ein Computer periphere Einheiten, wie z.B. eine Tastatur, (keyboard) und eine Anzeigeeinheit (display).

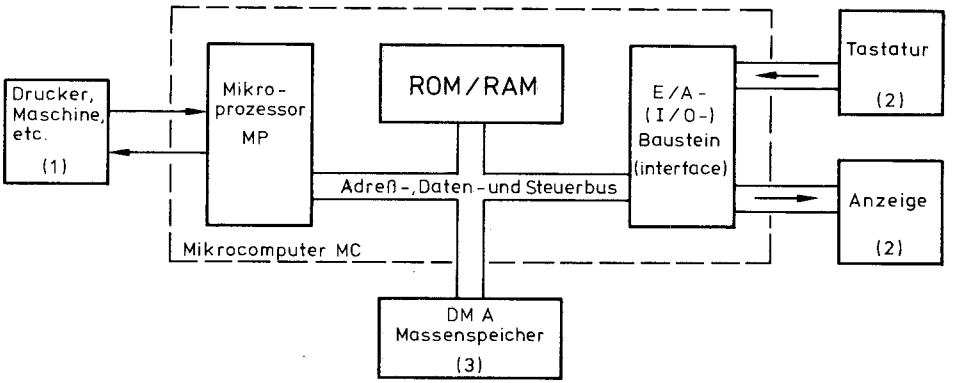


Abb. 10.1: Anschluß peripherer Einheiten an einen Mikrocomputer

Alle außerhalb des Mikrocomputers angeordneten Einheiten und Baugruppen sollen hier unter dem Begriff 'Peripherie-Geräte' zusammengefaßt werden. Wie Abb.10.1 zeigt, kann der Anschluß dieser Geräte an den Mikrocomputer auf verschiedene Weise erfolgen. Im einfachsten Fall (1) ist eine Kopplung über spezielle Kanäle des Mikroprozessors möglich (Kap.9.2).

Werden viele Kanäle gebraucht (2), so erfordert dies die Zwischenschaltung eines Ein-/Ausgabe-Bausteines (interface, Kap.9.1).

Zur Realisierung eines besonders schnellen Datenaustausches mit dem Mikrocomputer (3) wird die Technik des direkten Speicherzugriffes (DMA) angewandt. Hierbei wird der Mikroprozessor kurzzeitig angehal ten, so daß die periphere Einheit, unter Benutzung der Busse des Mikrocomputers, einen direkten, schnellen Datenaustausch mit einem internen Speicherbereich durchführen kann.

In den folgenden Kapiteln werden einige wichtige Geräte oder Baugruppen beschrieben, die insbesondere in der Mikrocomputertechnik Anwendung finden.

Darüberhinaus sollen, anhand dieser Beispiele, verschiedene Interface-Techniken und die Wirkungsweise der Geräte kurz beschrieben werden.

10.1 Eingabetastatur

Der Anschluß von Tasten oder Schaltern an einen Mikrocomputer, befähigt den Menschen einen direkten Einfluß auf die internen Vorgänge des Mikrocomputers auszuüben. Diese Wirkung ist jedoch nur zu erreichen, wenn die angeschlossenen Tasten oder Schalter im Programmablauf ständig abgefragt (polling) werden, oder auf Interrupt-Eingänge wirken (Kap.7.4).

Beispiele: Eingaben an den Mikrocomputer
- Signal für Programmverzweigungen
- Anforderung spezieller Aktivitäten
- Eingabe von Kommandos und Befehlen
- Speicherinhalte verändern

Für eine möglichst universelle Kommunikation mit dem Mikrocomputer, sind im einfachsten Fall 16 Tasten erforderlich. Hierbei ist jeder der 16 Tasten eine andere Hexadezimalzahl zugeordnet. Der Druck einer Taste bedeutet somit die Aussendung eines 4bit-Wortes (Tetrade) an den Mikrocomputer. Damit der Mikrocomputer dieses Wort empfängt und richtig 'versteht', muß er durch ein entsprechendes Programm dazu veranlaßt werden alle 16 Tasten abzufragen und, wenn eine Taste gedrückt wurde, ihren Wert (0...F bzw. LLLL...HHHH) in ein Register zu schreiben. Was dann mit diesem Wort, oder dieser Zahl weiter geschehen soll, oder welche Bedeutung sie hat, wird ebenfalls durch eine vorherige Programmierung festgelegt.

Eine Minimaltastatur, bestehend aus 16 Tasten, wird häufig als quadratische Matrix gemäß Abb.10.2 aufgebaut. Aufgrund dieser speziellen Anordnung sind nur 8 E/A-Kanäle zur Abfrage von 16 Tasten erforderlich.

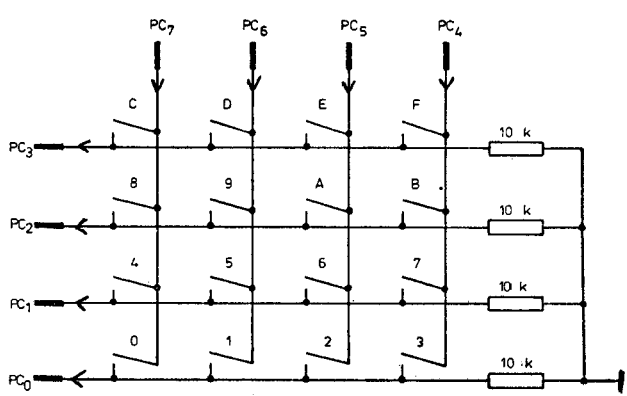


Abb. 10.2: Hexadezimal-Tastatur

Der hexadezimale Wert, welcher einer Taste zugeordnet wird, ist durch das Programm bestimmt.

In Abb.10.2 wurden die Zeilenleitungen an die niederwertige Tetrade des Ports C eines 8255 (Kap.9.1) und die Spaltenleitungen an die höherwertigen 4 Kanäle angeschlossen. Bei geöffneten Tasten nehmen die Eingänge $PC_0 \ldots PC_3$, infolge der pull-down-Widerstände, den L-Zustand an. Wenn nun auf einen der Ausgänge $PC_4 \ldots PC_7$ ein H gegeben wird und darüber hinaus in der selben Spalte eine Taste gedrückt ist, erscheint auf der entsprechenden Zeile ein H.

Beispiel:
```
        ; Die Taste A sei geschlossen (Abb.10.2).
        MVI  A,20H  ; LLHLLLLL
        OUT  PC     ; PC5=H
        IN   PC     ; ⟨A⟩=LLHLLHLL
        ; Auf dem Eingang PC2 (Abb.10.2) erscheint ein H.
```

Durch die Ausgabe eines H auf allen 4 Spaltenleitungen nacheinander und das jeweilige Einlesen der 4 Zeilenleitungen, ist eindeutig feststellbar, welche Taste gedrückt wurde. Der Aufbau eines entsprechenden Tastenabfrageprogrammes ist in Abb.10.3 dargestellt.

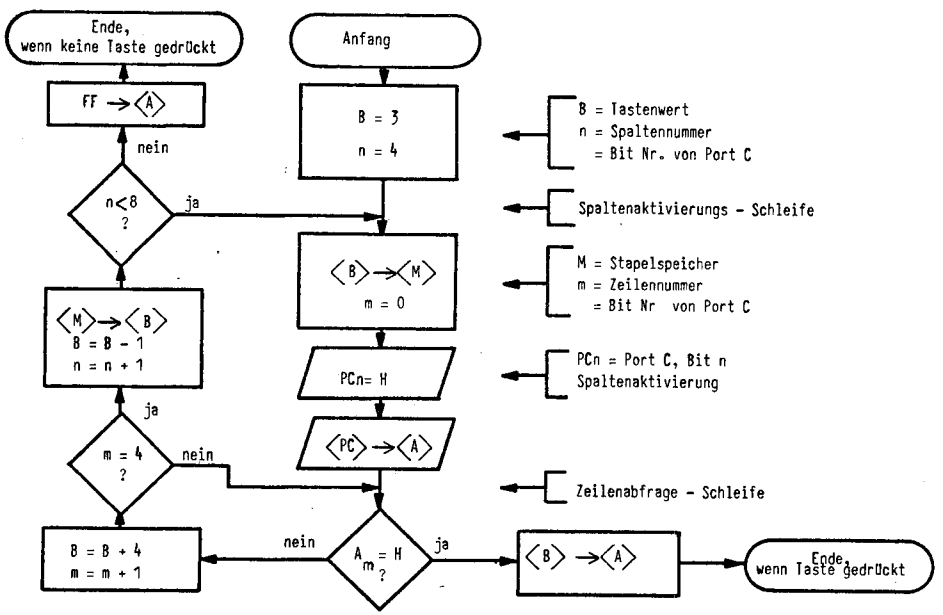


Abb. 10.3: Tastenabfrageprogramm für 4x4 Tasten gemäß Abb. 10.2

Die Tasten erhalten hierbei die in Abb.10.2 angeschriebenen Werte zugewiesen (siehe auch Übungsaufgabe 10.1)

Alphanumerische Tastatur

Wenn beliebige Texte in den Mikrocomputer eingegeben werden sollen, bedient man sich einer Tastatur wie sie von den Schreibmaschinen her bekannt ist.

Der am häufigsten benutzte Code zur Darstellung alphanumerischer Zeichen ist der ASCII-Code (Abb.1.8). Weiterhin enthält dieser Code noch eine Reihe nicht darstellbarer Steuerzeichen. Insgesamt sind durch 7bit 128 Zeichen codierbar. Die Anordnung von 128 Tasten, in einer Matrix gemäß Abb.10.2, würde zur Decodierung ca. 23 E/A-Kanäle erforderlich machen.

Im allgemeinen wird so nicht verfahren, sondern eine Mehrfachbelegung von Tasten vorgenommen. Hierdurch reduziert sich die Anzahl der Zeichentasten auf ca. 50. Hinzu kommen dann jedoch noch evtl. Sondertasten und vorallem die Tasten, welche zur Umschaltung der Tastenbelegung erforderlich sind.

Eine solche Umschaltung gibt es beispielsweise auch bei jeder Schreibmaschine zur Darstellung der Großbuchstaben. Bei der in Abb.10.4 dargestellten Tastatur wird diese Funktion durch die Taste 'SHIFT' ausgelöst.

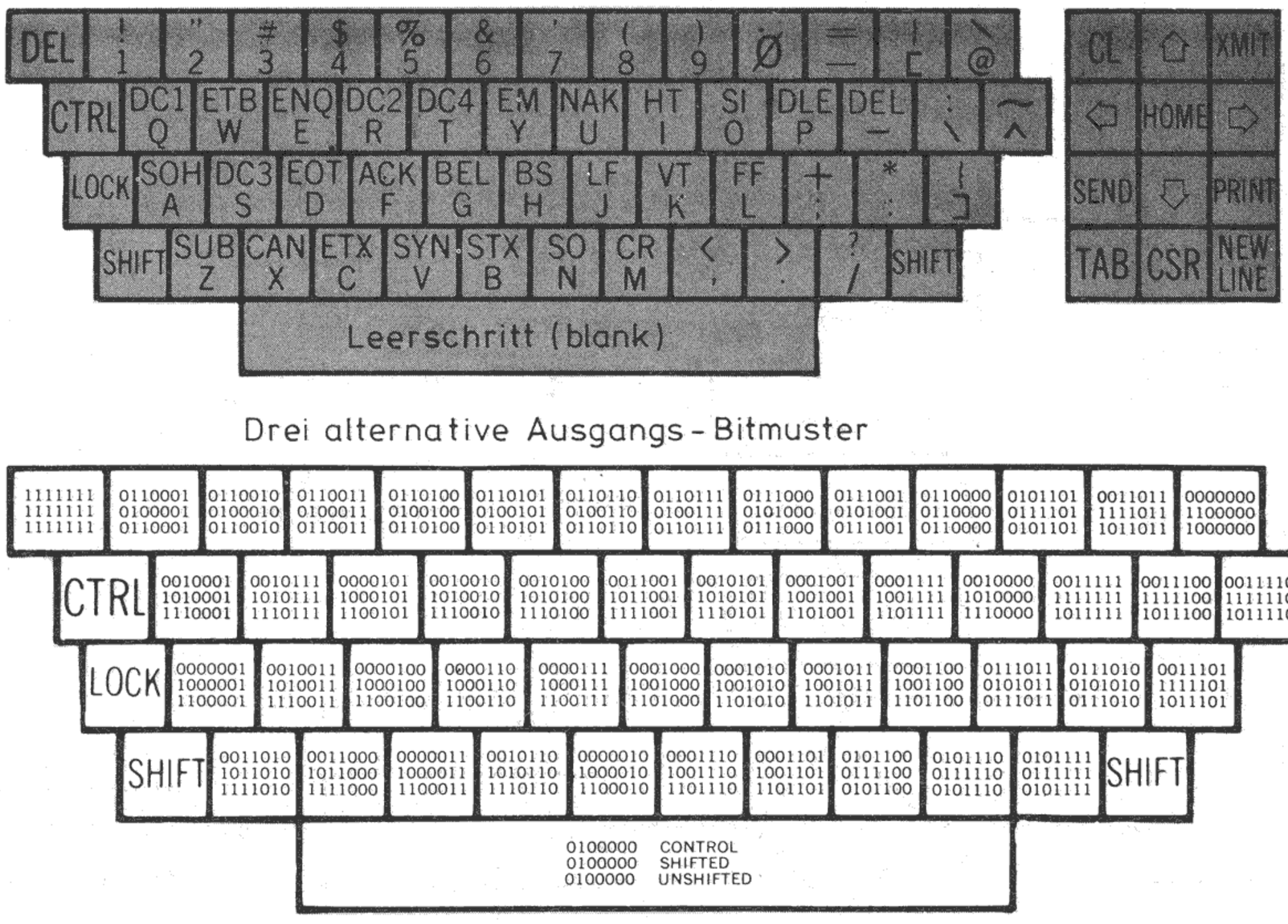


Abb. 10.4: 3-modale ASCII-Tastatur mit Cursor-Steuerung und Zusatztasten. 0/1 entspricht L/H.

Die Taste 'LOCK' dient zur dauerhaften Umschaltung auf Großbuchstaben und zur Aussendung des Codes, der auf den Tasten oben dargestellten Zeichen. Wenn eine Zeichentaste in Verbindung mit der zweiten Umschalttaste (CTRL), der sog. Control-Taste, gedrückt wird, kommt es zur Aussendung des entsprechenden Control- oder Steuerzeichencodes (Abb.1.8).
Damit eine solche Tastatur einfacher, und über weniger Leitungen, an einen Computer anschließbar ist, wird die Decodierung der Tasten meistens durch eine spezielle Schaltung vorgenommen, die unterhalb der Tasten auf der selben Leiterplatte angeordnet ist. Über sieben Anschlußstifte sendet dann die Tastaturbaugruppe den ASCII-Code der jeweils gedrückten Taste - einschließlich der Umschaltmöglichkeiten - an den Computer.

Im unteren Teil der Abb.10.4 sind die drei alternativen Bitmuster, die beim Druck einer Taste ausgesandt werden können, dargestellt.

Anders als bei einer Schreibmaschine trifft man bei einer Computer-Tastatur häufig noch eine mehr oder weniger große Anzahl von Zusatztasten an. Diese Zusatz- oder Sondertasten sind im allgemeinen nicht codiert sondern werden direkt auf Anschlußstifte geführt und können dann - z.B. über Portkanäle - per Programm eingelesen und beliebig interpretiert werden. Insbesondere die Tasten mit den 4 Pfeilen und die HOME-Taste (Abb.10.4) sind von großer Wichtigkeit, denn sie dienen der Cursor-Steuerung (Cursor = frei über einen Bildschirm verschiebbares Markierungszeichen).
Abschließend sei noch darauf hin gewiesen, daß bei einer voll decodierten Tastatur, wie in Abb.10.4, jedes Mal wenn eine Taste gedrückt wird, auf einer besonderen Leitung ein sog. Strobe-Signal (STB) abgegeben wird. Verbindet man den STB-Ausgang mit dem Interrupt-Eingang des Mikroprozessors, so 'merkt' der Mikrocomputer, wenn eine Taste gedrückt wurde und braucht deshalb die Tastatur nicht regelmäßig abzufragen.

Kontaktprellen

Wenn ein Schalter oder Taster betätigt wird, tritt im allgemeinen ein sog. Kontaktprellen auf. Hierunter versteht man einen alternierenden Zustand, bei dem es infolge mechanischer Schwingungen zu schnellen Ein- und Ausschaltvorgängen kommt. Wird ein solcher Kontakt sehr schnell hintereinander abgefragt, dann liest der Mikrocomputer mehrere L/H-Wechsel ein, obwohl der Schalter oder Taster nur einmal betätigt wurde.
Das Prellen ist sehr stark von der Bauart des Schalters abhängig. Bei modernen kontaktlosen Schaltern tritt das lästige Prellen praktisch garnicht mehr auf.

Negative Auswirkungen des Prellens können durch die Nachschaltung aktiver Bausteine, oder durch Software-Maßnahmen verhindert werden. Wenn innerhalb eines Programmes ein Schalter oder Taster abgefragt wird, sollte bis zur erneuten Abfrage eine Wartezeit von ca. 20 Millisekunden, zur Überbrückung der Prellphase, eingehalten werden.

Da eine Taste sicherlich länger als 20 Millisekunden gedrückt wird, ist es zur Vermeidung von Mehrfacheinlesungen des selben Wertes erforderlich, in einer speziellen Programmschleife solange zu warten, bis die Taste wieder losgelassen wurde, bevor alle Tasten erneut abgefragt werden.

10.2 Ausgabe auf Anzeigeelemente

Zur Kommunikation mit dem Menschen benötigt ein Mikrocomputer Ausgabeeinheiten. Im einfachsten Fall reichen hierzu einzelne Lämpchen oder Leuchtdioden, um bestimmte Signale nach außen zu geben. Diese Form der Ausgabe wird beispielsweise bei einfachen Geräten benutzt, die durch einen Mikrocomputer gesteuert werden (z.B. Nähmaschine, Filmprojektor, Unterhaltungselektronik etc.)

Zur Ausgabe numerischer Ergebnisse reichen diese Anzeigeelemente jedoch im allgemeinen nicht aus. Hierzu sind dann spezielle Ziffernanzeigeelemente erforderlich.

Siebensegment-Anzeigen

Die einfachste Form der Ausgabe numerischer Daten geschieht mit Hilfe sog. Siebensegment-Anzeigen.

Jedes Zeichen wird durch maximal sieben Striche einer fest vorgegebenen Matrix realisiert. Die 'Striche' können entweder aus Licht aussendenden Elementen, wie z.B. Leuchtdioden (LED), oder auch aus veränderlichen Strukturen, wie z.B. bei den Flüssigkristallen, bestehen. Das am weitesten verbreitete Beispiel für diese Form der Anzeige stellen die Digitaluhren dar. Die Abb.10.5 zeigt ein Beispiel für den Anschluß zweier LED-Siebensegment-Anzeigen an einen Mikrocomputer.

Die Anoden aller Leuchtdioden in dem Anzeigebaustein sind untereinander verbunden und werden über 2 Anschlüsse (3 + 14) nach außen geführt. Ein Schalttransistor, angesteuert über einen Kanal des Ports B, gestattet es, den Anzeigebaustein wahlweise an die + 5V Versorgungsspannung anzuschließen oder nicht. Die Kathoden der einzelnen Leuchtdioden sind nach außen geführt und können, verstärkt durch einen Treiber, über den Port A des Mikrocomputers mit Strom versorgt werden.

Um auf dem Baustein 1 ein Zeichen zur Anzeige zu bringen, wird zunächst auf Port A eine 7bit-Zahl ausgegeben, welche genau die Leucht-

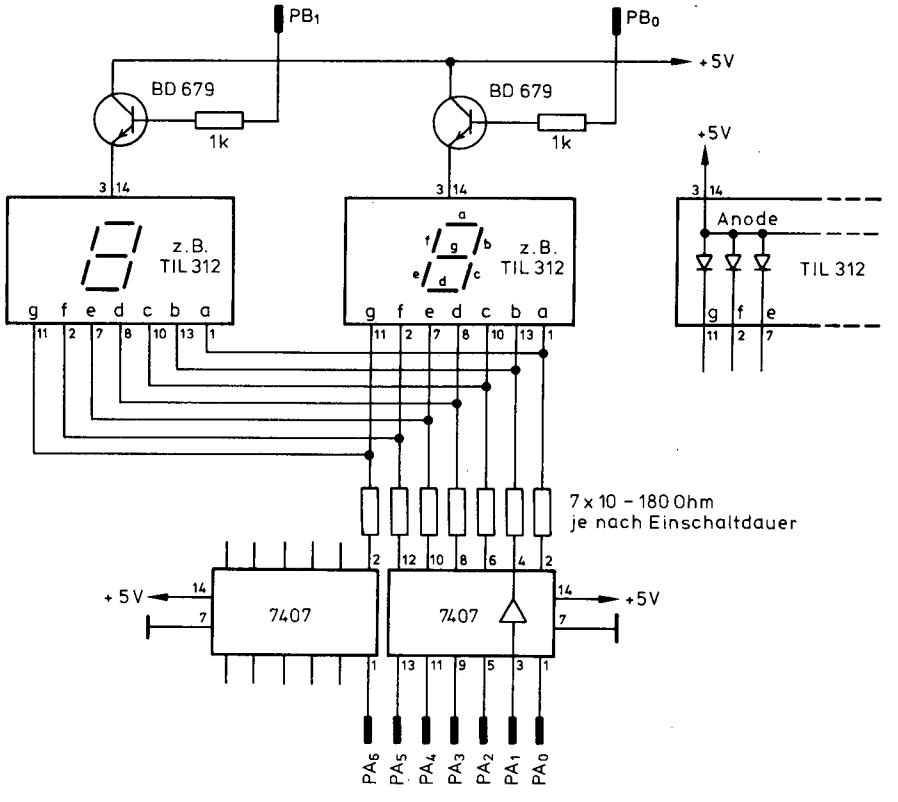


Abb. 10.5: Ansteuerung von 2 Siebensegment-Anzeigen

dioden gegen Masse schaltet, die zur Darstellung des gewünschten Zeichens erforderlich sind. Im nächsten Schritt wird dann auf dem Kanal PB_0 ein H ausgegeben, um die Anoden der Leuchtdioden in Anzeige 1 mit + 5V zu verbinden. Wird die Ausgabe auf Anzeige 2 gewünscht, so muß auf PB_1 ein H gegeben werden.

```
Beispiel:    MVI  A,LHHHHLLHB
             OUT  PA
             MVI  A,LLLLLLLHB
             OUT  PB
             ; Auf Anzeige 1 erscheint eine 1
```

Zur Darstellung einer zweistelligen Zahl wird das sog. Multiplex-Verfahren angewandt.
Aus Abb.10.6 ist ersichtlich, was hierunter zu verstehen ist. Während auf Port A die Bitkombination ansteht, die zur Darstellung der entsprechenden Ziffer auf Anzeige 1 erforderlich ist (Datum 1), führt der Kanal PB_0 ein H und PB_1 ein L.

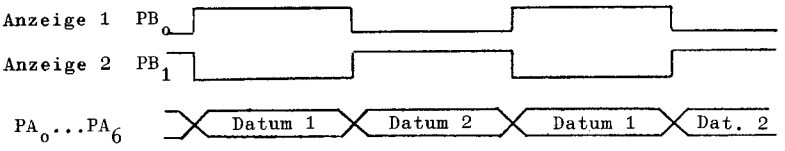


Abb. 10.6: Ansteuerung der Anzeigen aus Abb. 10.5 im Multiplex-Verfahren

Kurze Zeit später wird das Datum 2 auf Port A herausgegeben und die Anzeige 2 an die + 5V-Leitung angeschlossen. Es leuchtet also immer nur eine Anzeige. Wenn dieser Vorgang immer wieder in rascher Folge durchlaufen wird, erscheint für das Auge eine flackerfreie zweistellige Anzeige. Nach diesem Verfahren können auch vielstellige Anzeigen betrieben werden. Für jeden weiteren Anzeigebaustein ist lediglich ein zusätzlicher Auswahlkanal zum Anschluß an die Versorgungsspannung erforderlich.

Ein Nachteil dieses Verfahrens ist darin zu sehen, daß die Anzeige nur arbeitet, wenn sie vom Mikrocomputer permanent bedient wird. Aus diesem Grund gibt es heute spezielle Ansteuerbausteine, die das Multiplexen der Anzeige übernehmen.

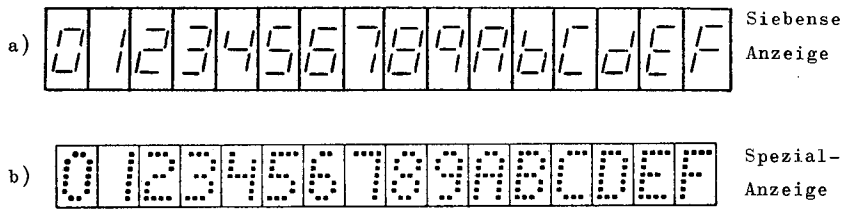


Abb. 10.7: Darstellung der Hexadezimalzeichen

Beim Einsatz dieser Anzeigen in frei programmierbaren Mikrocomputersystemen müssen neben den Dezimalziffern auch noch die zum hexadezimalen Alphabet gehörenden Buchstaben A bis F dargestellt werden. Hierfür wurden die Siebensegmentanzeigen ursprünglich nicht konzipiert. Wie Abb.10.7a zeigt, ist jedoch eine eindeutige Darstellung möglich.

Jedes Anzeigeelement kann dadurch zur Ausgabe einer 4bit-Zahl (Tetrade) herangezogen werden.

Hexadezimal-Anzeigen

Eine bessere Lesbarkeit der Hexadezimalzahlen wird durch die Verwendung spezieller Anzeigeelemente erreicht (Abb.10.7b).

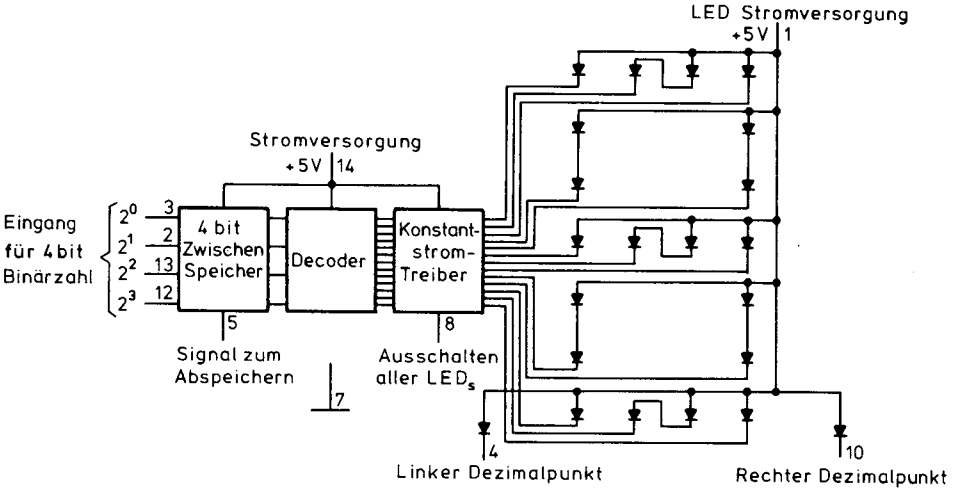


Abb. 10.8: Aufbau einer Hexadezimalanzeige mit integrierter Schaltung (TIL 311)

Der Anzeigebaustein TIL 311 - dargestellt in Abb.10.8 - enthält beispielsweise 20 punktförmige LEDs für den Aufbau der 16 Hex-Zeichen gemäß Abb.10.7b. Darüberhinaus enthält die Anzeige noch eine integrierte Schaltung zur Speicherung der als Hex-Zahl darzustellenden Tetrade, einen Decoder für die Umsetzung in ein entsprechendes Punktmuster und Konstantstromquellen zur Erreichung einer gleichen Helligkeit aller LEDs.

Alphanumerische Anzeigen

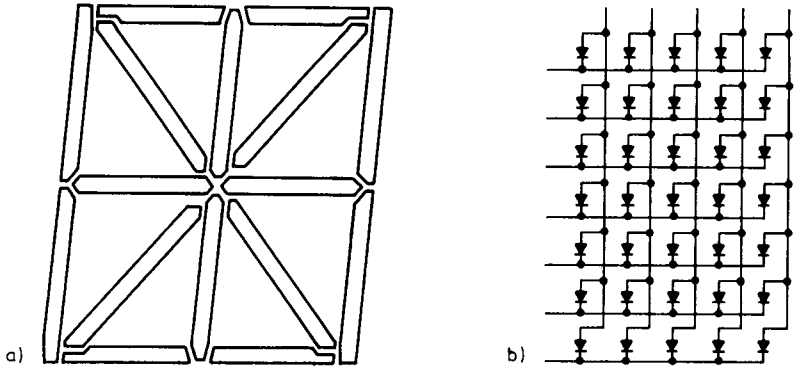


Abb. 10.9: Anordnungen zur Darstellung alphanumerischer Zeichen.
a) 16-Segment-Anzeige. b) 5x7 Dioden-Punktmatrix.

Zur Darstellung allgemeiner Daten - Buchstaben, Zahlen und Sonderzeichen - dienen sog. alphanumerische Anzeigen.

Mit der in Abb.10.9a dargestellten 16-Segment-Anzeige läßt sich gut ein Zeichenvorrat, bestehend aus den Dezimalzahlen, den großen Buchstaben und den Sonderzeichen, aufbauen.

Sollen beliebige Zeichen aufgebaut werden, bedient man sich meistens einer 5x7 Punktmatrix. In Abb.10.9b ist eine entsprechende Matrix, bestehend aus 35 Leuchtdioden dargestellt. Die Ansteuerung eines solchen Bausteines ist sehr aufwendig.

Inzwischen gibt es jedoch bereits mehrstellige Anzeigen, sowohl in der 16-Segment- als auch in der Punktmatrix-Form, mit integrierter Ansteuerungselektronik.

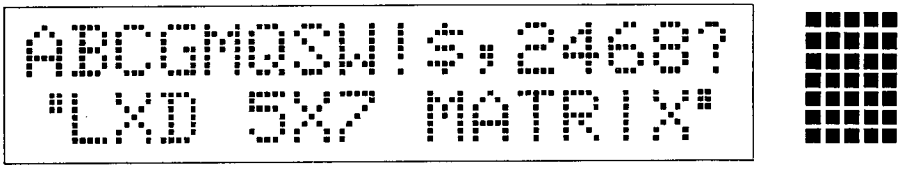


Abb. 10.10: Darstellung alphanumerischer Zeichen auf einer 2x16-stelligen Flüssigkristallanzeige unter Verwendung einer 5x7-Punktmatrix

Ein Beispiel hierfür zeigt die Abb.10.10

10.3 Ausgabe auf Bildschirmen

Das gebräuchlichste und vielseitigste Ausgabemedium stellt der Bildschirm dar. Als Bildschirm finden normale Fernsehbildröhren oder spezielle Weiterentwicklungen hiervon Anwendung. Im amerikanischen spricht man deshalb auch von Kathodenstrahlröhren (CRT = cathode ray tube).

Ein Bildschirm ist für die Ausgabe von Computerdaten besonders gut geeignet, da große Datenmengen gleichzeitig und übersichtlich darstellbar sind.

Zum Anschluß an einen Mikrocomputer eignen sich sowohl handelsübliche Fernsehgeräte als auch spezielle Geräte, welche häufig als Videomonitore bezeichnet werden. Die Übertragung der Bildinformation an ein Bildschirmgerät kann entweder direkt - auf der Basis von Videosignalen - erfolgen, oder aber bei Fernsehgeräten, unter Zwischenschaltung eines VHS- oder UHF-TV-Modulators, über den Antenneneingang.

Auf einem Bildschirm sind beliebige Zeichen und, bei entsprechender Auslegung, auch graphische und farbige Bilder darstellbar.

Fernsehtechnik:

Das Bild auf einem Kathodenstrahl-Bildschirm (CRT) wird durch einen, den Schirm zeilenförmig überstreichenden, Elektronenstrahl erzeugt, der einen Leuchtstoff anregt.

Um ein flackerfreies Bild zu bekommen, werden im ersten Durchlauf nur die ungeraden Zeilen (1. 3. 5.....) und im zweiten Durchlauf die dazwischenliegenden Zeilen (Nr. 2. 3. 4....) geschrieben. Dies geschieht mit einer Frequenz von 50 Hz, so daß insgesamt 25 Bilder/s erscheinen. Dieses sogenannte Zeilensprungverfahren wird jedoch beim Anschluß an einen Mikrocomputer nicht ausgenutzt, es wird vielmehr zweimal die gleiche Information auf den Bildschirm gegeben.

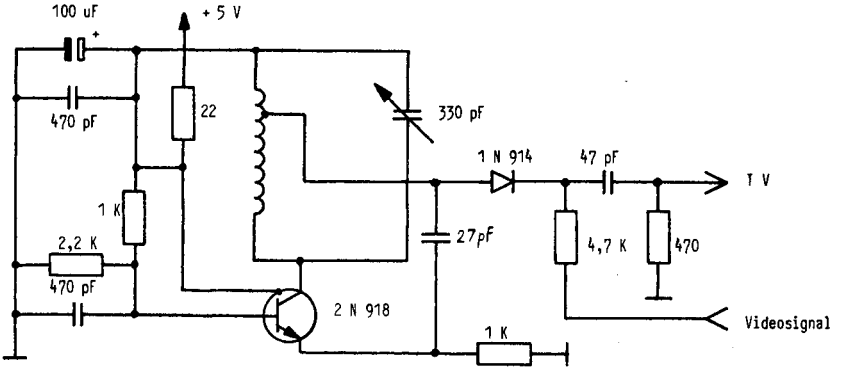


Abb. 10.11: Schaltung eines Videomodulators

Wenn zur Informationsübertragung an das Fernsehgerät der Antenneneingang benutzt wird, muß das Videosignal über einen Modulator (Abb.10.11) für einen Fernsehkanal aufbereitet werden.

Der Vorteil dieses Verfahrens besteht darin, daß der Mikrocomputer an ein beliebiges Fernsehgerät angeschlossen werden kann. Nachteilig ist die zur Verfügung stehende Bandbreite, welche häufig, insbesondere bei Billiggeräten, entweder zu einer schlechten Bildqualität führt, oder nur die Darstellung einer geringeren Zeichenanzahl zuläßt. Wenn immer das selbe Fernsehgerät eingesetzt wird, ist es günstiger den Video-Eingang zu benutzen. Falls keiner vorhanden ist, so ist er i.a. sehr leicht nachrüstbar.

Zeichendarstellung:

Die Zeichen auf einem Bildschirm setzen sich aus einzelnen Punkten zusammen, ähnlich wie bei einer Anzeige gemäß Abb.10.10. Auch hier kann eine 5x7 oder 7x9 Matrix benutzt werden, je nachdem, wie komfortabel die Darstellung sein soll.

Als nächstes ist zu entscheiden, wieviel Zeichen in welcher Aufteilung auf den Bildschirm gebracht werden sollen. Wobei hier eine obere Grenze durch die Lesbarkeit und die zur Übertragung erforderliche Bandbreite gegeben ist.

Bei der Benutzung billiger Fernsehgeräte und deren Antenneneingang, sollten nicht mehr als 16 Zeilen mit 32 Zeichen/Zeile dargestellt werden. Wenn ein transportables Fernsehgerät der mittleren Preisklasse und dessen Videoeingang benutzt wird, sind 24 Zeilen und 64 bis 80 Zeichen/Zeile noch gut lesbar.

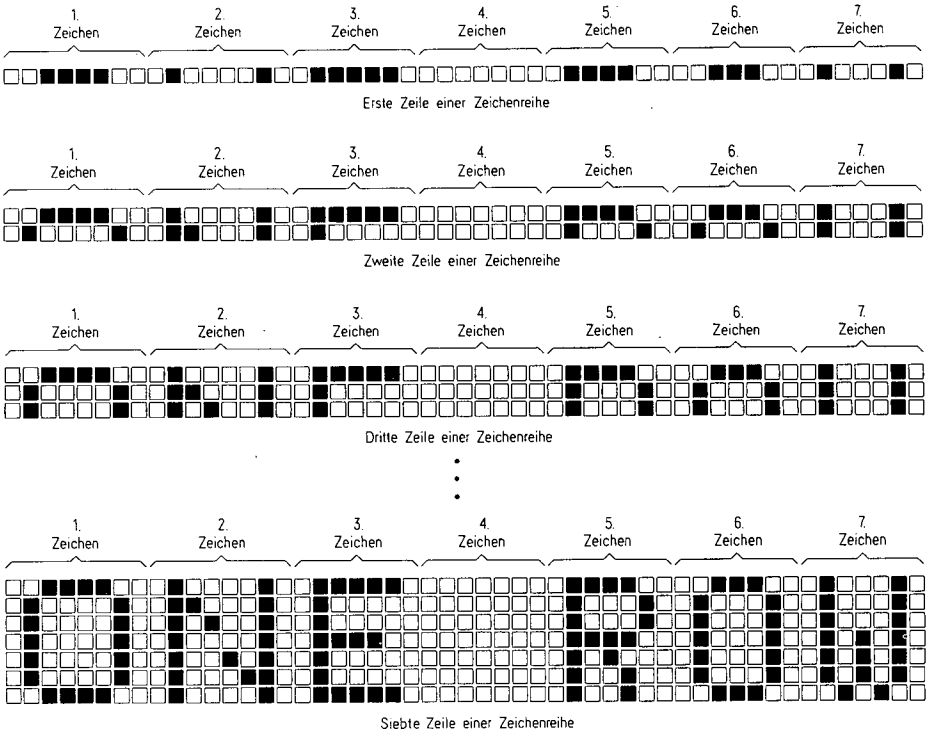


Abb. 10.12: Entstehungsstufen einer Zeichenzeile

Eine Zeichenreihe (Abb.10.12) entsteht dadurch, daß z.B. 7 Zeilen untereinander geschrieben werden und immer an den richtigen Stellen ein leuchtender Punkt erzeugt wird.
Da ein Fernsehbild aus 625 Zeilen besteht, jedoch nur ein Halbbild ausgenutzt wird, stehen 312 Zeilen für die Zeichendarstellung zur Verfügung. Bei einer 5x7 Zeichenmatrix und zwei Leerzeilen nach jeder Zeichenreihe, lassen sich also maximal 35 Zeichenreihen darstellen.

Wenn eine Zeile in genausoviel Bildpunkte zerlegt wird wie es Zeilen gibt (625), so lassen sich bei einer 5x7 Zeichenmatrix mit 2 Leerspalten (Abb.10.12) maximal 89 Zeichen/Reihe abbilden.

Zeichenerzeugung:

Zur Zeichenerzeugung auf dem Bildschirm ist eine genaue zeitliche Steuerung der Momente erforderlich, in denen ein Leuchtpunkt auf dem Schirm erscheinen soll. Hierzu gibt es heute hochintegrierte Bausteine (CRTC = cathode ray tube controller), die in Verbindung mit einigen anderen Bausteinen, diese Aufgabe übernehmen.

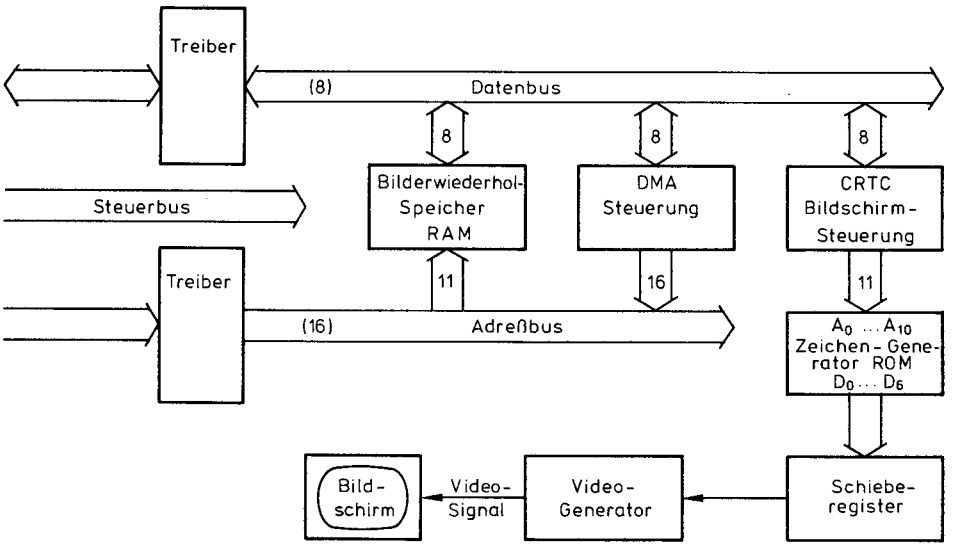


Abb. 10.13: Aufbau einer Bildspeicher- und Videosteuereinheit (Videoram)

Der in Abb.10.13 dargestellte Aufbau einer kompletten Steuer- und Speichereinheit, eignet sich besonders gut für die Anwendung in Mikrocomputersystemen, denn die gesamte Einheit wird einfach wie ein RAM an den Mikroprozessor angeschlossen.
Über den Datenbus wird dem Bildschirm-Steuerbaustein (CRTC) mitgeteilt, welche Zeichen in der zu schreibenden Reihe auf dem Bildschirm erscheinen sollen. Wenn z.B. die beiden Worte ONE ROW (Abb.10.12) zu schreiben sind, sendet der CRTC an das Zeichengenerator-ROM eine Adresse, so daß der Speicherplatzinhalt LLHHHHLL an das Schieberegister weitergeleitet wird. Diese 8 Bits werden dann hintereinander als Videosignal zur Hell/Dunkel-Tastung der ersten Zeile des ersten Zeichens, dargestellt in Abb.10.12, herangezogen. Sofort anschließend wird durch eine andere Adresse die erste Zeile des zweiten Zeichens geschrieben.
In dem Zeichengenerator-ROM stehen auf aufeinanderfolgenden Speicherplätzen die Punktmuster aller darstellbaren Zeichen.

Zeichengenerator ROM															
Adr.	Inhalt								Adr.	Inhalt							
410H	L	L	L	L	L	L	L		420H	L	L	L	L	L	L	L	
411	L	L	L	H	L	L	L		421	L	H	H	H	H	L	L	
412	L	L	H	L	H	L	L		422	L	L	H	L	L	H	L	
413	L	H	L	L	L	H	L		423	L	L	H	L	L	H	L	
414	L	H	L	L	L	H	L		424	L	L	H	H	H	L	L	
415	L	H	H	H	H	H	L		425	L	L	H	L	L	H	L	
416	L	H	L	L	L	H	L		426	L	L	H	L	L	H	L	
417	L	H	L	L	L	H	L		427	L	H	H	H	H	L	L	
418	L	L	L	L	L	L	L		428	L	L	L	L	L	L	L	

ASCII-Code des Buchstaben A

Abb. 10.14: Beispiel für den Inhalt eines Zeichengenerator ROMs zur Darstellung der Buchstaben A und B. 5x7 Zeichenmatrix mit 2 Leerzeilen zur Zeichentrennung.

Während der CRTC die Darstellung einer Zeichenreihe steuert, lädt er gleichzeitig in ein zweites Zeichenreihen-Register die ASCII-Codes der nächsten auszugebenden Zeichenreihe ein. Hierbei wird die Methode des direkten Speicherzugriffs angewandt (DMA). Die DMA-Steuerung macht die Bustreiber hochohmig und sendet eine entsprechende Adresse aus, so daß der angesprochene Speicherplatz des Bildwiederholspeichers seinen Inhalt - den ASCII-Code eines darzustellenden Zeichens - über den Datenbus an den Bildschirmsteuerbaustein gibt. In dem Bildwiederholspeicher müssen also die ASCII-Codes all derjenigen Zeichen stehen, die auf dem Bildschirm sichtbar gemacht werden sollen.

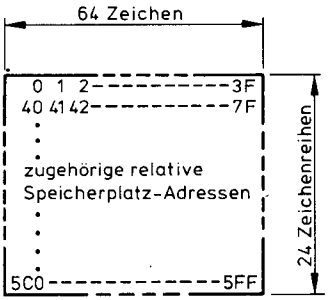


Abb. 10.15:
Zuordnung von Speicherplatzadressen zu Bildschirmzeichenpositionen

Bei der Darstellung von 24 Zeichenreihen zu je 64 Zeichen - wie in Abb.10.15 angedeutet - werden insgesamt 600H Speicherplätze benötigt. Die absoluten Adressen der den einzelnen Zeichen zugeordneten Speicherplätze ergeben sich aus der Anordnung des Bildwiederholspeicher-

RAMs im verfügbaren Adreßraumes des Mikrocomputers. Wenn die RAM-Anfangsadresse z.B. den Wert 3000H hat, so nimmt dieser Speicherplatz den ASCII-Code des Zeichens auf, welches ganz links oben auf dem Bildschirm erscheinen soll (Abb.10.15). Auf dem Speicherplatz 35FFH steht dann entsprechend der Code für das Zeichen ganz unten rechts.

Soll beispielsweise in der zweiten Zeichenreihe auf der ersten Position ein A sichtbar werden, so braucht man nur in den Speicherplatz mit der Adresse 3040H den ASCII-Code des Buchstaben A (41H) zu schreiben.

```
Beispiel:   LXI  H,3040H ; Adresse des Speicherplatzes
            MVI  M,'A'   ; Der Buchstabe A erscheint auf dem Bild-
                           schirm
```

Zum 'Löschen' des Bildschirmes - es wird kein Zeichen dargestellt - ist es nur erforderlich, in alle Speicherplätze des Bildwiederholspeichers den Wert 20H einzuschreiben, denn dies ist der ASCII-Code für das Leerzeichen (blank).

Durch Lesezugriffe auf den Bildwiederholspeicher ist es jederzeit möglich festzustellen, welche Zeichen momentan auf dem Bildschirm zu sehen sind.

Beispielschaltung

Unter Verwendung des Bildschirmsteuerbausteines 8275 wurde die in Abb.10.16 dargestellte Schaltung - zum Anschluß an einen MP 8085 - entwickelt.

Der Aufbau entspricht dem in Abb.10.13 beschriebenen Videoram. Von dem Bildwiederholspeicher wird der Adreßbereich 3000 bis 37FFH belegt. Ein Bildschirm-Format mit 1 bis 80 Zeichen pro Reihe und 1 bis 64 Reihen pro Bild ist programmierbar. Hier wurden 24 Zeichenreihen mit 64 Zeichen/Reihe festgelegt, wodurch der Bildwiederholspeicher eine Kapazität von 1536 Byte benötigt (24x64).

Als Zeichenmatrix findet die in Abb.10.14 dargestellte (5+2)x(7+2)-Matrix Verwendung.

Die Funktion des CRTC und des Zeichengenerators wurde bereits beschrieben.

Der 1 aus 8 Decoder (17) erzeugt die Bausteinauswahlsignale für den CRT- und den DMA-Controller. Im Zusammenhang mit den Adreßleitungen können dann über den Datenbus die internen Steuerregister geladen werden.

Die Bausteine (26) und (27) dienen der Modifizierung des Videosignales. Die Zeichen können hiermit blinkend, invers oder grau dargestellt werden.

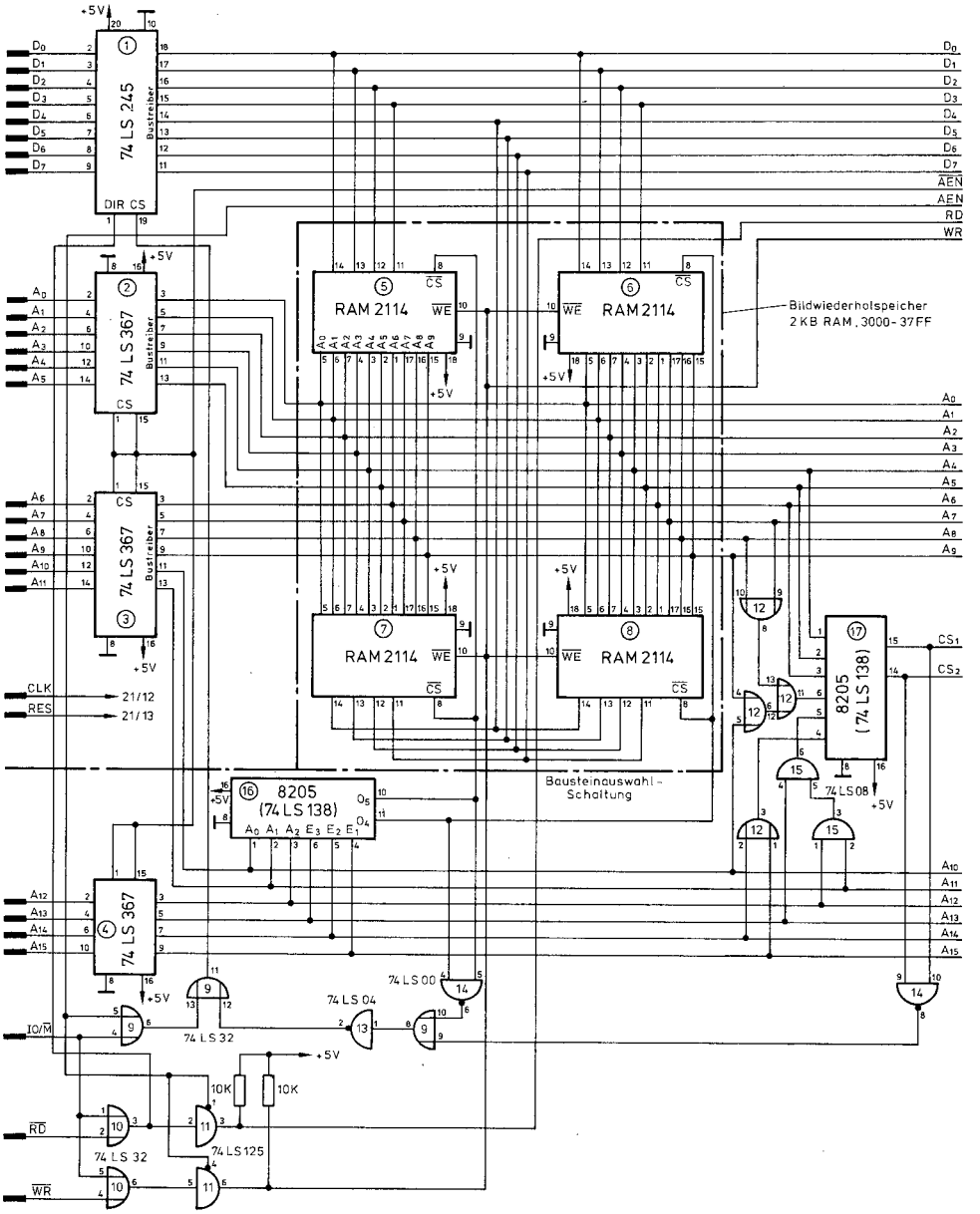
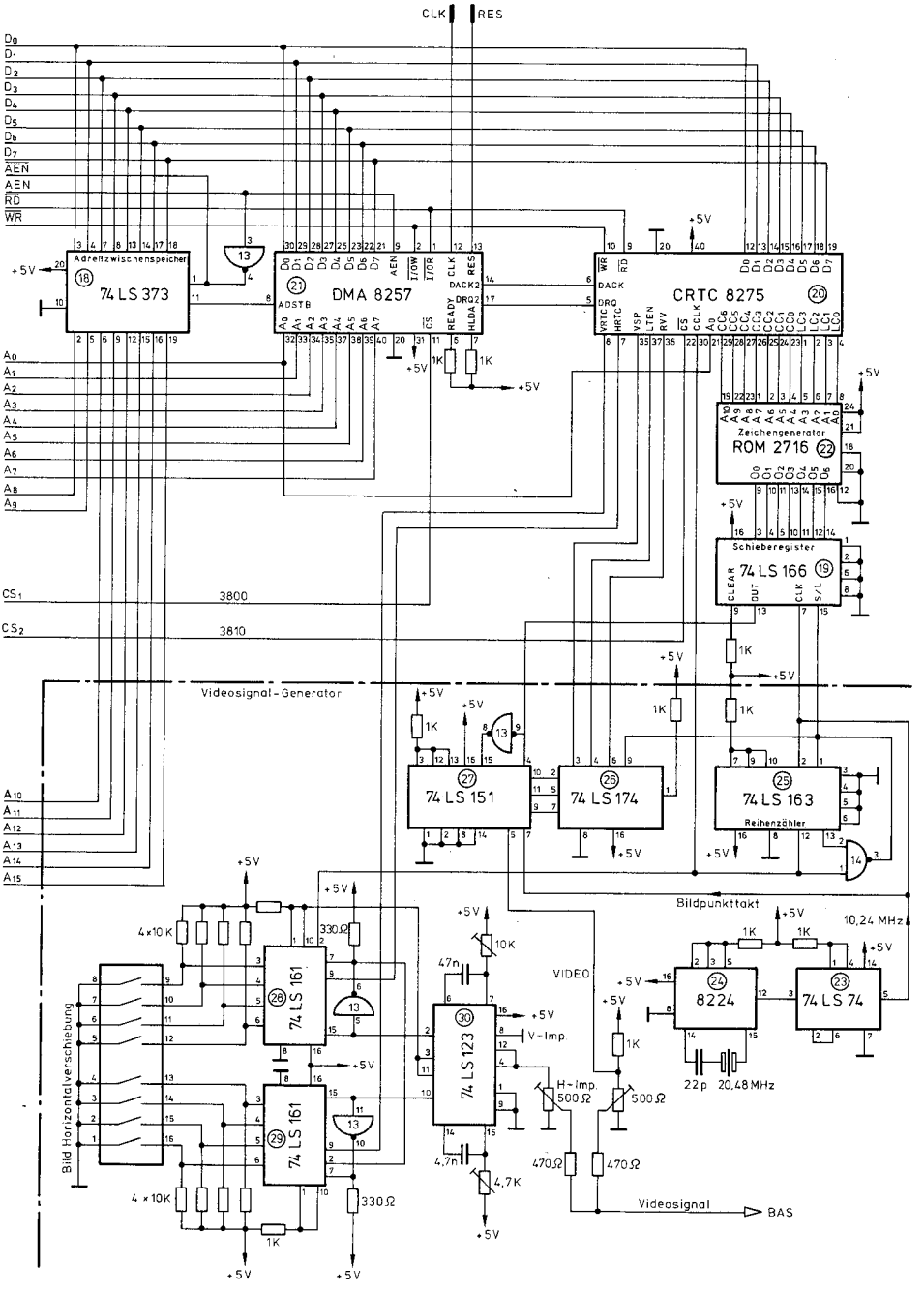


Abb. 10.16: Beispiel für den Aufbau eines sog. Videorams zum Anschluß an den MP 8085. Die zweite Hälfte der Schaltung ist auf der nächsten Seite dargestellt.



Mit der Baugruppe, die aus den Bausteinen (28) bis (30) und der Schaltergruppe besteht, kann das Bild horizontal justiert, sowie eine Verzögerung und eine Veränderung der Impulsbreite des V- und H-Signales erreicht werden.

Direkter Speicherzugriff:

Zur ständigen Auffrischung des Bildes, wird der Inhalt des Bildwiederholspeichers 50 mal pro Sekunde auf dem Bildschirm dargestellt. Bei einer Gesamtzahl von 1536 Zeichen pro Bild, bedeutet dies im Mittel 76800 Zugriffe auf den Bildwiederholspeicher pro Sekunde. Die Zugriffsfrequenz von Seiten des Mikroprozessors ist demgegenüber wesentlich geringer. Im Normalfall werden über eine Tastatur Zeichen in den Bildwiederholspeicher und damit auf den Bildschirm gebracht. Hierbei bleibt die Frequenz im allgemeinen deutlich unter 10 Zeichen (Zugriffe) pro Sekunde. Aus diesem Grunde werden in der vorliegenden Schaltung, durch den DMA-Controller bei einem Speicherzugriff, nur die Bustreiber hochohmig geschaltet und nicht, wie sonst üblich, der Mikroprozessor angehalten. Wenn dann der Mikroprozessor auf den Bildwiederholspeicher zugreifen will, muß vorher, durch das Programm, der DMA-Controller angehalten werden.

Die niederwertigen 8 Bits der vom DMA-Controller zu erzeugenden Adresse werden direkt herausgegeben, während die höherwertigen über die Datenbusleitungen zur Zwischenspeicherung an den Baustein (18) gegeben werden.

Minimalprogramm:

Ein einfaches Programm, zur Initialisierung des DMA- und CRT-Controllers, sowie zur Übertragung von Tastaturzeichen auf den Bildschirm, ist im folgenden wiedergegeben.

```
Beispiel: ; Initialisierung des CRTC 8275
          LXI   H,3811H    ;Adresse des Kommandoregisters
          MVI   M,0        ;Kommandoregister rücksetzen
          DCX   H          ;Adresse des Parameterregisters
          MVI   M,3FH      ;4 Parameter zur Festlegung des
          MVI   M,0D7H     ;Bildformates etc. übergeben.
          MVI   M,9AH
          MVI   M,4EH

          CALL  INIDMA     ;Initialisierung des DMAC 8257

          INX   H          ;Kommandoregister
          MVI   M,0E0H     ;Zähler voreinstellen
          MVI   M,2FH      ;Bildfreigabe

          ; Tastatur einlesen

          LXI   H,3000H    ;Anfangsadresse des Bildwiederholspeichers
    TAST: IN    22H        ;Tastatur einlesen
          CPI   0FFH       ;wurde eine Taste gedrückt?
          JZ    TAST
          ANI   7F         ;8. Bit auf L setzen (7bit-ASCII-Code)
```

```
        MOV   B,A        ;Zeichencode sichern
TASLOS: IN    22H
        CPI   OFFH       ;wurde die Taste losgelassen?
        JNZ   TASLOS
        CALL  DMASPE     ;DMA sperren
        MOV   M,B        ;Zeichencode in den Bildwiederholspeicher
        INX   H
        CALL  INIDMA     ;DMA freigeben
        JMP   TAST       ;nächstes Zeichen holen
        ; UP Initialisierung des DMAC 8257
INIDMA: PUSH  H
        LXI   H,3808H    ;Adresse des Betriebsartenregisters
        MVI   M,0        ;Betriebsartenregister rücksetzen
        LXI   H,3804H    ;Adresse für Kanal 2
        MVI   M,0        ;DMA-Adresse (3000H) laden
        MVI   M,30H
        INX   H
        MVI   M,OFFH     ;DMA-Blocklänge Kanal 2 laden
        MVI   M,85H
        INX   H          ;Adresse für Kanal 3 (3806H)
        MVI   M,0        ;Kanal 3 wie 2 laden
        MVI   M,30H
        INX   H
        MVI   M,OFFH
        MVI   M,85H
        INX   H          ;Adresse des Betriebsartenregisters (3808H)
        MVI   M,84H      ;Betriebsartenregister laden
        MVI   A,0F3H
        STA   35FFH      ;Bildsynchronisation
        POP   H
        RET
        ; UP DMAC Sperren
DMASPE: XRA   A          ;Akku=0
        STA   3808H
        RET
```

10.4 Drucker

Unter einem Drucker soll hier ein Mikrocomputer-Ausgabegerät verstanden werden, das dauerhaftes, für den Beobachter lesbares, Informationsmaterial erstellt. Die hierbei angewandten Techniken und Verfahren sind sehr vielfältig. Einige in der Großcomputertechnik eingesetzten Geräte, wie z.B. der Laserdrucker oder der Zeilendrucker sind für die Mikrocomputertechnik zu aufwendig.

Als Ausgabegeräte für Mikrocomputer finden vorallem Matrixdrucker, wie Nadel- Metallpapier- und Thermodrucker, aber auch Typendrucker, wenn es um eine besonders schöne Schrift geht, Anwendung. In der Anfangszeit wurden häufig Fernschreiber (Teletyp) als kombinierte Ein-/Ausgabe-Geräte verwandt.

Fernschreiber

Der 5-Kanal-Fernschreiber wird im deutschen Fernmeldenetz eingesetzt. Das dort zur Codierung benutzte Fünfer-Alphabet ist in Abb.10.17

Bit 3 4 / 2 1 0	0 0 B	0 0 Z	1 0 B	1 0 Z	0 1 B	0 1 Z	1 1 B	1 1 Z
0 0 0			≡	≡	E	3	A	-
0 0 1	T	5	L	)	Z	+	W	2
0 1 0	<	<	R	4	O	✣	J	♀
0 1 1	O	9	G		B	?	Ze	Ze
1 0 0	⌄	⌄	I	8	S	*	U	7
1 0 1	H		P	0	Y	5	Q	1
1 1 0	N	,	C	:	F		K	C
1 1 1	M	.	V	=	X	/	Bu	Bu

< = Wagenrücklauf
⌄ = Leerschritt
≡ = Zeilenvorschub
✣ = Anrufzeichen (Wer da ?)
♀ = Glocke
Ze = Walze auf Zeichen (Z) umschalten
Bu = Walze auf Buchstaben (B) umschalten

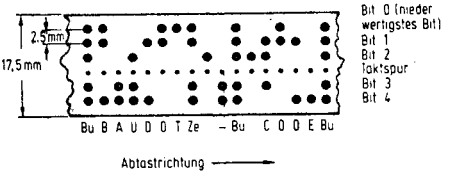


Abb. 10.17: Baudot-Code Tabelle und Realisierung auf einem 5 Kanal-Lochstreifen

dargestellt. Jedes darin vorkommende Zeichen besteht aus fünf Strom bzw. kein Strom (H bzw. L) Impulsen. Es sind somit 2^5 = 32 Kombinationen möglich. Da diese Anzahl von Kombinationen nicht ausreicht um alle Buchstaben, Ziffern und Sonderzeichen darzustellen, werden einige Kombinationen doppelt belegt. Eine Unterscheidung zwischen Buchstaben und Ziffern oder Sonderzeichen erfolgt durch die vorherige Übermittlung einer Ziffern/Buchstaben- oder Buchstaben/Ziffern-Umschaltung. Die Großschreibweise von Buchstaben ist nicht möglich. Da die verwendeten Fernschreiber Start-Stoppmaschinen sind, wird jeder Kombination ein Start und ein Stoppimpuls angefügt. Die Startimpulse sind L-Impulse (kein Strom), die Stoppimpulse H-Impulse (Strom).

Der Startimpuls sowie die fünf Codierimpulse haben eine Zeitdauer von 20ms. Der Stoppimpuls ist 1,5 mal so lang, also 30ms. Daraus resultiert die Zeit, die für die Übertragung eines Zeichens notwendig ist, nämlich 150ms. In einer Sekunde werden somit 6 2/3 Zeichen übertragen. Die Schreibgeschwindigkeit wird in der Fernschreibtechnik üblicherweise in Baud angegeben. Dies entspricht der Anzahl der pro Sekunde übertragenen Impulse. 6 2/3 Zeichen x 7,5 Impulse = 50. Die Schreibgeschwindigkeit beträgt also 50 Baud.

Anwendung finden aber auch Fernschreiber mit einer Schreibgeschwindigkeit von 100 Baud. Für die Dauer der Impulse ergeben sich somit entsprechend kürzere Zeiten.

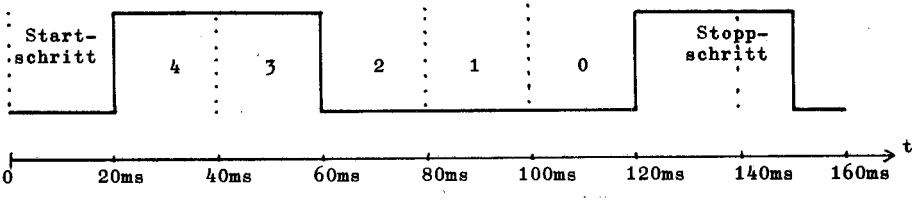


Abb. 10.18: Impulsfolge für den Buchstaben a bei 50 Baud

In der Regel besitzen Fernschreiber ein Tastenfeld, ähnlich dem einer Schreibmaschine. Über diese Tasten kann direkt mit der Gegenstelle korrespondiert, oder aber zunächst ein Lochstreifen erstellt und dessen Inhalt später ausgesandt werden.

Falls der Fernschreiber über einen Lochstreifenstanzer und -leser verfügt, können diese auch als MC- Ein-/Ausgabe-Geräte Anwendung finden.

Hier soll jedoch nur ein reines Empfangsgerät beschrieben werden.

Der Empfangsvorgang verläuft folgendermaßen. Durch den Startimpuls fallen alle fünf Anker des Empfangsmagneten ab. Dadurch wird die Empfangswelle für eine Umdrehung eingeschaltet. Auf dieser Welle sitzen Nocken, die der Reihe nach alle fünf Anker dem Empfangsmagneten anbieten. Wird ein Stromimpuls empfangen, zieht der Empfangsmagnet diesen Anker an und verriegelt ihn mechanisch. Wird kein Stromimpuls empfangen, fällt der Anker - nach Weiterdrehung der Empfangswelle - wieder ab. Über diese Anker werden fünf Empfangsschienen betätigt, welche Aussparungen haben, die eine durchgehende Nut, entsprechend dem Code, bilden. In diese Nut fällt eine Zugstange, die auch gleichzeitig den entsprechenden Typenhebel betätigt. Das entsprechende Zeichen wird auf dem Papier sichtbar. Empfängt der Fernschreiber ca. 30s keine Signale, wird der Antriebmotor abgeschaltet. Der Fernschreiber bleibt aber betriebsbereit.

Anpassung an den MP 8085:

Die Anpaßschaltung (interface) in Abb.10.19 dient der Ansteuerung eines Fernschreibers mit Hilfe des seriellen Ausganges des MP 8085 (SOD).

Der Optokoppler sorgt für eine galvanische Trennung zwischen den Stromkreisen des Mikrocomputers und des Fernschreibers; dies ist zur Vermeidung von Störungen unbedingt erforderlich. Der eingezeichnete

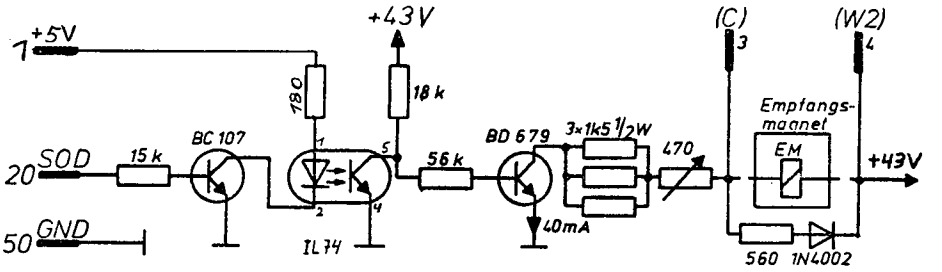


Abb. 10.19: Interface-Schaltung für den Anschluß einer Fernschreib-Empfangsstation an einem MP 8085.

Empfangsmagnet befindet sich im Fernschreiber und wird über die Anschlüsse 3 und 4 oder C und W2, des entsprechenden Steckers, mit der Interface-Schaltung verbunden.
Die parallel zum Empfangsmagneten geschaltete Diode soll Selbstinduktionsspannungen abbauen und der Widerstand Abfallverzögerungen verhindern. Der Transistor BD679 schaltet den Empfangsmagneten gegen Masse wenn ein H-Signal anliegt, wobei die zwischengeschalteten Widerstände zur Einstellung des durch den Empfangsmagneten fließenden Stromes auf 40mA dienen. Es ist zu beachten, daß die Signale durch die Schaltung invertiert werden d.h. L bei SOD bedeutet H am BD679 und umgekehrt. Dieses Verhalten muß insbesondere bei der Programmierung Berücksichtigung finden.

Wenn keine Signale übertragen werden, soll ständig der Linienstrom fließen, damit der Empfangsmagnet angezogen bleibt und somit ein ständiges Anbieten und Zurückfallen der Empfangsschienen verhindert; dieses führt darüber hinaus zu einer verminderten Geräuschentwicklung.

Wenn die in Abb.10.19 dargestellte Schaltung an einem MP 8080 angeschlossen werden soll, kann dort anstelle des SOD (serial output data) ein Portbaustein (Kap.9.2) benutzt werden.

Druckprogramm:

Im folgendem ist ein Unterprogramm wiedergegeben, welches 69 Zeichen, die codiert im ASCII-Code in einem Zeilenspeicher vorliegen, auf einem Fernschreiber ausgedruckt. Die Anfangsadresse des Zeilenspeichers muß dem Programm im Registerpaar HL übergeben werden. Das Unterprogramm verändert den Inhalt aller Register und Flags.

Zur Abspeicherung eines Kennbytes wird ein Speicherplatz benötigt. Der Speicherplatz hat die symbolische Adresse AD01. Über eine EQU-Anweisung kann ihm eine beliebige absolute Adresse zugeordnet werden.

Wenn der Zeilenspeicher - ein RAM-Bereich mit 69 Plätzen - im Hauptprogramm mit den ASCII-Codes der zu druckenden Zeichen gefüllt wird, ist darauf zu achten, daß, bei nicht vollständig gefüllter Zeile, der Zeilenrest mit Leerzeichen (20H) zu füllen ist. Bei der Steuerung des Druckvorganges durch das Unterprogramm, wird dann ein Wagenrücklauf und Zeilenvorschub gemacht, wenn der Rest der Zeile nur noch aus Leerzeichen besteht.

Die Tabelle ab 6118H enthält den Fernschreiber-Code, einschließlich des Start- und Stopp-Schrittes; außerdem wurde hier berücksichtigt, daß die Interface-Schaltung alle Signale invertiert.

Wenn das Programm mit einer anderen ORG-Adresse versehen werden soll, ist darauf zu achten, daß das niederwertige Byte der Adresse 'TABEL' den Wert 18H hat.

```
                ;Ausdruck einer aus 69 Zeichen bestehenden Zeile
                ;auf einem Fernschreiber.
                ;Anfangsadresse des Zeilenspeichers in HL.

0000                    ORG     6045H
6045 3E00               MVI     A,00H
6047 322620             STA     AD01    ;Kennbyte = 00 setzen
604A CDB260             CALL    UP01    ;UP Anlauf und Wagenrücklauf
604D D5                 PUSH    D
604E 28                 DB      28H;LDHI
604F 44                 DB      44H     ;<DE>=<HL>+44H = Zeilenendadr.
6050 4B                 MOV     C,E
6051 42                 MOV     B,D
6052 D1                 POP     D
6053 CDCA60             CALL    UP02    ;Prüfen auf Leerzeichen
6056 7E         SP0E:   MOV     A,M     ;Ein Byte vom RAM > A
6057 FE18               CPI     18H     ;<A> = Zeichen oder Buchstabe ?
6059 DAA560             JC      SP08    ;Sprung, wenn nicht
605C FE40               CPI     40H     ;<A> = Zeichen + Ziffer ?
605E DA7D60             JC      SP09    ;Sprung wenn ja
6061 F5                 PUSH    PSW     ;** <A>=Buchstabe **
6062 3A2620             LDA     AD01
6065 FE80               CPI     80H     ;Kennbyte für Buchstabe geladen?
6067 CA7260             JZ      SP0A    ;Sprung, wenn geladen
606A 3E80               MVI     A,80H
606C 322620             STA     AD01    ;Kennbyte laden
606F CDD560             CALL    UP03    ;UP: 5 bit Ausgabe
6072 F1         SP0A:   POP     PSW
6073 FE60               CPI     60H     ;Ist Großbuchstabe ?
6075 DA8F60             JC      SP0B    ;Sprung, wenn ja
6078 D620               SUI     20H     ;Kleinbuchstabe >> Großbuchstabe
607A C38F60             JMP     SP0B
607D F5         SP09:   PUSH    PSW
607E 3A2620             LDA     AD01
6081 FE90               CPI     90H     ;Kennbyte für Zeichen geladen ?
6083 CA8E60             JZ      SP0C    ;Sprung, wenn geladen
6086 3E90               MVI     A,90H
6088 322620             STA     AD01    ;Kennbyte laden
608B CDD560             CALL    UP03    ;UP: 5 bit Ausgabe
608E F1         SP0C:   POP     PSW
608F C5         SP0B:   PUSH    B
6090 011861             LXI     B,TABEL
6093 4F                 MOV     C,A
6094 0A                 LDAX    B
6095 C1                 POP     B
6096 CDD560     SP0F:   CALL    UP03    ;UP: 5 bit Ausgabe
6099 E5                 PUSH    H
609A 08                 DB      08;DSUB ;Endadresse erreicht ?
609B DAA060             JC      SP0D    ;Sprung, wenn nicht
609E E1                 POP     H
609F C9                 RET
60A0 E1         SP0D:   POP     H
60A1 23                 INX     H
60A2 C35660             JMP     SP0E
60A5 3E90       SP08:   MVI     A,90H
60A7 322620             STA     AD01    ;Kennbyte laden
60AA CDD560             CALL    UP03
60AD 3EA4               MVI     A,0A4H  ;A4H = Leerschritt
60AF C39660             JMP     SP0F

                ;**** UP Anlauf und Wagenrücklauf ****
60B2 3EC0       UP01:   MVI     A,0C0H  ;Abschalten des Linienstromes
60B4 30                 DB      30H;SIM
60B5 CDEE60             CALL    UP04    ;UP Verzögerung
60B8 3E40               MVI     A,40H   ;Einschalten des Linienstromes
```

```
60BA 30              DB      30H;SIM
60BB CDEE60          CALL    UP04
60BE 3EF4            MVI     A,0F4H  ;Ausgabe des Fünfercodes für
60C0 CDD560          CALL    UP03    ;Wagenrücklauf
60C3 CD0961          CALL    UP05    ;Zeitschleife 30 ms
60C6 CD0961          CALL    UP05
60C9 C9              RET

                     ;******* UP Prüfen auf Leerzeichen *******
60CA 0A              UP02:   LDAX    B
60CB FE20                    CPI     20H
60CD C2D460                  JNZ     SP10
60D0 0B                      DCX     B
60D1 C3CA60                  JMP     UP02
60D4 C9              SP10:   RET

                     ;******* UP 5 Bit Ausgabe *******
60D5 D5              UP03:   PUSH    D
60D6 1E06                    MVI     E,06H   ;Ausgabe des
60D8 07              SP11:   RLC
60D9 57                      MOV     D,A
60DA 3E80                    MVI     A,80H   ;Startschrittes und
60DC 1F                      RAR
60DD 30                      DB      30H;SIM ;des Fünfercodes
60DE CDFA60                  CALL    UP06    ;Zeitschleife 20 ms
60E1 7A                      MOV     A,D
60E2 1D                      DCR     E
60E3 C2D860                  JNZ     SP11
60E6 3E40                    MVI     A,40H   ;Ausgabe des Stoppschrittes
60E8 30                      DB      30H;SIM
60E9 CD0961                  CALL    UP05    ;Zeitschleife 30 ms
60EC D1                      POP     D
60ED C9                      RET

                     ;******* UP Verzögerung *******
60EE C5              UP04:   PUSH    B
60EF 0608                    MVI     B,08H
60F1 CD0961          SP12:   CALL    UP05    ;Zeitschleife 30 ms
60F4 05                      DCR     B
60F5 C2F160                  JNZ     SP12
60F8 C1                      POP     B
60F9 C9                      RET

                     ;******* UP Zeitschleife 20 ms, bei 3,072 MHZ  *******
60FA C5              UP06:   PUSH    B
60FB 0611                    MVI     B,11H
60FD 0EFF            SP14:   MVI     C,0FFH
60FF 0D              SP13:   DCR     C
6100 C2FF60                  JNZ     SP13
6103 05                      DCR     B
6104 C2FD60                  JNZ     SP14
6107 C1                      POP     B
6108 C9                      RET

                     ;******* UP Zeitschleife 30 ms, bei 3,072 MHZ  *******
6109 D5              UP05:   PUSH    D
610A 1E1A                    MVI     E,1AH
610C 16FA            SP16:   MVI     D,0FAH
610E 15              SP15:   DCR     D
610F C20E61                  JNZ     SP15
6112 1D                      DCR     E
6113 C20C61                  JNZ     SP16
6116 D1                      POP     D
6117 C9                      RET
```

```
6118 94A4A4A4  TABEL: DB      94H,0A4H,0A4H,0A4H,0A4H
611C A4
611D D0A4A4EC        DB      0D0H,0A4H,0A4H,0ECH
6121 A4A4B4A4        DB      0A4H,0A4H,0B4H,0A4H
6125 E8A4AC84        DB      0E8H,0A4H,0ACH,84H,0D8H
6129 D8
612A A4B8E49C        DB      0A4H,0B8H,0E4H,9CH,0E0H
612E E0
612F A0C88898        DB      0A0H,0C8H,88H,98H,0BCH
6133 BC
6134 D4F8A88C        DB      0D4H,0F8H,0A8H,8CH,0CCH
6138 CC
6139 F0C4A4A4        DB      0F0H,0C4H,0A4H,0A4H,0C0H
613D C0
613E A4B0EC9C        DB      0A4H,0B0H,0ECH,9CH,0B0H
6142 B0
6143 C4B4BCA4        DB      0C4H,0B4H,0BCH,0A4H,0D0H
6147 D0
6148 E8CC9484        DB      0E8H,0CCH,94H,84H,0D8H
614C D8
614D E0E4F0C8        DB      0E0H,0E4H,0F0H,0C8H,88H
6151 88
6152 D4ACF88C        DB      0D4H,0ACH,0F8H,8CH,0C0H
6156 C0
6157 98A0A8B8        DB      98H,0A0H,0A8H,0B8H,0ECH
615B EC
615C ECECECEC        DB      0ECH,0ECH,0ECH,0ECH

2026           AD01  EQU     2026H   ;Speicher für Kennbyte
0000                 END
```

SYMBOL TABLE

AD01	2026	SP08	60A5	SP09	607D	SP0A	6072
SP0B	608F	SP0C	608E	SP0D	60A0	SP0E	6056
SP0F	6096	SP10	60D4	SP11	60D8	SP12	60F1
SP13	60FF	SP14	60FD	SP15	610E	SP16	610C
TABEL	6118	UP01	60B2	UP02	60CA	UP03	60D5
UP04	60EE	UP05	6109	UP06	60FA		

Matrixdrucker

Ein Matrixdrucker hat keine einzelnen Typen, wie beispielsweise eine Schreibmaschine, sondern erzeugt auf dem Papier einzelne Punkte, aus denen sich, ähnlich wie auf einem Bildschirm, das Zeichen zusammen setzt.

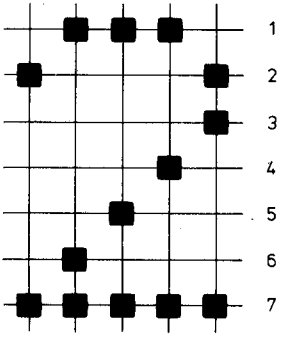


Abb. 10.20:
5x7 Matrix zur Darstellung der Zahl 2 mittels eines Druckkopfes mit 7 senkrechten Stiften, die von links nach rechts über das Papier laufen.

Nadeldrucker:

Bei einem Nadeldrucker werden die Punkte der Matrix, aus denen sich ein Zeichen zusammen setzt, durch eine Reihe von beweglichen Stiften erzeugt.

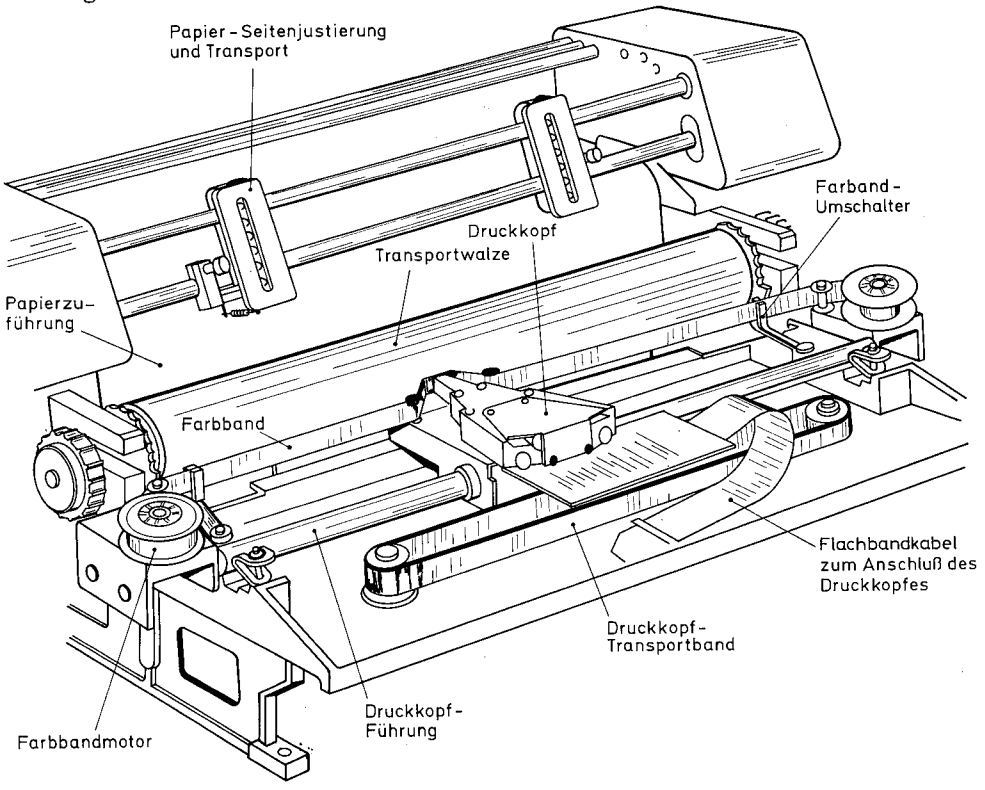


Abb. 10.21: Aufbau eines Nadeldruckers

In Abb.10.21 sind die wesentlichen Teile eines Nadeldruckers dargestellt und bezeichnet. Der Druckkopf enthält sieben oder neun übereinanderliegende Stifte, die durch Elektromagnete gegen das Farbband geschossen werden können. Hierdurch entsteht dann ein Punkt auf dem Papier das unter dem Farbband liegt.

Während der Druckkopf sich mit gleichbleibender Geschwindigkeit von links nach rechts bewegt, sorgt die elektronische Steuerung dafür, daß die entsprechenden Stifte, zur rechten Zeit, auf das Farbband schlagen. Zur Erzeugung der Zahl 2, gemäß Abb.10.20, müssen nacheinander die Stifte 2+7, 1+6+7, 1+5+7 u.s.w. angesteuert werden. Allein durch Veränderungen in der Ansteuerung des Druckkopfes, sind unterschiedliche Schriftbilder erzielbar.

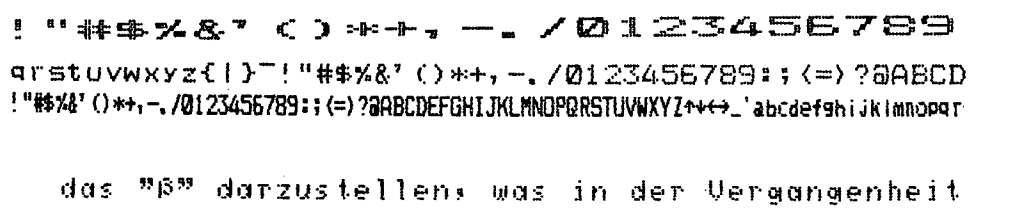


Abb. 10.22: Schriftbilder von Nadeldruckern. Zeile 1 bis 3: 5x7 Matrix, gleicher Drucker, Zeile 4: 5x9 Nadeldrucker.

Die drei oberen Schriftzeilen in Abb.10.22 wurden mit dem selben Drucker erzeugt.

Jedesmal, nachdem einige der Druckstifte betätigt worden sind, wartet die Steuerung eine kurze Zeit, um dann die entsprechenden Druckstifte, zur Erzeugung der nächsten Punktspalte eines Zeichens, anzusprechen.

Eine Verkürzung dieser Wartezeit erzeugt aus dem Druckbild der zweiten Zeile das der dritten Zeile, in Abb.10.22.

Neben der bekannten 5x7 Matrix, zur Darstellung der Zeichen, wird manchmal auch eine 5x9 oder sogar eine 7x9 Matrix verwandt. Dies insbesondere, um eine bessere Lesbarkeit der Kleinbuchstaben zu erreichen (Abb.10.22 unterste Zeile).

Zur Steuerung des Druckvorganges in modernen Druckern, werden meistens Mikrocomputer eingesetzt.

Große Druckgeschwindigkeiten (200 Zeichen/s) sind erreichbar, wenn der Druckkopf beim Rücklauf - von rechts nach links - sofort die nächste Zeile schreibt. Hierzu ist es natürlich erforderlich, daß dem Drucker der Inhalt der nächsten Zeile bereits bekannt ist, da die Zeile und die Zeichen rückwärts geschrieben werden müssen. Ein moderner Drucker enthält deshalb einen Speicher für den Zeichencode einer oder mehrerer zu druckender Zeilen. Diese Zwischenspeicherung, der Druckdaten im Mikrocomputer des Druckers, hat darüber hinaus den großen Vorteil, daß der Mikrocomputer, an den der Drucker angeschlossen wird, während des Druckvorganges noch andere Aufgaben erledigen kann, denn die Übertragung der Daten für einige Druckzeilen erfolgt sehr schnell.

Beispiel: 1KB (1024 Zeichen), Zwischenspeicher im Drucker.
80 Zeichen/s Druckgeschwindigkeit.
80KB/s Datenübertragungsgeschwindigkeit (Abb.10.26).
Wenn der Zwischenspeicher leer ist - es wurde alles gedruckt - fordert der Drucker neue Daten an - z.B. über einen Interrupt -.
Datenübertragungsdauer = 1KB/(80KB/s) = 12,5ms
Druckdauer = 1024Z/(80Z/s) = 12,8s
Zeitverhältnis Datenübertragung/Druck = 1:1024

Metallpapier- und Thermodrucker:

Auch die Metallpapier- und Thermodrucker erzeugen einzelne Punkte auf dem Papier, aus denen sich dann die Zeichen (Abb.10.20) zusammensetzen. Hierzu wird jedoch kein Farbband benötigt, sondern spezielle Papiersorten. Beim Metallpapierdrucker besteht der Druckkopf aus Nadeln, die ständig auf dem mit Metall beschichteten Papier aufliegen. Wenn ein Punkt der Matrix 'gedruckt' werden soll, wird eine Spannung von ca. 40V - gegenüber der Metallschicht - an die entsprechende Nadel gelegt und auf diese Weise ein Loch in der Schicht erzeugt, so daß die darunter liegende schwarze Papierschicht sichtbar wird.

Der Thermodrucker hat einen Druckkopf mit elektrisch beheizbaren Punkten, die auf einem Spezialpapier aufliegen. Wird ein Punkt erhitzt, so verflüchtigt sich ein kleiner Bereich in der Beschichtung des Spezialpapieres und die darunter liegende schwarze Schicht wird sichtbar.

Drucker als Zeichengerät:

Ein Matrixdrucker kann auch zur Anfertigung graphischer Darstellungen benutzt werden. Hierzu ist es erforderlich, daß jeder Druckstift einzeln ansteuerbar ist. Es lassen sich dann Graphiken erstellen, die aus lauter einzelnen Punkten bestehen (dot plotting). Computer-Peripherie-Geräte zur Anfertigung von Zeichnungen und Graphiken, werden meistens mit dem englischen Namen "Plotter" bezeichnet.

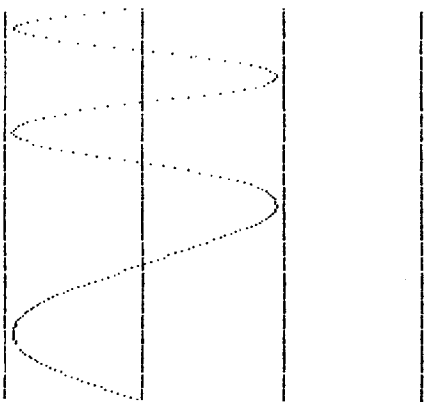


Abb. 10.23:
Graphische Darstellungen mit Hilfe eines Nadeldruckers (dot plotting). Hier wurde die durch einen Analog/Digital-Wandler analysierte Schwingung durch einen Nadeldrucker dargestellt.

Ein Beispiel für solch eine Anwendung zeigt Abb.10.23. Über einen, an einem Mikrocomputer angeschlossenen, Analog/Digital-Wandler, wurde ein Sinussignal variabler Frequenz analysiert, die Werte abgespeichert und dann das Ergebnis auf dem Nadeldrucker graphisch dargestellt.

Die Realisierung solcher Darstellungen ist sehr einfach, wenn die sieben Stifte eines Nadeldruckers, beispielsweise durch die Übermittlung einer 7bit-Zahl, ansteuerbar sind. Jedem Stift ist dann ein Bit zugeordnet und wenn es den Wert H hat erzeugt dieser Stift einen Punkt auf dem Papier.

Soll etwa auf diese Weise die Zahl 2 (Abb.10.20) dargestellt werden, sind die folgenden Zahlen nacheinander an den Drucker zu senden: LHLLLLH,HLLLLHH,HLLLHLH,HLLHLLH,LHHLLLH.

Auch Metallpapier- und Thermodrucker lassen sich als Plotter benutzen, wobei es bei einem Thermodrucker sogar möglich ist, durch eine Variierung der erzeugten Wärme, Halbtöne, d.h. unterschiedliche Grauwerte zu erzeugen.

Abb. 10.24:
Darstellung von Halbtonbildern durch einen Thermodrucker

Druckeranschluß

Wie bereits in Kapitel 9 ausgeführt wurde, kann zwischen einer seriellen und einer parallelen Schnittstelle gewählt werden. Wobei mit einer parallelen Schnittstelle eine größere Datenübertragungsgeschwindigkeit - bei einem höheren Leitungsaufwand - erreichbar ist.

Serielle Schnittstelle:

Die am häufigsten anzutreffende serielle Schnittstelle, zum Anschluß von Druckern, Plottern und Bildschirmgeräten an einen Mikrocomputer, ist die sog. RS232C- oder auch die sog. V24-Schnittstelle.

Zum Anschluß werden meistens 25 polige Steckverbinder der Serie D benutzt.

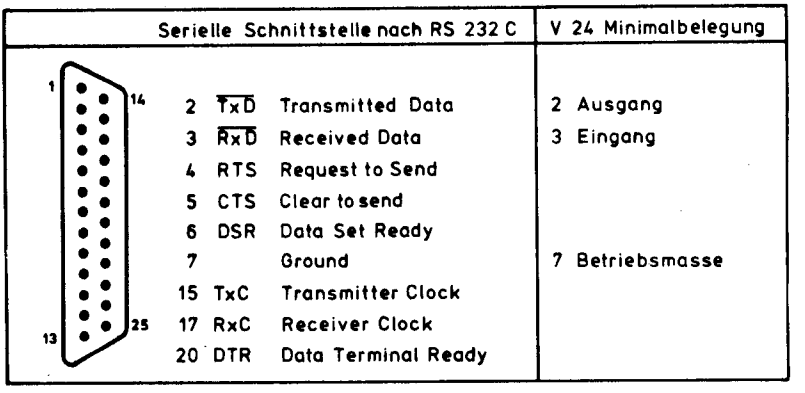


Abb. 10.25: Serielle Schnittstelle zum Anschluß von Peripherie-Geräten

Parallele Schnittstelle:

Bei einer Parallelschnittstelle wird meistens 1 Byte parallel übertragen, oder aber auch nur die 7 Bits des ASCII-Codes.

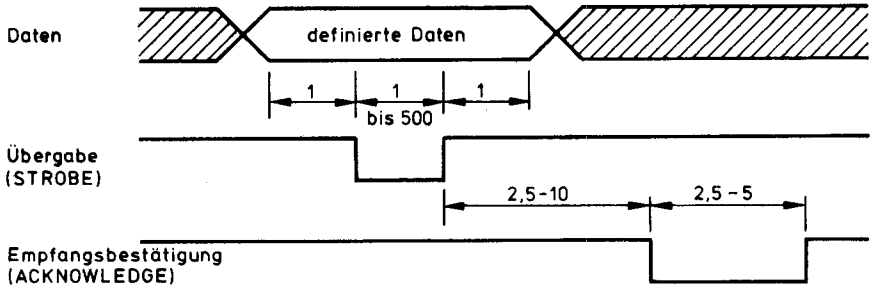


Abb. 10.26: Timing einer Parallelschnittstelle (Centronics). Die eingetragenen Zeiten stellen Richtwerte in Mikrosekunden dar.

Zunächst werden der ASCII-Code des zu übertragenden Zeichens (Daten) und ein Übergabeimpuls (STROBE) ausgegeben. Anschließend wartet der Mikrocomputer auf die Empfangsbestätigung (ACKNOW) durch den Drucker und sendet dann das nächste Zeichen.

Ist ein Drucker beispielsweise an einem Portbaustein 8255 (Kap.9.1) angeschlossen, so kann die Übergabe, des 7bit-ASCII-Codes eines Zeichens, durch das folgende Programm geschehen. Voraussetzung hierzu ist, das an PA_0 bis PA_6 die sieben Dateneingangskanäle, an PA_7 der STROBE-Kanal und an PC_0 der ACKNOWLEDGE-Ausgang angeschlossen wurde.

```
;                DRUCKER MIT PARALLELSCHNITTSTELLE
;ASCII-Code eines Zeichens in A übergeben und an den Drucker senden

        PUSH    PSW
ACKN:   IN      PC          ;Einlesen des ACKNOWLEDGE-Kanales auf PC0
        RRC
        JNC     ACKN        ;Sprung, wenn ACKN.=H, (keine neuen Daten senden)
        POP     PSW         ;Auszusendende ASCII-Code in den Akku laden
        ADI     80H         ;Das 8.Bit H setzen, da hieran der STROBE-Kanal
                            ;angeschlossen ist.
        OUT     PA          ;Ausgabe des Zeichencodes (7bit) und STROBE = H
        ANI     7FH         ;8. Bit L setzen
        OUT     PA          ;STROBE=L
        ADI     80H         ;8. Bit H setzen
        OUT     PA          ;STROBE=H
ACKNO:  IN      PC          ;Einlesen von ACKNOWLEDGE auf PC0
        RRC                 ;ACKN.=L bedeutet: das Zeichen wurde angenommen
        JC      ACKNO       ;Sprung, wenn noch kein Annahmesignal
        RET
```

10.5 Massenspeicher

Massenspeicher in Mikrocomputersystemen benutzen durchweg magnetische Speicherverfahren. Die gebräuchlichsten Speichermedien sind die Magnetbandkassetten und die Magnetplatten. In beiden Fällen gibt es eine Reihe unterschiedlicher technischer Ausführungsformen.

Tonkassetten

Die Kassettenrecorder der Unterhaltungselektronik, mit den dazu gehörenden Tonkassetten, stellen die einfachste und billigste Form von Magnetbandkassetten-Speichern dar.

Zur Aufzeichnung digitaler Informationen auf ein solches Magnetband, können verschiedene Verfahren angewandt werden. Hierzu zwei Beispiele:

- Dem H-Pegel wird ein bestimmter Ton zugeordnet und dem L-Pegel gar kein Ton.
- Der H-Pegel wird durch einen hohen und der L-Pegel durch einen niedrigen Ton repräsentiert.

Insbesondere das zweite Verfahren wird häufig angewandt und ist unter dem Namen Kansas-City-Standard bekannt geworden. Das Kansas-City Format ordnet dem Binärzeichen H 8 Schwingungen von 2400Hz und dem L 4 Schwingungen von 1200Hz zu.

Die Erzeugung dieser Schwingungen kann durch ein Programm vorgenommen werden. Hierbei tritt eine Datenübertragungsrate von 300 Baud auf.

Zum Lesen der Bandinformationen wird eine spezielle Schaltung verwandt. Dieser sog. FSK-Demodulator (frequenz-shift-keying = Frequenzumtastverfahren) erzeugt, aus den vom Band gelieferten Schwingungen, wieder die logischen Signale H oder L und leitet sie über einen Eingabekanal an den Mikroprozessor weiter.

Als E/A-Kanäle bieten sich beim MP 8085 beispielsweise der SID-Eingang und der SOD-Ausgang an.

Datenmagnetbandkassetten

Es gibt eine Reihe spezieller Magnetbandkassetten, für die Datentechnik, mit dazugehörigen Speziallaufwerken.

Die Vorteile dieser Systeme bestehen vorallem in einer höheren Aufzeichnungsgeschwindigkeit und einer größeren Datendichte auf dem Band.

Zur Aufzeichnung der Daten werden keine Töne mehr benutzt, sondern eine Reihe unterschiedlicher Verfahren, wovon die 6 gebräuchlichsten im DIN-Blatt 66010 dargestellt sind.

Allen Verfahren ist gemeinsam, daß zur Abspeicherung eines Binärzeichens auf dem Band höchstens ein magnetischer Flußwechsel erzeugt wird.

Dies geschieht, indem ein Strom durch den Schreibkopf fließt, welcher einen magnetischen Streufluß hervorruft, der die Magnetschicht des vorbeilaufenden Bandes bis zur Sättigung magnetisiert. Wird die Richtung des Stromes durch den Schreibkopf umgepolt, so wechselt auch die magnetische Flußrichtung.

Beim Lesen induziert ein Flußrichtungswechsel in dem Lesekopf eine Spannung. Die Polarität der Spannung hängt von der Änderungsrichtung (N-S oder S-N) des magnetischen Flusses ab.

Stellvertretend für die unterschiedlichen Aufzeichnungsverfahren, soll hier kurz auf das weit verbreitete Verfahren der sog. Richtungstaktschrift (phase encoding-PE) eingegangen werden.

Dieses Schreibverfahren findet beispielsweise auch Anwendung in der von der ECMA (European Computer Manufacturer Association) herausgegeben Norm - ECMA-34 - für die digitale Datenaufzeichnung auf Kassetten, mit den gleichen Abmessungen wie die bekannten Tonkassetten.

Grundzüge der Richtungstaktschrift:

Bei der Richtungstaktschrift wird zur Codierung eines Bits auf einem kleinen Bereich des Magnetbandes (Spurelement) eine Nord- und eine Südmagnetisierung vorgenommen. Dies bedingt, daß pro Spurenelement ein Flußwechsel stattfindet. Je nachdem, ob dieser Flußwechsel von Nord nach Süd oder entgegengesetzt erfolgt, wird ihm einer der beiden Binärzeichen fest zugeordnet. Wenn die Information aus alternierenden Bits (L/H/L/H...) besteht, ergeben sich die Flußwechsel automatisch (Abb.10.27); bei gleichbleibenden Bits (L/L oder H/H sind dagegen sog. Phasenflußwechsel einzufügen.

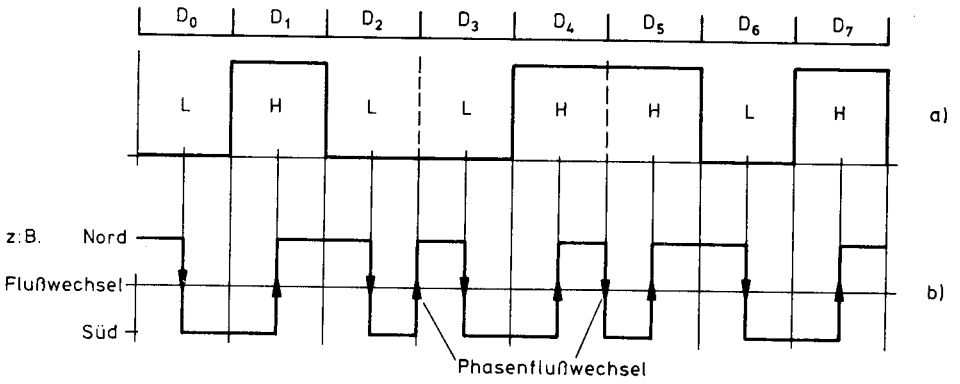


Abb. 10.27: Richtungstaktschrift

Die Umformung der Bit-Darstellung in Abb.10.27, in die entsprechende Flußdarstellung (Abb.10.27b), ist durch ein Programm leicht vollziehbar; indem beispielsweise der Flußrichtung Nord ein H und Süd ein L zugeordnet wird. Danach muß dieser unipolare Signalfluß noch in einen bipolaren umgewandelt und dem Schreibkopf zugeführt werden.

Diese Software-Lösung bietet sich an, wenn die Datenübertragungsrate - wie im folgenden Beispiel - nicht zu hoch ist.

Beispiel: Mini-DCR der Firma Philips

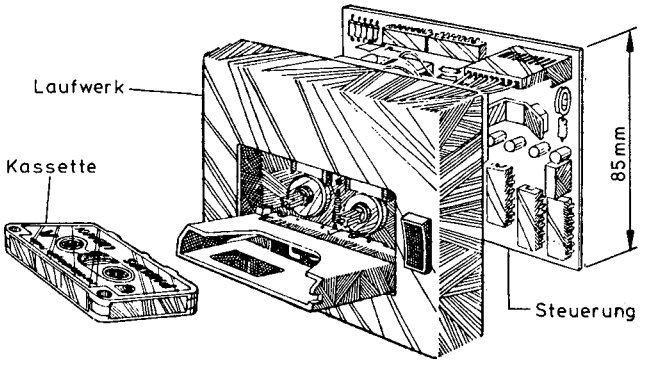


Abb. 10.28: Kassettenlaufwerk (Mini DCR der Firma Philips)

Das in Abb.10.28 abgebildete Kassettenlaufwerk verwendet kleine Spezialkassetten mit einem 3,81mm breiten und 36m langen Magnetband. Es wird im Halbspurverfahren beschrieben und kann deshalb durch einfaches Umdrehen, wie bei Tonkassetten, in beiden Richtungen zur Aufzeichnung von Daten benutzt werden. Die Schreib-, Lese- oder Umspulzeit beträgt einheitlich 90s. Bei einer Datenübertragungsrate von

6000bit/s ergibt sich eine Aufzeichnungsdichte von 15bit/mm und eine Bandkapazität von 2 x ca. 64KB.
Das Laufwerk wurde über einen Portbaustein 8255 an einen MP 8085 angeschlossen (Abb.10.29).
Der TTL-Baustein 7405 enthält invertierende Treiber mit offenem-Kollektor-Ausgang. Durch pull-up-Widerstände nach + 12V, werden die + 5V-TTL-Signale an die 12V-Pegel des Laufwerkes angepaßt. Der CMOS-Baustein 4010 wandelt die vom Laufwerk abgegebenen 12V-Signale in TTL-Signale um.

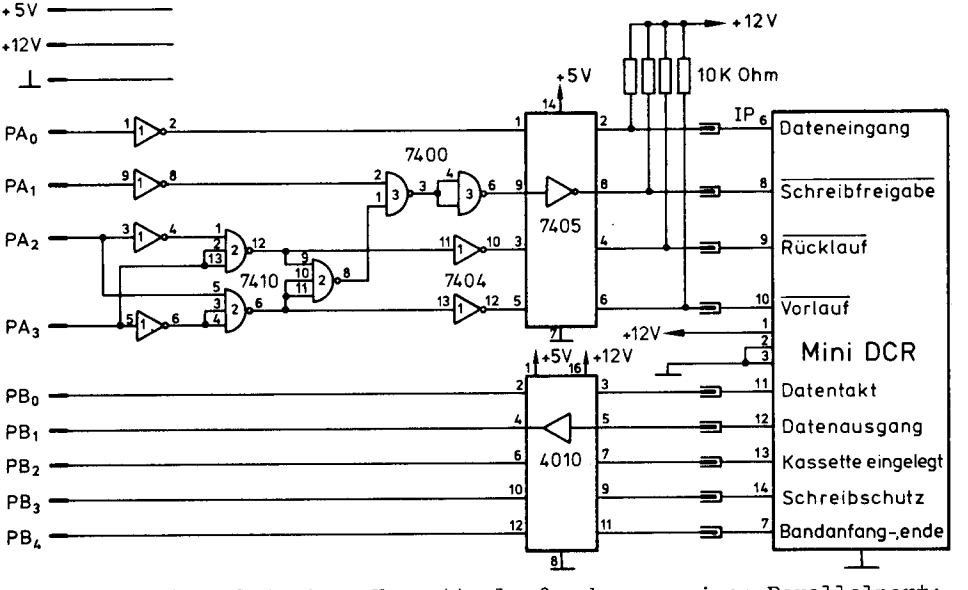


Abb. 10.29: Anschluß eines Kassettenlaufwerkes an einen Parallelport; mit Pegelwandler und Verriegelungsschaltung

Die zusätzlichen einzelnen Gatter sind zur Verriegelung der Laufwerkseingänge erforderlich; damit, bei unbestimmten Zuständen auf den Portleitungen, Fehlfunktionen des Laufwerkes, insbesondere ein unbeabsichtigtes Überschreiben, nicht eintreten können.Dies ist ein Beispiel für den Anschluß eines Peripherie-Gerätes mit seriellem Ein-/Ausgang an ein Parallelport, wobei die weiteren Kanäle des Ports Steuerungsaufgaben übernehmen.
Zur Aufzeichnung der Daten in der Richtungstaktschrift, gibt ein spezielles Schreibprogramm - über den Portkanal PA_0 - Impulse, gemäß Abb.10.27a, an den Dateneingang des Laufwerkes. Besondere Beachtung muß dabei die Schreibfrequenz von 6Kbit/s finden, da die aufgezeichneten Signale beim Lesen - durch eine spezielle Schaltung des Laufwerkes, die auf 6kbit/s eingestellt ist - für die Weitergabe an den Mikroprozessor aufbereitet werden.

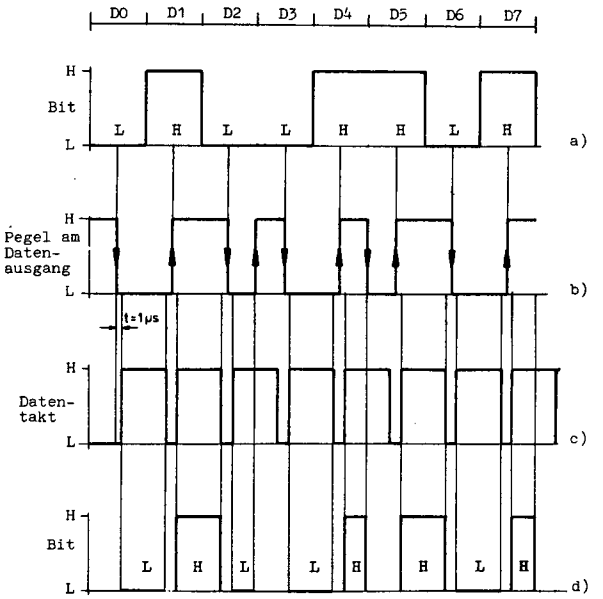


Abb. 10.30: Lesesignale bei der Richtungstaktschrift

Falls das in Abb.10.27a dargestellte Byte aufgezeichnet wurde und dann gelesen werden soll, ergeben sich die in Abb.10.30 zu sehenden Signale. Der Datenausgang zeigt die gleichen H/L-Pegel, wie in Abb. 10.27b Nord/Süd-Magnetisierungen fürs Schreiben dargestellt wurden. Damit der Mikroprozessor weiß, wann am Datenausgang des Laufwerkes ein neues Bit erscheint, gibt es zusätzliche Datentaktsignale; hierdurch sind die Daten für den Mikroprozessor leicht einlesbar.

Diskettenspeicher

Eine Diskette (floppy-disk) besteht aus einer flexiblen Mylar-Folie, die mit einer magnetisierbaren Schicht überzogen ist und in einer Plastikhülle steckt (Abb.10.31).

Während die Diskette einen Durchmesser von 8" hat, gibt es außerdem sog. Minidisketten mit einem Durchmesser von 5,25". Zum magnetischen Beschreiben und Lesen der (Mini) Disketten werden entsprechende (Mini) Diskettenlaufwerke (floppy disk drive) - oft auch kurz Floppy genannt - benutzt.

Die Diskette, bestehend aus der Magnetscheibe und der Hülle, wird komplett in das Laufwerk eingeführt. Dort dreht sich die Magnetscheibe innerhalb der Hülle, wobei der Schreib-/Lesekopf - durch die längliche Öffnung in der Diskettenhülle - auf die Magnetscheibe aufgesetzt werden kann.

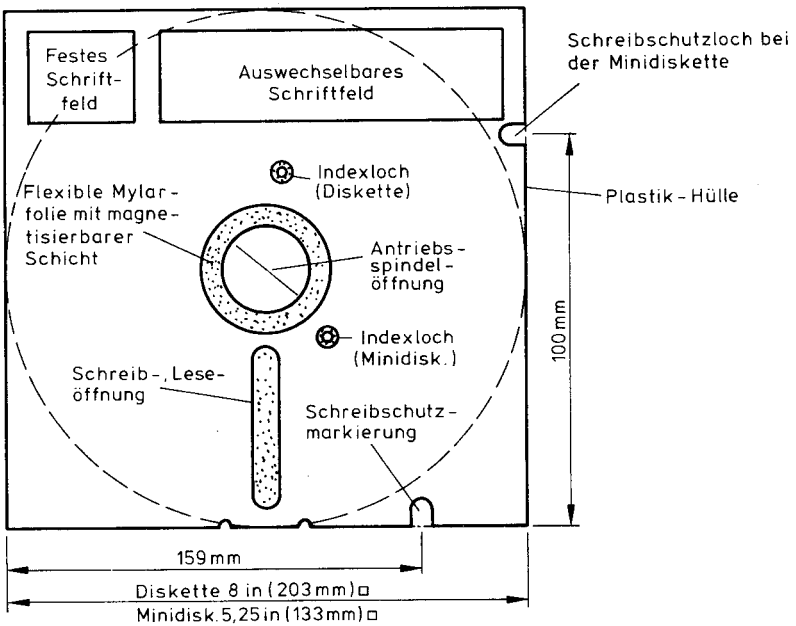


Abb. 10.31: Aufbau einer Diskette bzw. Minidiskette.

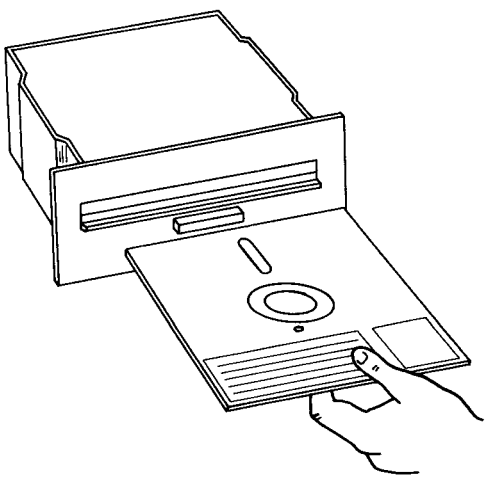


Abb. 10.32: Einführen einer Diskette in ein Diskettenlaufwerk (Floppy)

Die Plazierung der Daten erfolgt auf konzentrischen Spuren, mit Hilfe eines beweglichen Schrei-/Lesekopfes (Abb.10.33). Der Schreib-/Lese-kopf kann - gesteuert durch einen Schrittmotor - z.B. 77 verschiedene radiale Positionen einnehmen. In jeder dieser Positionen wird eine kreisförmige Spur beschrieben oder gelesen, wobei der Kreisumfang nochmals in sog. Sektoren unterteilt ist.

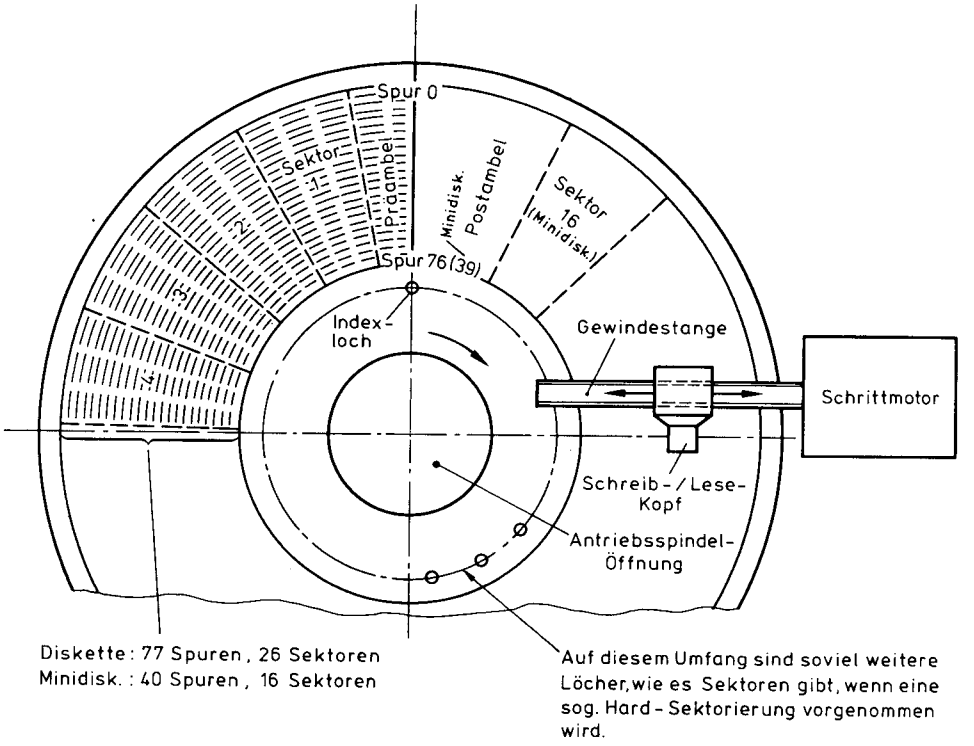


Abb. 10.33: Eine in Spuren und Sektoren unterteilte Diskette mit Schreib-/Lesevorrichtung. Präambel (= Datenvorspann, Postampel = Nachspann (Abb. 10.34)).

Formatierung:

Das Indexloch kennzeichnet den Beginn des 1. Sektors. Zur Kennzeichnung der weiteren Sektoren gibt es zwei verschiedene Methoden, die Soft- und die Hard-Sektorierung (Programm- oder Loch-Sektorierung). Bei der Hard-Sektorierung sind neben dem Indexloch noch soviel weitere Löcher gleichmäßig auf dem Umfang verteilt, wie es Sektoren auf der Diskette gibt. Durch eine Lichtschranke, die diese Löcher abtastet, wird jeder Sektor - 18 bzw. 32 bei einer Minidiskette bzw. einer Diskette - erkannt.
Bei der Soft-Sektorierung werden dem gegenüber nur 16 bzw. 26 Sektoren auf einem Umfang untergebracht. Da in beiden Fällen pro Sektor 128 Byte Speicherkapazität erreicht wird, hat eine hardsektorierte Diskette eine etwas höhere Gesamtspeicherkapazität.
Im Fall der Soft-Sektorierung hat der Benutzer die Möglichkeit, über die Anzahl der Sektoren frei zu bestimmen. Üblich ist jedoch das IBM3740-Format - oder leichte Abwandlungen davon - mit einer Anzahl von 16 bzw. 26 Sektoren (Minidisk./Diskette) und 128 Datenbytes pro Sektor (bei einfacher Schreibdichte). Zur eindeutigen Identifizierung

eines jeden Sektors, sind diese durch Aufzeichnungslücken (gap) voneinander getrennt und besitzen jeweils einen Vorspann mit Kennzeichnungsinformationen.

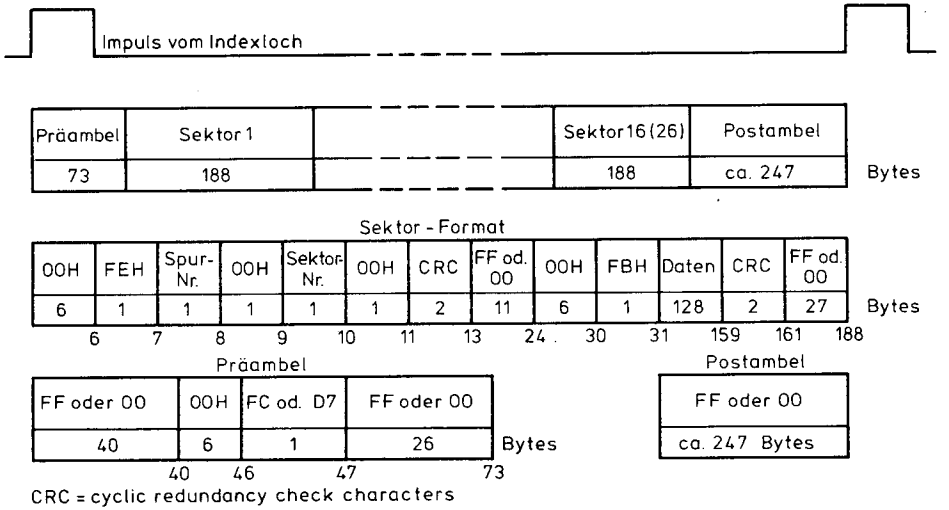


Abb. 10.34: Diskettenaufzeichnungsformat IBM 3740

Anhand des in Abb.10.34 dargestellten Formates sind die wesentlichen Anforderungen erkennbar.
Die Präambel steht am Anfang jeder Spur und beginnt mit der steigenden Flanke des von der Indexlichtschranke abgegebenen Impulses. Sie dient der Synchronisation und Datentrennung; insbesondere die sechs Nullbytes (41. bis 46.) stellen die klassische Form hierfür dar.

Jeder Sektor besteht einheitlich aus 188 Bytes. Die ersten 13 Bytes des Sektors sind zu seiner Identifizierung (ID field) da. Neben der Spur- und Sektornummer, enthält dieser Bereich z.B. noch zwei CRC-Prüfbytes, zur Kontrolle der richtigen Datenübernahme aus diesem Feld. Nach einer 11 Bytes langen Lücke, folgen sechs Nullbytes zur Synchronisation und ein Byte das angibt, ob die folgenden Daten als vorhanden betrachtet werden sollen (FB) oder nicht (F8). Ein F8 an dieser Stelle bedeutet, daß das folgende Datenfeld gelöscht wurde. Beendet wird der Sektor durch 2 Prüfbytes und eine Lücke (data gap) von 27 Bytes.
Die Länge der abschließenden Lücke (Postambel oder trailing gap) kann etwas schwanken, da ein absoluter Gleichlauf der Diskette nicht erreichbar ist.
Dem Ausgleich geringfügiger Drehzahlschwankungen dienen insbesondere auch die beiden Lücken vor und hinter dem Datenfeld (14.-24. und 162.-188. Byte).

Datenaufzeichnung:

Zur Datenaufzeichnung werden heute zwei verschiedene Verfahren benutzt, das ältere mit der sog. einfachen Dichte (single-density) und das weiterentwickelte mit doppelter Dichte (double-density).

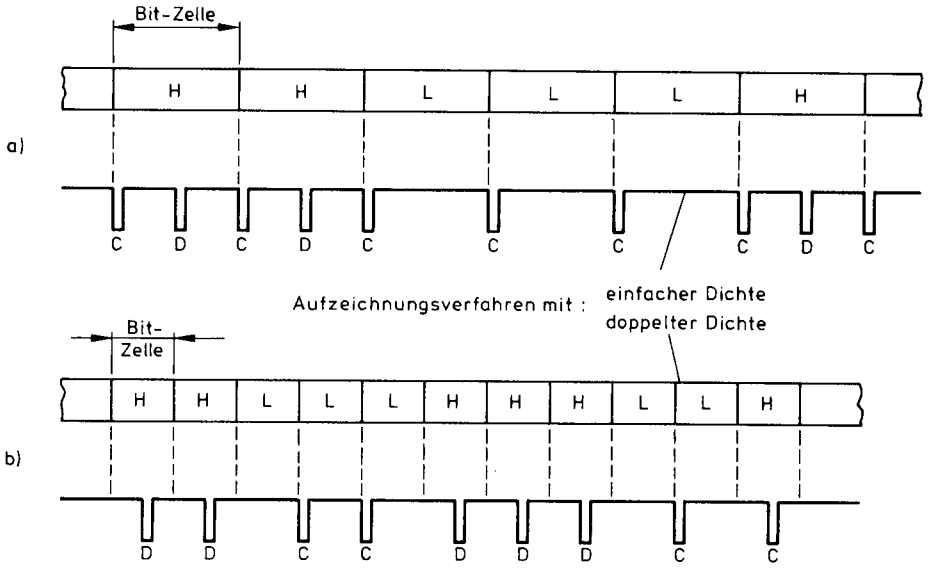


Abb. 10.35: Disketten Aufzeichnungsverfahren mit einfacher und doppelter Dichte. C = Taktimpuls (clock), D = Datenimpuls.

Die Bezeichnung einfache bzw. doppelte Dichte bedeutet, daß im zweiten Verfahren, der für die Aufzeichnung eines Bits erforderliche Platzbedarf, halb so groß ist (Abb.10.35). Die Speicherkapazität einer Diskette (Minidiskette) wird dadurch verdoppelt. Gleichlaufschwankungen machen sich jedoch stärker bemerkbar, woraus eine etwas geringere Datensicherheit resultiert.
Die Aufzeichnung in einfacher Dichte, geschieht in der Frequenzmodulationstechnik (FM) (Abb.10.35a). Jede Bit-Zelle beginnt mit einem Taktimpuls (C) und enthält einen weiteren Impuls (D) oder nicht, je nachdem ob ein H oder L codiert werden soll.
In Abb.10.35b ist das sog. modifizierte Frequenzmodulationsverfahren (MFM) dargestellt. Hierbei wird maximal ein Impuls pro Bit-Zelle erzeugt, woraus die höhere Aufzeichnungsdichte resultiert. Wenn in einer Zelle, zur Darstellung eines H, ein Datenimpuls (D) gegeben wird, verzichtet man auf den Taktimpuls (C). Nur wenn mehrere L hintereinander darzustellen sind, wird zur Vermeidung einer größeren Lücke, vom 2. L an ein Taktimpuls geschrieben.

Anschluß an einen Mikrocomputer:

Zum Anschluß eines Diskettenlaufwerkes an einen Mikrocomputer, werden heute meistens hochintegrierte Steuerbausteine eingesetzt (FDC = floppy disk controller). Der FDC8271 ist ein solch universeller Baustein (Abb.10.36). Er ist kompatibel mit dem Softsektor-Format IBM3740 (Abb.10.34) und nimmt eine interne CRC-Prüfbyte-Erzeugung und Prüfung vor.

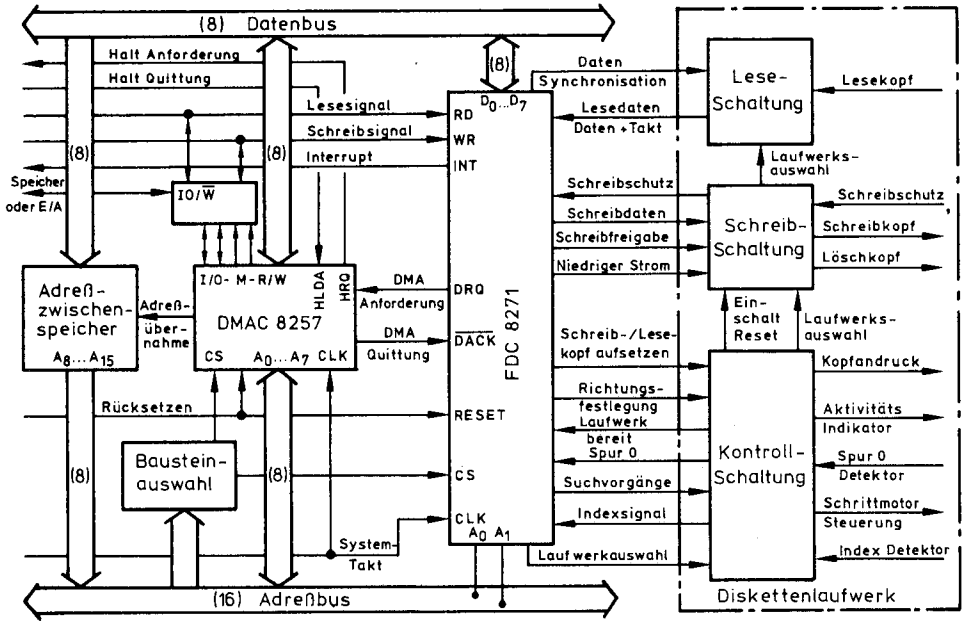


Abb. 10.36: Anschluß eines Diskettenlaufwerkes mit Hilfe eines FDC (floppy disk controller) und eines DMAC (direct memory access controller) an einen MP 8085

Wie aus der Abb.10.36 ersichtlich ist, wird der FDC an die Busse des Mikrocomputers angeschlossen und erzeugt dann alle Signale die zum Betrieb eines Disketten- (Minidisketten-) Laufwerkes erforderlich sind.
Aufgrund der hohen Datenübertragungsgeschwindigkeit ist es sinnvoll, den Datenaustausch im direkten Speicherzugriff (DMA) durchzuführen. Der DMAC8257 erzeugt hierzu die Adressen, wobei das niederwertige Byte direkt und das höherwertige Byte über den Datenbus - unter Zwischenspeicherung - auf den Adreßbus gegeben wird.

Zur Frage, ob ein direkter Datenaustausch erforderlich ist, sei folgendes Beispiel angeführt.

Beispiel: Datenübertragungsrate zu einem FDC der ein Diskettenlaufwerk steuert.

Anzahl der Bytes pro Spur (Abb.10.34): (73+26x188+247) Bytes = 5208 Bytes.

Drehzahl der Diskette: n = 360/min = 6/s

Datenübertragungsrate = 5208 Bytes x 6/s = 31248 Bytes/s.

Zur Übertragung eines Bytes stehen damit 32 Mikrosekunden zur Verfügung. Bei einer Aufzeichnung mit doppelter Dichte, verkürzt sich diese Zeit auf die Hälfte, da pro Spur doppelt soviel Daten untergebracht werden.
Diese hohen Datenraten rechtfertigen den Einsatz eines DMAC.

Zusammenfassende Daten:

Die Bezeichnungen Diskette und Minidiskette unterscheiden im wesentlichen nur zwei verschiedene Größen flexibler Magnetplatten.
Zwei weitere Unterscheidungsmerkmale sind durch die Art der Sektorierung (Hard oder Soft) und die Aufzeichnungsdichte (einfach oder doppelt) gegeben (Abb.10.37).
In den letzten Jahren wurden die Laufwerke - im Hinblick auf eine größere Datendichte - weiterentwickelt und es gibt nun Laufwerke mit denen Minidisketten beidseitig beschrieben werden können und auch solche mit der doppelten Anzahl von Spuren pro Diskettenseite. Beide Techniken zusammen führen zu einer Vervierfachung der Speicherkapazität gegenüber einfachen Minidisketten doppelter Dichte (640KB).

Eine Minidiskette mit 40 Spuren hat 48 Spuren pro Zoll (48 tpi) und eine mit 80 Spuren entsprechend 96 Spuren pro Zoll (96 tpi). Diese beiden Angaben werden heute vielfach zur Klassifizierung von Laufwerken benutzt. (Z.B. 96-tpi-beidseitig = 640KB-Laufwerk).

Eine völlig neue Klasse von Speichern stellen die Mikrofloppys dar. Diese Disketten haben einen Durchmesser von nur 3,5 Zoll und sind stärker gekapselt als die anderen Disketten.

	Minidiskette				Diskette			
Anzahl der Spuren	40				77			
Größe der Magnetscheibe	5,25" (133mm)				8" (203mm)			
Drehzahl	300/min				360/min			
Aufzeichnungsdichte	einfach		doppelt		einfach		doppelt	
Sektorierung	Hard	Soft	Hard	Soft	Hard	Soft	Hard	Soft
Anzahl der Sektoren	18	16	18	16	32	26	32	26
Datenbytes pro Sektor	128	128	256	256	128	128	256	256
Speicherkapazität/KB	90	80	180	160	310	250	620	500
Übertragungsrate·s/kbit	150	130	300	260	307	250	614	500

Abb. 10.37: Vergleich verschiedener Disketten-Standards (Soft: IBM 3740)

10.6 Übungsaufgaben

10.1 Schreiben Sie ein Programm zur Abfrage der Tastatur gemäß Abb.10.2, unter Zuhilfenahme des Flußdiagrammes in Abb.10.3. Der Anschluß soll an Port C aus Aufgabe 9.3 erfolgen.

10.2 Was geschieht, wenn zwei Tasten - bei einer Schaltung gemäß Abb.10.2 - gleichzeitig gedrückt werden?

10.3 Welche Binärzahl erscheint am Ausgang der Tastatur (Abb.10.4), wenn die Tasten CTRL und M gleichzeitig gedrückt werden? Was bedeutet dieser Code?

10.4 Welche Binärzahl muß auf Port A in Abb.10.5 ausgegeben werden, damit auf einer Anzeige der Buchstabe A erscheint?

10.5 Schreiben Sie ein Programm um auf einer zweistelligen Anzeigeeinheit gemäß Abb.10.5, die Zahl 1A ausgeben zu können. Es soll der Portbaustein aus Aufgabe 9.3 benutzt werden.

10.6 Welches Bitmuster steht in der Speicherzelle eines Zeichengenerator ROMs mit der Adresse 432H? Betrachten Sie hierzu die Abb.10.10 und 10.14.

10.7 Schreiben Sie ein Programm um einen Bildschirm gemäß Abb.10.15 zu löschen. Der Bildwiederholspeicher beginnt bei der Adresse 3000H.

10.8 Welche Bandlänge x wird zur Aufzeichnung eines Bits auf eine Tonkassette - im Kansas-City-Format - benötigt und welche Speicherkapazität C, in KB, hat eine 60min-Kassette?

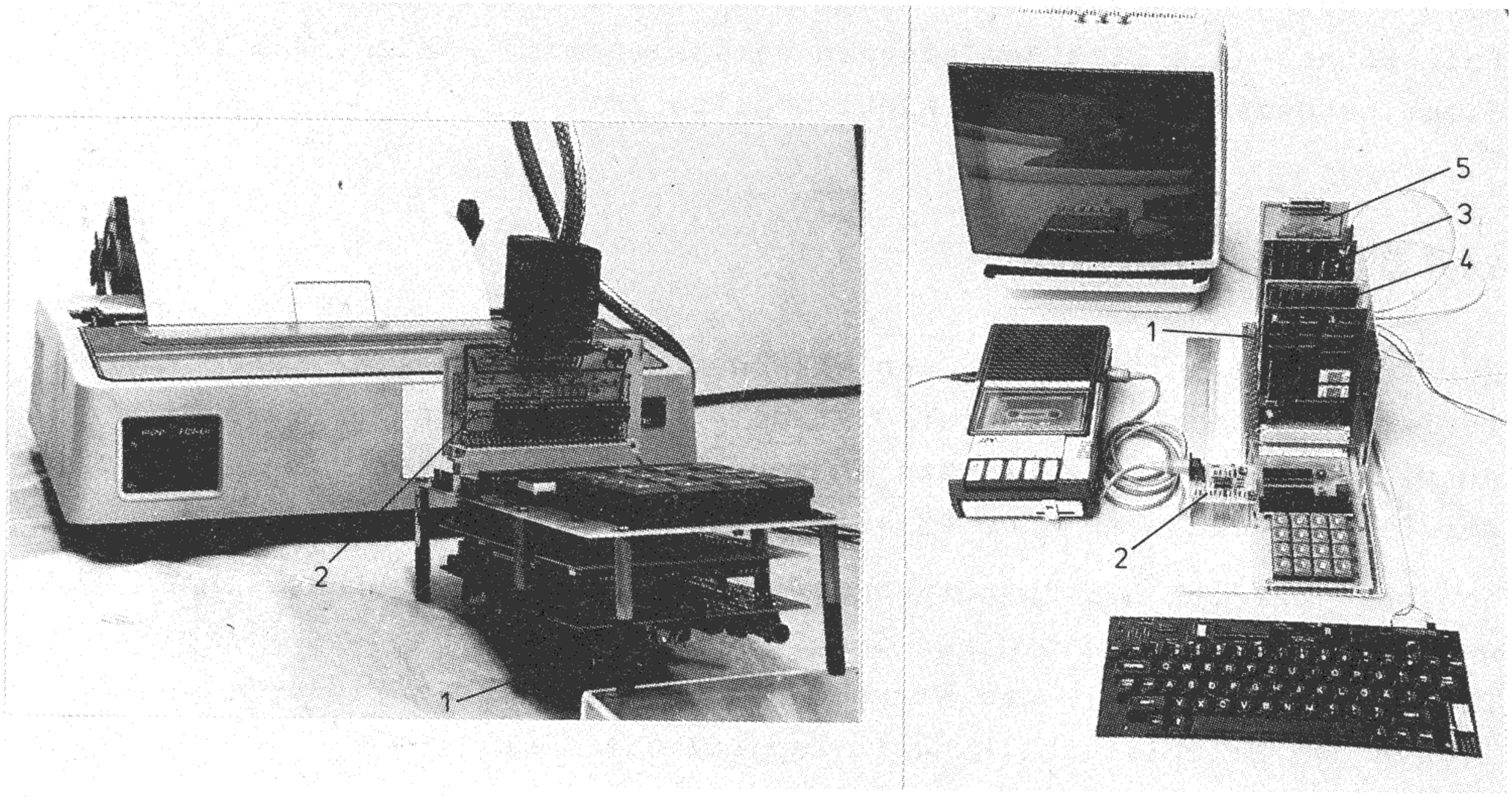
10.9 Wieviel Taktzyklen (f_T = 6,144MHz/2) stehen, zwischen den einzelnen Ausgaben an den Dateneingang eines Kassettenlaufwerkes (Abb.10.29), minimal zur Verfügung, wenn eine programmgesteuerte Richtungstaktschrift, mit 6kbit/s, realisiert werden soll?

10.10 Wie lange dauert es, den Inhalt einer Diskettenspur zu lesen, wenn der Lesekopf bereits auf dieser Spur steht?

C) Anwendungen

11. MP-8085-Mikrocomputersystem

In diesem Kapitel werden die im Bisherigen vermittelten Kenntnisse angewandt zum Aufbau eines Mikrocomputersystemes mit dem Mikroprozessor 8085. Hierbei handelt es sich um den MICO 85, der schon im Kap.4. als Übungssystem Anwendung gefunden hat (s.a. Abb.3.3).



a) b)

Abb.11.1: Mikrocomputersysteme auf der Basis des MICO 85 (Fa.PTFE)
a) MICO 85 P mit Bildschirmsteuereinheit (1) und Druckeranschluß (2).
b) MICO 85 W mit Busplatine (1), Kassetteninterface (2), Bildschirmsteuereinheit (3), Speichererweiterung (4) und EPROM-Programmierplatine (5).S.a. Bezugsquellennachweis.

In Abb.11.1 sind Erweiterungen des MICO 85 dargestellt.

Wie zu sehen ist, gibt es zwei Möglichkeiten die einzelnen Platinen untereinander zu verbinden. Entweder werden sie parallel aufeinander gestapelt - unter Verwendung von Federleisten mit Wickelanschlüssen (64- oder 96-polig, nach DIN 41612, Bauform C) - oder durch Benutzung einer Busplatine (mother board) untereinander verbunden (Abb.11.8).

Die Mikrocomputer in Abb.11.1 enthalten zusätzlich eine Bildschirmsteuereinheit, über die Ausgaben auf einen Bildschirm gemacht werden können. Eine Erklärung der Funktionsweise enthält Kap.10.3.

Weiterhin wurde ein MICO 85 um eine Adapterplatine ergänzt (hinten auf der Tastaturplatine), über die ein Zugriff auf die Ein-/Ausgabekanäle der Mikrocomputerplatine möglich ist. Hieran kann dann ein Matrixdrukker (Kap.10.4 u. Kap.11.5 b) angeschlossen werden, um z.B. Programmlisten (Beisp.4.15) oder Protokolle ausdrucken zu können.

Mit einem solchen System ist es möglich, Steuerungs- oder Überwachungsaufgaben zu lösen. Hierbei können alle aktuellen Zustände etc. auf dem Bildschirm dargestellt und alle wichtigen Ereignisse auf dem Drucker festgehalten werden.

Die Abb.11.1b zeigt eine Möglichkeit, das System noch stärker zu erweitern unter Hinzunahme einer sog. Busplatine (Abb.11.8). In diesem Fall können alle Platinen einzeln herausgenommen werden, was bei der Stapelbauweise (Abb.11.1a) nicht möglich ist.

11.1 Der Mikrocomputer

Zur Anwendung in industriellen Mikrocomputersteuerungen wurde vom Verfasser eine Mikrocomputer-Platine - auf der Basis des MP-8085 - entwickelt. Neben dem Einsatz in verschiedenen Geräten wurde mit dieser Platine auch der MICO 85 aufgebaut.

In Abb.11.2a ist das Grundsystem - einschließlich der Bustreiber und der Bausteinauswahllogik für den Teil b der Schaltung - dargestellt. Dieser Schaltungsteil der Mikrocomputerplatine entspricht im wesentlichen der Beispielschaltung in Kapitel 7.3 (Abb.7.16).

Die Bausteinauswahlschaltung ist hier jedoch für 2KB-Speicher ausgelegt, und an den gemultiplexten Adreß-/Datenbus (AD-Bus) werden die Speicher und der Baustein 8155 der MC-Platine angeschlossen.

Da die Speicher und der Multifunktionsbaustein 8155 direkt am AD-Bus angeschlossen sind (Abb.11.2b) - ohne Datenbustreiber -, muß über eine logische Verknüpfung dafür gesorgt werden, daß der Datenbustreiber (4) nur dann ausgewählt wird (Eingang G), wenn auf einen Baustein außerhalb der MC-Platine ein Zugriff erfolgt. Mit anderen Worten: der Bustreiber verbleibt solange im hochohmigen Zustand, wie der Baustein 11 Auswahlsignale (CS) erzeugt.

Der zweite Teil der MC-Platine kann bis zu 4 EPROM-Speicher aufnehmen und verfügt damit über einen 8KB-Speicherraum (0000 bis 1FFF). Anstelle der eingezeichneten 2716-EPROMs können auch pinkompatible ROMs (z.B.2616) oder RAMs (z.B. NMOS-4016, CMOS-6116) benutzt werden.

Sollte diese Speicherkapazität nicht ausreichen, so lassen sich - mit einer leichten Schaltungsänderung - auch die 4KB-EPROMs 2732 einsetzen.

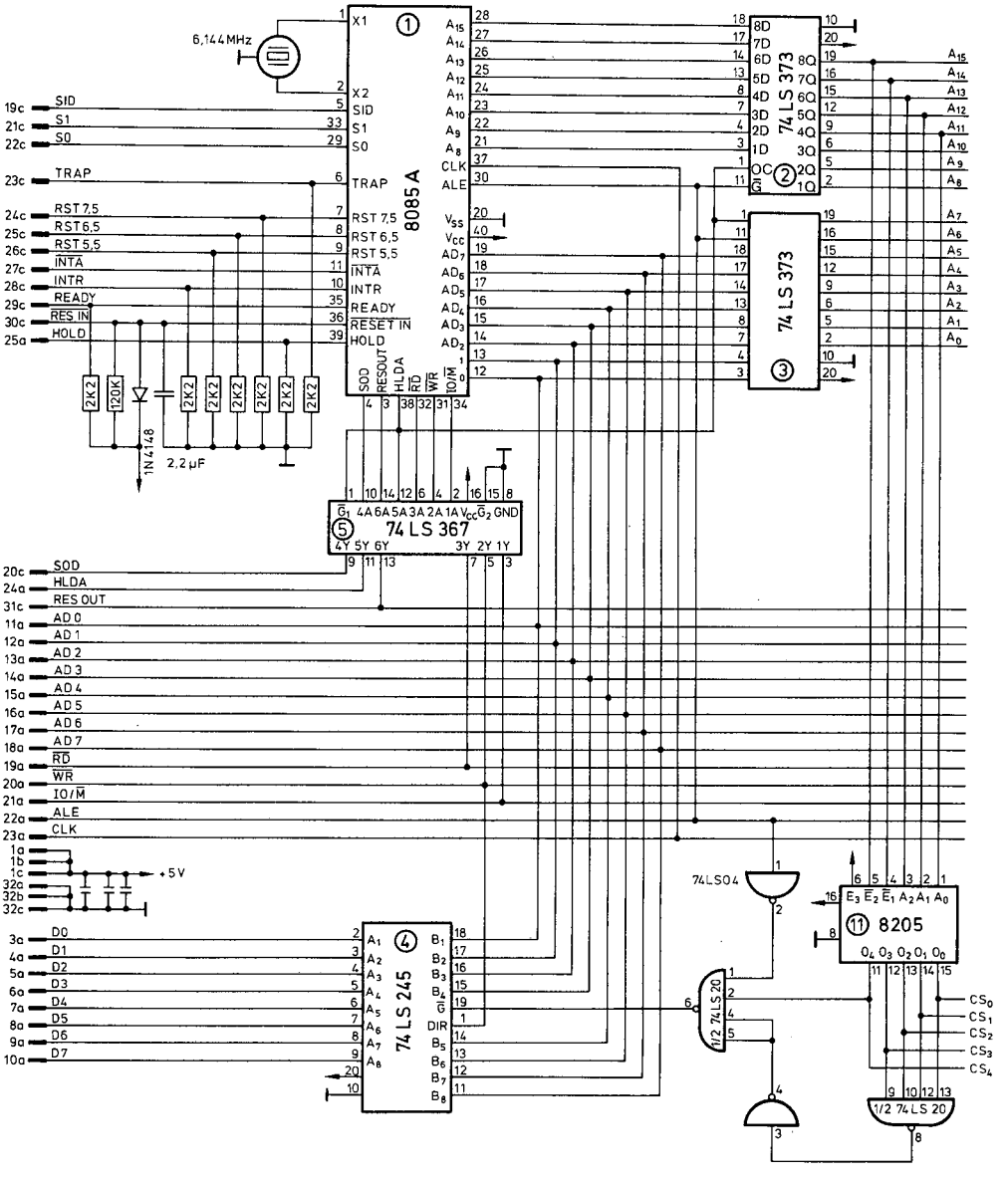


Abb.11.2: MP-8085-Mikrocomputer-Platine
a) Grundsystem mit Bustreibern und Bausteinauswahllogik (s.a. Abb.7.16).

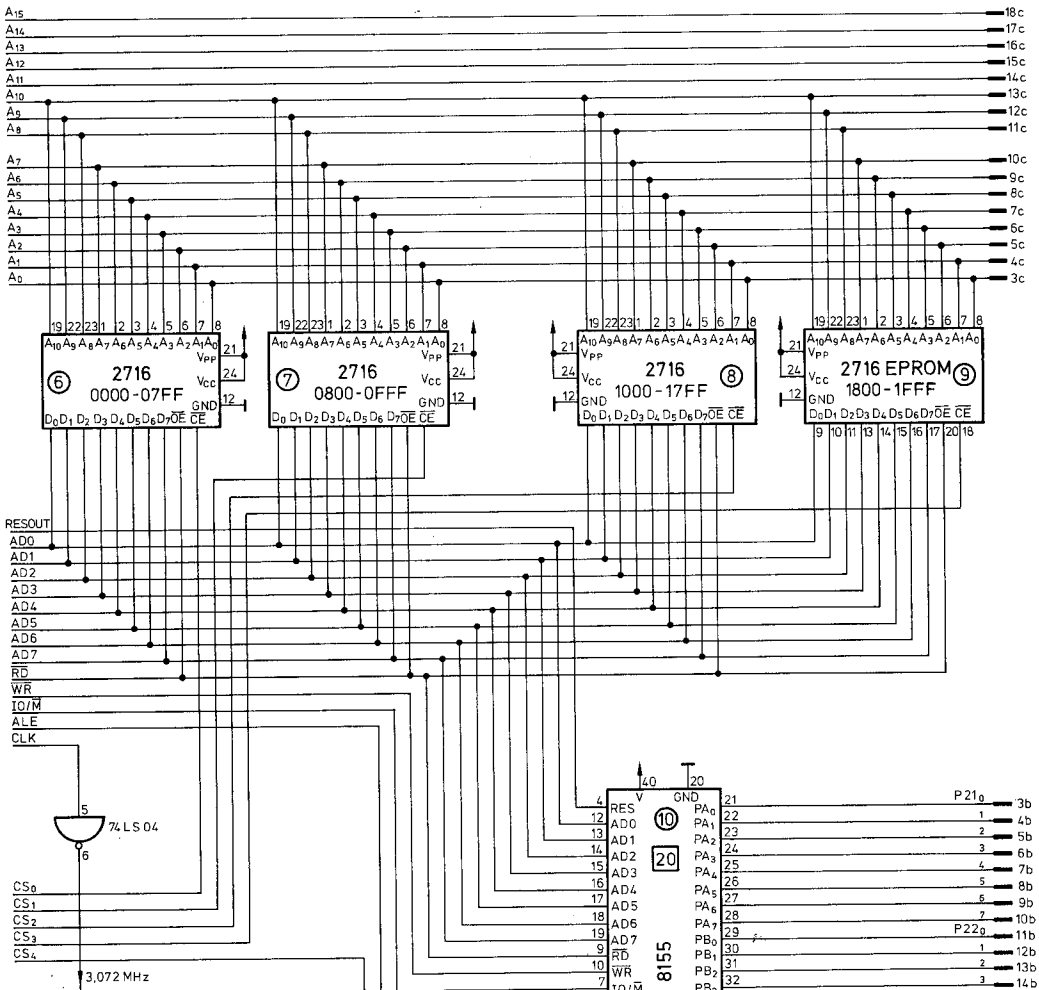


Abb.11.2: b) Anschluß der Speicher und des Multifunktionsbausteines (8155: RAM, Port, Timer) an das Grundsystem.

Die MC-Platine (Abb.11.2b) enthält auch einen Baustein der MP-8085-Familie,welcher direkt an den AD-Bus anzuschließen ist; die Adreßdecodierung erfolgt intern. Dieser Baustein (8155) besteht aus drei Funktionsgruppen, einem statischen RAM (256 x 8bit), zwei 8bit-Ports, einem 6bit Port und einem Zeitgeber (timer, Frequenzteiler). Der programmierbare 14bit-Zeitgeber erhält am Eingang (TIMER IN) den durch die Bausteine 12 und 13 in der Frequenz auf 750Hz reduzierten Takt des Mikroprozessors. Durch die Hinzunahme dieses Bausteines beinhaltet die Platine einen kompletten Mikrocomputer, mit dem sich kleinere Steuerungsschaltungen ohne viel zusätzliche Hardware realisieren lassen.

Der Zeitgeber ist zur Lösung von Echtzeitaufgaben (real time) gut geeignet, wenn der Ausgang TIMER OUT mit einem Interrupt-Eingang des 8085 verbunden wird (s.a.Kap.11.5a).

Adreßbereiche

Da der 1 aus 8 Decoder (11) sowohl Speicherbausteine als auch Ein-/Ausgabe-Ports auswählt, findet in dem Adreßbereich 0000 bis 1FFF bzw. 00 bis 1F keine IO/M-Trennung (in/out oder memory) statt.

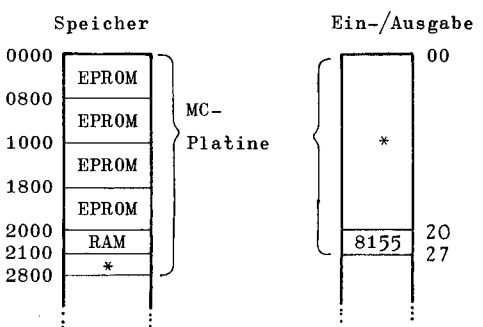


Abb.11.3:

Adreßaufteilung auf der Mikrocomputer-Platine
* nicht benutzbare Bereiche.

Wenn der Speicherbereich von 0 bis 1FFF voll bestückt ist, können keine Ein-/Ausgabe-Bausteine auf die Adressen 0 bis 1F gelegt werden (Abb 11.3). Die Adreßbereiche im Baustein 8155 überschneiden sich, da intern eine Aufteilung - durch das IO/M-Signal - stattfindet. Der 1 aus 8 Decoder teilt den Speicherraum in 2KB-Bereiche auf, wodurch das 1/4 KB-RAM im 8155 eine Mehrfachauswahl erfährt, d.h.,es gibt in dem Adreßbereich von 2000 - 27FF immer 8 Adressen - die sich um den Wert 100H voneinander unterscheiden - die denselben Speicherplatz ansprechen.

Beispiel 11.1: Mehrdeutigkeit der RAM-Adressen im 8155.

Der RAM-Speicherplatz mit der Adresse 203AH wird außerdem noch durch die Adressen 213A, 223A bis 273A angewählt.

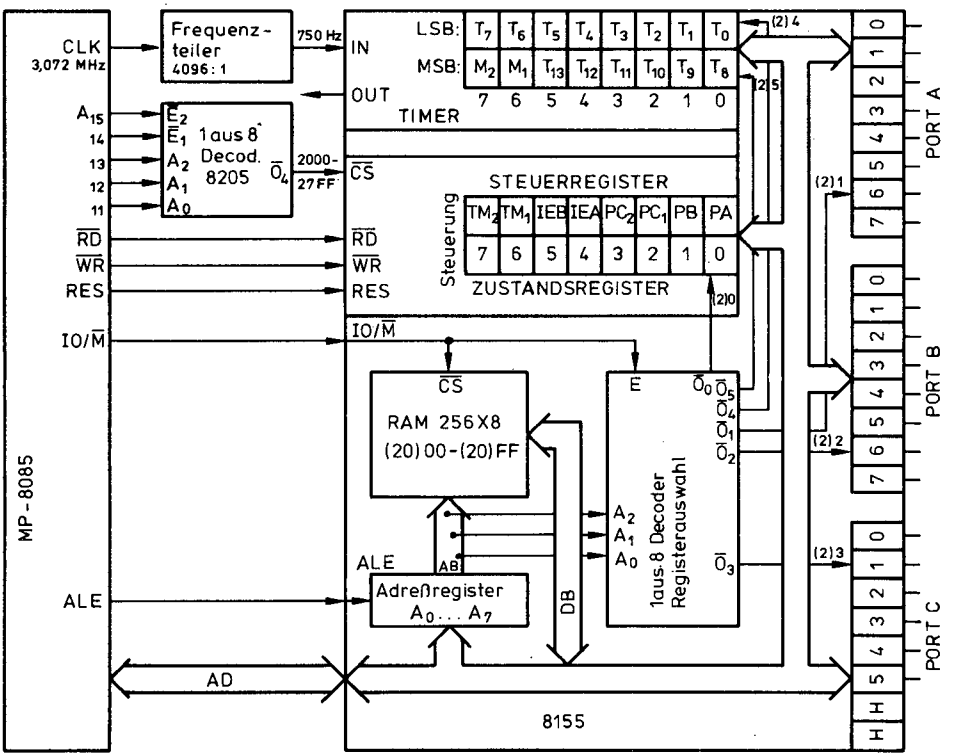


Abb.11.4: Prinzipieller Aufbau des Multifunktionsbausteines 8155 mit Anschluß an den MP-8085 gemäß Abb.11.2b.
LSB = least significant byte = niederwertiges Byte,
MSB = most significant byte = höherwertiges Byte.

Der Speicherbereich ab 2800H und der Ein-/Ausgabebereich ab 28H sind frei für weitere Anwendungen. Der Adreßbereich von 3000 - 3FFF ist z. B. für den Bildwiederhol- und Attributspeicher einer Bildschirmsteuerplatine (VRAM) reserviert und von der Adresse 4000H an können bis zu 3 RAM-Platinen - wie in Abb.8.11 beschrieben - angeordnet werden.

Der Multifunktionsbaustein 8155

Die drei Komponenten dieses Bausteines - RAM, Timer, Port - werden über den gemultiplexten Adreß-/Datenbus (AD) mit dem MP-8085 verbunden. Ein interner Adreß-Zwischenspeicher sorgt für eine Aufteilung der

beiden Busse (Abb.11.4); gesteuert durch das Signal ALE (Adresse einschreiben). Das niederwertige Adreßbyte ($A_{0...7}$) dient zur Auswahl der Speicherplätze des RAMs (IO/M= L) und der 6 Register des Timers und der Ports (IO/M = H).

Zu den internen Adressen des Bausteines muß noch die Bausteinauswahladresse addiert werden, um die für die Programmierung gültigen Adressen zu erhalten.

Jede Adresse, die zwischen 2000 und 27FF liegt, unabhängig vom Zustand des IO/M-Kanals, selektiert den Baustein 8155 - bei einer Verschaltung gemäß Abb.11.4. Auch die Portadresse 21 (Adreßbusausgabe: 2121) führt

Komponente im 8155	interne Adresse	IO/M	Lesen (RD) Schre.(WR)	absolute Adresse
RAM	00 - FF	L	RD o. WR	2000-20FF
Steuerregister	00	H	WR	20H
Zustandsregister	00	P	RD	20H
Port A	01	H	RD o. WR	21H
Port B	02	H	RD o. WR	22H
Port C	03	H	RD o. WR	23H
Zählregister LSB	04	H	WR	24H
Zählregister MSB	05	H	WR	25H

Abb.11.5: Auswahl der einzelnen Komponenten des 8155.

deshalb zu einer Auswahl des 8155 und damit, infolge IO/M=H, zu einem Kontakt zum Port A (s.a. Abb.11.5).

Programmierung des 8155:

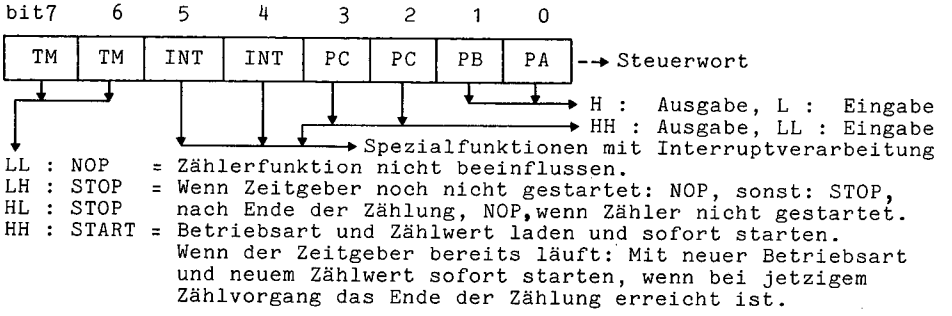


Abb.11.6: Steuerwort für den Mehrfunktionsbaustein 8155

Ähnlich wie beim 8255 (Kap.9.1) ist auch dieser Baustein durch das Einschreiben eines Steuerwortes ins Steuerregister programmierbar.

Die Funktionen der einzelnen Bits des Steuerwortes zeigt Abb.11.6.

Beispiel 11.2: Programmierung des 8155

a) Port A und B: Ausgabe
Port C: Eingabe
Timer : unbeeinflußt

L L L L L L H H --→ 03H

```
MVI  A,3
OUT  20
```

b) Port A,B und C wie bei a)

Timer : starten

H H L L L L H H --→ C3

```
MVI  A,C3
OUT  20
```

Bei einem Lesezugriff auf die interne Adresse 0 (IN 20) wird der Inhalt des Zustandsregisters in den Akku gebracht. Die einzelnen Bits geben dann im wesentlichen Auskunft über Zustände die im Zusammenhang mit den Spezialfunktionen der Interruptverarbeitung (Abb.11.6, s.a. Datenbücher) stehen.

Timer (Zeitgeber):

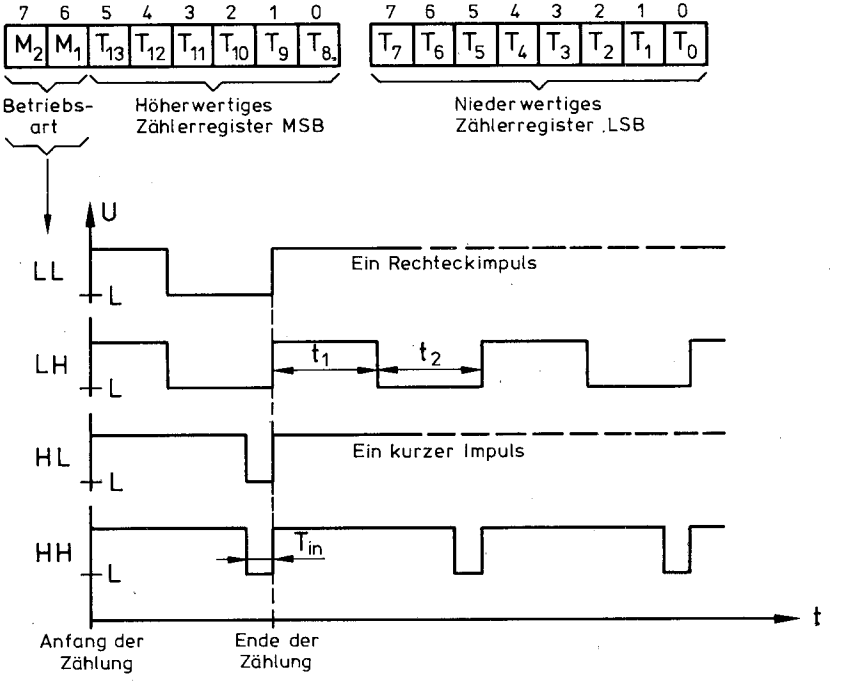


Abb.11.7: Timer-Format und Betriebsarten. M: Modus; T: Timerbit; MSB:, LSB: siehe Abb.11.4.
T_{in} = Periodendauer der Eingangssignale (TIMER-IN),
$T_{out} = t_1+t_2$ = Periodendauer der Ausgangssignale.

Solange der Timer im 8155 nicht programmiert wurde, hat der Ausgang TIMER-OUT den Zustand H. Die Form und Dauer der Ausgangsimpulse in Abhängigkeit von der Eingangsfrequenz (TIMER-IN, z.B.750Hz, gemäß Abb. 11.4) ist durch die beiden Zählerregister LSB und MSB bestimmbar.

Der Timer besteht aus einem 14bit-Abwärtszähler ($T_{0...14}$), d.h., jeder Eingangsimpuls (TIMER-IN) vermindert den Zählwert um 1 . Hierdurch ist eine maximale Reduktion der Eingangsfrequenz um den Faktor 2^{14}=16384 möglich. Der kleinste mögliche Zählwert beträgt 2.

Beispiel 11.3: TIMER-OUT-Bereich bei 750Hz Eingangsfrequenz.

f = 375 bis 0,0457..Hz, T = 0,00266 bis 21,84..s
Sekundentakt: Zählwert = 750_D = $2EE_H$ = HL HHHL HHHL

Die Form und Anzahl der Ausgangsimpulse, wird durch die beiden Bits M_2 und M_1 im MSB-Register festgelegt (Abb.11.7).

Ist der Zählwert ungerade, so ist t_1 um T_{in} größer als t_2.

Beispiel 11.4: Programmierung kurzer Sekundentakte, f_{in} = 750Hz.

Timer Start, Port A und B auf Ausgabe, Port C auf Eingabe (Beisp.11.2b): Steuerwort = C3.

LSB : EE (Beisp.11.3) MSB : $M_2 = M_1 = H$, $T_9 = H$ --→ C2

```
MVI  A,EE
OUT  24   ; LSB laden
MVI  A,C2
OUT  25   ; MSB laden
MVI  A,C3
OUT  20   ; Timer starten
s.a. Kap.11.5a
```

Der Systembus (MICOBUS)

Die MC-Platine hat Europa-Format (160 x 100mm^3) und trägt auf beiden Seiten Leiterbahnen. An einem Ende wird ein 96-poliger (3 x 32) Steckverbinder (DIN 41612, Bauform C) angelötet. Dieser gebräuchliche Steckverbinder besteht aus drei Kontaktreihen die man mit a, b und c bezeichnet, während jede Reihe von 1 bis 32 durchnumeriert wird.

Die Reihen a und c sind mit den Bussen des Mikrocomputers belegt, während die Reihe b im wesentlichen für die Ein-/Ausgabeleitungen reserviert ist (Abb.11.8).

An die Ports der Reihe b können z.B. die Hex-Tastatur und die Anzeigeeinheiten angeschlossen werden.

Zum Anschluß weiterer Einheiten an die Busse und Steuerleitungen der Reihen a und c - wie z.B. einer Speicher- oder Ein-/Ausgabeplatine - gibt es zwei unterschiedliche Möglichkeiten. Entweder werden die Platinen parallel übereinander gestapelt (Abb.11.1a), oder durch eine sog. Busplatine (mother board, Abb.11.1b) miteinander verbunden.

Bei der Stapelbauweise erhalten die Platinen Steckverbinder, sog. Federleisten (DIN 41612, Bauform C), mit Wickelanschlüssen (wire wrap, Abb.11.9). Diese langen, über die Lötseite der Platine hinaus ragenden Wickelstifte, dienen als Stecker für die Federleiste der

nächsten Platine. Auf diese Weise werden alle Signale von einer Platine zur nächsten übertragen (Abb.11.9).

Pin Nr.	Reihe a	Reihe c	Reihe b	Pin Nr.	Reihe a	Reihe c	Reihe b
1	+5V	+5V	+5V	17	AD6	A14	P22(6)
2				18	7	15	7
3	Do	A o	P21(o)	19	-RD	SID	P23(o)
4	1	1	1	20	-WR	SOD	1
5	2	2	2	21	IO/M	S1	2
6	3	3	3	22	ALE	So	3
7	4	4	4	23	CLK	TRAP	4
8	5	5	5	24	HLDA	RST7.5	5
9	6	6	6	25	HOLD	RST6.5	TOUT
10	7	7	7	26		RST5.5	
11	ADo	8	P22(o)	27		-INTA	
12	1	9	1	28		INTR	
13	2	10	2	29		READY	
14	3	11	3	30		-RESIN	
15	4	12	4	31		RESOUT	
16	5	13	5	32	Mass.	Masse	Masse

Abb.11.8: Systembus (MICOBUS) und Ein-/Ausgabebelegung der MC-Platine. Ein Minuszeichen kennzeichnet ein lowaktives Signal.

Sollen größere Systeme aufgebaut werden, bietet sich die Verwendung einer speziellen Busplatine (Mutterplatine) an. In diesem Fall erhalten die Computerplatinen eine abgewinkelte Messerleiste der schon beschriebenen Bauform, während auf die Busplatine Federleisten aufgelötet werden. Die Busplatine verbindet alle Kontakte der a- und c-Reihen untereinander. Wird dann an einer Stelle die Mikrocomputerplatine auf die Busplatine gesteckt, sind auf allen anderen Steckplätzen alle Busse und sonstigen Signale gemäß Abb.11.8 verfügbar.

Die b-Reihen der Steckverbinder sind nicht untereinander verbunden, sondern dienen der Übertragung kartenspezifischer Signale, wie z.B. der Ein-/Ausgabe-Schnittstellen der MC-Platine.

11.2 Die Tastatur- und Anzeigeeinheit

Die Tastatur-(16 Tasten) und die Anzeigeeinheit (8-stellig) sind zusammen auf einer Platine untergebracht (Abb.11.1). Die Verbindung zur Mikrocomputerplatine erfolgt über eine 96polige Federleiste am Kopf der Platine. Hier kann die MC-Platine entweder rechtwinkelig von oben oder parallel von unten aufgesteckt werden.

HEX-Tastatur

Zur Eingabe von Kommandos (Kap.4.2) und zum Füllen der Speicher mit Daten und Befehlen, dient eine Hexadezimaltastatur wie in Abb.10.2 beschrieben.

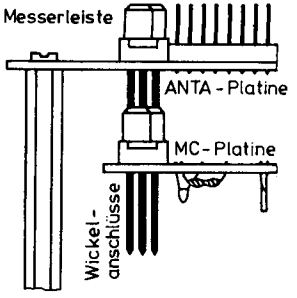


Abb.11.9:
Stapelbauweise für Platinen mittels Federleisten mit Wickelanschlüssen

	$P22_3$	^{2}	^{1}	^{0}
$P23_3$ ←	C	D	E	F
$P23_2$ ←	8	9	A	B
$P23_1$ ←	4	5	6	7
$P23_0$ ←	0	1	2	3

Abb.11.10:
Anschluß der Hexadezimal-Tastatur an die Ports 22 und 23 des 8155 auf der MC-Platine

Die Abb.11.10 zeigt den Anschluß der 4 Tastatur-Spalten an Port $B_{o...3}$ (P22o...3) und der 4 Zeilen an Port $C_{o...3}$ (P23o...3) des Ein-/Ausgabebausteines auf der MC-Platine (Kap.11.1).
Der logische Ablauf beim Einlesen eines Tastenwertes wurde in dem Flußdiagramm in Abb.10.3 beschrieben.
Durch die Verwendung eines anderen Port-Bausteines und anderer Ports ergeben sich jedoch leichte Veränderungen; hierauf wird später noch eingegangen.
Wie aus Abb.11.10 ersichtlich, erfordert der Spaltenanschluß nur 4 Kanäle des Ports B (P22) - es bleiben 4 Ausgabekanäle frei - und beim Zeilenanschluß bleiben 2 der 6 Eingabekanäle des Ports C (P23, Abb. 11.4) ungenutzt.
Diese Kanäle stehen dem Anwender an einem speziellen Stecker der Anzeige-/Tastaturplatine für Ein-/Ausgaben zur Verfügung.

Anzeigeeinheit

Die Anzeigeeinheit besteht aus 8 Siebensegment-LED-Bausteinen, mit denen sich ja das Hexadezimalalphabet (0.....F) darstellen läßt (Abb.10.7a). Zur Entlastung des Mikrocomputers wurde hier jedoch auf eine direkte Ansteuerung der Anzeigeelemente - wie in Abb.10.5 gezeigt - verzichtet. Stattdessen kommen hochintegrierte Spezialbausteine der Firma Intersil zur Anwendung.
Jeder dieser Bausteine (7212MIPL) enthält 4 Speicher, Decoder und Treiber zum Anschluß von 4 Anzeigeelementen. Die vier Datenspeicher, welche jeweils die 4bit einer darzustellenden Tetrade aufnehmen, sind an Port $A_{o...3}$ (P21o..3) der MC-Platine angeschlossen und können über die zwei DSC-Leitungen (digit select code) angewählt werden. Über zwei generelle Auswahlleitungen (CS1,2), angeschlossen an Port $A_{6,7}$, ist festlegbar, welcher Baustein die auf den Datenleitungen anstehende Tetrade ($D_{o...3}$) übernehmen soll.

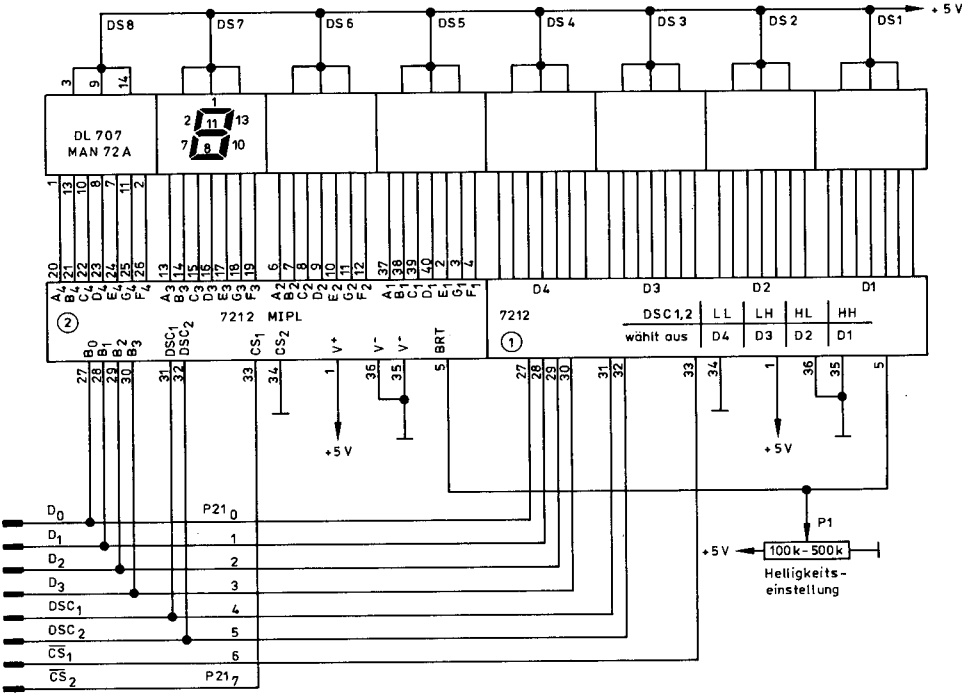
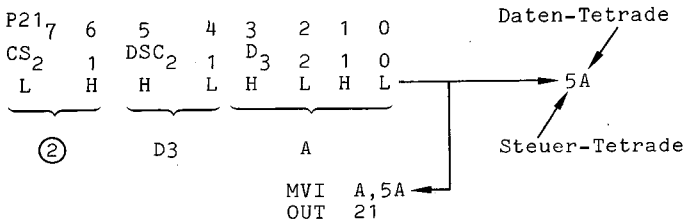


Abb.11.11: 8-stellige, hexadezimale Anzeigeeinheit, angeschlossen an Port A des 8155 auf der MC-Platine. Die Steuertetraden (siehe Beisp.11.5) der Datenstellen 1 bis 8 (DS1...8) lauten: 1.= B, 2.=A, 3.=9, 4.=8, 5.=7, 6.=6, 7.=5, 8.=4.

Beispiel 11.5: Ausgabe der Zahl A auf der Datenstelle 7.



Entsprechend den Darlegungen in Beispl.11.5 lassen sich die Steuertetraden aller Anzeigeeinheiten festlegen (s.Abb.11.11).

Software-Treiber

Der elektronische Aufbau und der Anschluß der Tastatur und der Anzeigeelemente wurden bereits beschrieben. Hier soll nun geklärt werden, wie ein entsprechendes Programm zur Bedienung dieser Einheiten unter Berücksichtigung der gegebenen Hardware auszusehen hat. Da die Einheiten an dem Portbaustein 8155 der MC-Platine (Abb.11.4) angeschlos-

sen sind, muß dieser zunächst in der richtigen Weise programmiert werden:

Port A und B auf Ausgabe (Anzeige-Einheit + Tastatur-Spalten)
Port C auf Eingabe (Tastaturzeilen)

Aus Abb.11.6 folgt dann für das Steuerwort der Wert 03.

Da das Steuerregister die Adresse 20H hat, programmieren die beiden Befehle MVI A,03 und OUT 20 den Portbaustein in der gewünschten Weise. Dies wurde bereits im Beispiel 4.3b mit den ersten beiden Befehlen durchgeführt (Initialisierung im Monitorprogramm). Die Treiber-Routinen für die beiden Einheiten, Tastatur und Anzeige, werden als Unterprogramme ausgeführt. Hierdurch stehen sie auch dem Anwender in Form der Service-Routinen (Kap.4.2) zur Verfügung.

HEX-Tastatur:

Beispiel 11.6: Tastaturabfrage-Programm

```
  TASAB: PUSH B      ;Inhalte der benutzten Register retten.
         PUSH D
         MVI  B,3    ;Tastenwert rechts unten ins Tastenwertregister.
         IN   22H    ;Spaltenport lesen, die
         ANI  0F0H   ;      höherwertige Tetrade separieren
         MOV  D,A    ;      und für später in D speichern.
         ADI  1      ;Akkubit 0 = H für ganz rechte Spalte.
  SPSCH: PUSH B      ;Tastenwert der untersten Reihe retten.
         OUT  22H    ;Spalte 0, 1, 2 oder 3 auf H setzen.
         ANI  0FH    ;Niederw. Tetrade (Spaltenwert) separieren
         PUSH PSW    ;      und abspeichern (retten).
         IN   23H    ;Zeilenzustand laden.
         MVI  C,4.   ;Schleifenzähler für die Zeilenauswertung laden.
  ZESCH: RRC         ;Akkubit 0, 1, 2 oder 3 ins Carry-Flag schieben.
         JC   TASGE  ;Sprung wenn H gefunden (Tastendruck!).
         INR  B      ;Tastenwertregister-Inhalt um
         INR  B      ;      4 erhöhen, dies entspricht
         INR  B      ;      jeweils dem Wert der darüber
         INR  B      ;      liegenden Taste (z.B. 3, 7, B, F).
         DCR  C      ;Zeilen-Schleifenzähler minus 1.
         JNZ  ZESCH  ;Sprung bis alle 4 Zeilen abgefragt wurden.
         POP  PSW    ;Alten Spaltenzustand zurückladen.
         POP  B      ;Alten Tastenwert, ganz unten, rückladen.
         DCR  B      ;Tastenwert links daneben (3,2,1,0).
         CPI  8      ;Letzte Spalte erreicht?
         JZ   ENDE1  ;Sprung,wenn keine Taste gedrückt wurde.
         ADD  A      ;H-Signal um eine Spalte nach links.
         ADD  D      ;Höherwertige, nicht benutzte Tetrade, addieren
         JMP  SPSCH  ;Nächste Spalte H setzen
  ENDE1: MVI  A,0FFH;Kennung: "Keine Taste gedrückt" laden.
         JMP  ENDE2  ;Programmende
  TASGE: POP  PSW    ;Stapelzeiger um 4 erhöhen
         POP  PSW
         MOV  A,B    ;Tastenwert in den Akku bringen
  ENDE2: POP  D      ;Registerinhalte rückladen
         POP  B
         RET
         POP  B
         RET
```

Gemäß dem Flußdiagramm in Abb.10.3 und unter Berücksichtigung der hier benutzten Ports, kann zur Tastaturabfrage das im Beisp.11.6 aufgezeigte Unterprogramm Anwendung finden. Der Wert einer gedrückten Taste wird im Akku übergeben. Wurde keine Taste gedrückt, hat der Akku den Inhalt FF.

Da von den Ports B und C (P22 und P23) jeweils nur die niederwertige Tetrade benutzt wird und die höherwertige Tetrade des Ausgabeports (Port B,P22) durch den Anwender belegt werden kann, darf das Tastaturabfrageprogramm die Zustände auf $P22_{4...7}$ nicht verändern.

Ein solches Problem ist lösbar, indem, wie in Beisp.11.6, 4. bis 6. Befehl gezeigt wird, zunächst der Ausgabeport gelesen und dann die höherwertige Tetrade gesichert wird. Wenn dann später Ausgaben auf diesem Port zu machen sind, wird vorher der Wert der höherwertigen Tetrade des Ports zum Akkuinhalt hinzu gefügt und erst dann eine Ausgabe vorgenommen (siehe Beisp.11.6).

Bei der Benutzung des Unterprogrammes aus Beisp.11.6 ist zu beachten, daß jeder mechanische Taster zum Prellen neigt (Kap.10.1). Dies kann dazu führen, daß bei einem wiederholten Aufruf des Programmes - bei einer einmaligen Tastenbetätigung - der Tastenwert viele Male eingelesen wird. Eine Gegenmaßnahme zeigt Beisp.11.7.

Sehr häufig wird eine Service-Routine benötigt, bei der die Tastatur solange abgefragt wird, bis irgendeine Taste gedrückt wurde - z.B. Kommandoeingabe - (s.a.Beisp.4.5, 4.6, 4.9). Ein solches Zusatzprogramm zeigt Beisp.11.7.

Beisp.11.7: Warten auf Tastendruck

```
TASDR: PUSH B
NOTAS: CALL TASAB ;Tastaturabfrage, Beisp.11.6
       CPI  OFFH  ;Wurde keine Taste gedrückt,
       JZ   NOTAS ;       dann erneute Abfrage.
       MOV  B,A   ;Tastenwert sichern
       MVI  A,64H ;Zeitparameter für 100ms.
       CALL ZA1MS ;Prellphase abwarten (Warteschleife, Beisp.4.3).
WARTE: CALL TASAB ;Tastenabfrage,bis losgelassen wurde (Beisp.11.6).
       CPI  OFFH  ;Wurde Taste losgelassen?
       JNZ  WARTE ;       Wenn ja, dann weiter.
       MOV  A,B   ;Tastenwert laden
       POP  B
       RET
```

Anzeigeeinheit:

Da die Anzeigentreiber nicht ausgelesen werden können, ist es ohne besondere Vorkehrungen nicht möglich festzustellen, welche Zahl gerade auf einer Anzeigeeinheit steht.

Abhilfe bringt die Verwendung eines zusätzlichen Anzeigespeichers. Dieser Anzeigepuffer belegt z.B. beim MICO 85 die Speicherplätze 2000-2007H, d.h., für jedes Anzeigeelement (4bit) ist ein Speicherplatz reserviert. Jeder dieser Speicherplätze nimmt einen darzustellenden Hexadezimalwert (Tetrade) so auf, wie er von dem Tastenabfrageprogramm geliefert wird, d.h. in der niederwertigen Tetrade (z.B.A entspr. LLLL HLHL). Das Anzeigeprogramm in Beisp.11.9 lädt dann in die höherwertige Tetrade die entsprechenden Steuerbits - zur Anwahl eines Anzeigeelementes - und gibt das so vervollständigte Byte auf Port 21 aus (Abb.11.11).

Beispiel 11.8: Darstellung der Zahl 87654321 auf den 8 Anzeigeelementen

Anzeigeelement:	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1
Darzustellender Wert:	8	7	6	5	4	3	2	1
Steuer-+Datentetrade:	48H	57H	66H	75H	84H	93H	A2H	B1H
Speicheradr.: 2000 +	7	6	5	4	3	2	1	0

Soll z.B. die Zahl B auf der Anzeigeeinheit Nr.2 dargestellt werden, so wird diese Zahl zunächst in den Speicherplatz mit der Adresse 2001H gebracht (z.B. LXI H,2001 und MVI M,0B) und dann das Anzeigeprogramm aus Beisp.11.9 aufgerufen (CALL ANZEI).

Der Inhalt einer Anzeigeeinheit ist lesbar, indem die niederwertige Tetrade, auf dem, dieser Einheit zugeordneten Speicherplatz, gelesen wird (z.B. LDA 2001 und ANI 0F: im Akku steht die angezeigte Zahl).

Beispiel 11.9: Anzeigepuffer setzen und zur Anzeige bringen.

```
ANZEI: PUSH B
       PUSH H
       PUSH PSW
       LXI  H,2000H ;Anfangsadresse des Anzeigepuffers
       MVI  B,0B0H  ;Steuertetrade für die 1. Anzeige laden (HLHH).
ANZB2: MOV  A,M     ;Zeichen aus dem Puffer holen
       ANI  0FH     ;Höherwertige Tetrade Null setzen
       ADD  B       ;Steuertetrade hinzuaddieren
       OUT  21H     ;Zeichen zur Anzeige bringen
       ORI  0C0H    ;Bausteinauswahlsignal zurücknehmen.
       OUT  21H     ;CS1=CS2=H
       MOV  A,B     ;     Steuerwort für die
       SUI  10H     ;     nächste Anzeige,
       MOV  B,A     ;     links daneben, erzeugen.
       INX  H       ;Adresse des nächsten Zeichens im Puffer
       MOV  A,L
       CPI  8       ;Pufferende (=Anzeigenende) erreicht?
       JNZ  ANZB2   ;Sprung, wenn kein Pufferende
       POP  PSW
       POP  H
       POP  B
       RET
```

Ein entsprechendes Anzeigeprogramm, welches die Steuertetraden erzeugt, diese in den Anzeigespeicher bringt und alle Anzeigewerte an die Anzeigesteuerbausteine sendet, enthält Beisp.11.9.

11.3 Der Massenspeicher

Massenspeicher dienen in der Mikrocomputertechnik zum dauerhaften Aufbewahren großer Datenmengen. Sie zählen zu den externen Speichern, die als Peripherie-Geräte an einen Computer angeschlossen werden (Kap.10.5). Als Datenträger werden die weit verbreiteten Magnetplatten, Magnetscheiben (Disketten) und Magnetbänder eingesetzt. Neben speziellen, für die Datentechnik entwickelten Magnetbandkassetten sind auch die aus der Unterhaltungselektronik bekannten Audio-Kassetten (Tonkassetten) verwendbar und dies zu besonders günstigen Preisen.

Die im folgenden beschriebene Schaltung und das zugehörige Programm übertragen beliebige Daten (Programme) aus dem internen Speicher eines Mikroomputers auf eine handelsübliche Audio-Kassette als Datenspeicher und umgekehrt.

Speicherverfahren

Zur Speicherung der Daten auf einer Audio-Kassette wurden der Kansas-City-Standard und das Split-Phase-Verfahren eingesetzt. Hierbei werden zur Unterscheidung der logischen Signale H oder L zwei Töne unterschiedlicher Frequenz (2400 bzw. 1200Hz) benutzt.

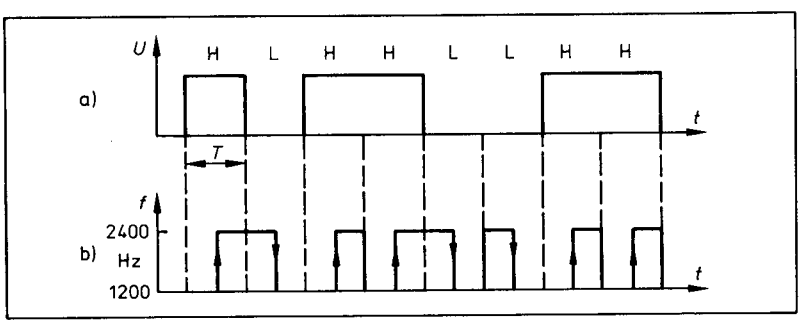


Abb.11.12: Datenaufzeichnung im Kansas-City-Split-Phase-Verfahren. a) Datensignal, b) entsprechendes Split-Phase-Signal.

Das Split-Phase-Verfahren (Abb.11.12) legt fest, daß pro Bitzelle (Länge eines Bits auf dem Band) ein Wechsel des Tones zu erfolgen hat.

Soll ein logisches H geschrieben werden, so wird die erste Hälfte der Bitzelle mit 2 Schwingungen von 1200Hz und der Rest mit 4 Schwingungen von 2400Hz beschrieben, d.h. in der Mitte der Zelle erfolgt ein Wechsel von 1200 nach 2400Hz.

Zum Aufzeichnen eines L wird die umgekehrte Reihenfolge angewandt. In Abb.11.12 ist dieses Verfahren anhand einiger Beispiele dargestellt. Die Datenübertragungsrate ergibt sich hierbei zu 300 Baud (300 bit/s).

Wie Abb.11.13 zeigt, werden die zum Aufzeichnen benutzten Töne durch wechselnde logische Pegel am seriellen Ausgang des Mikroprozessors 8085 erzeugt. In gleicher Weise können sie jedoch auch über den Kanal eines Ausgabeports (z.B. 8255) erzeugt werden. Diese Rechteckschwingungen mit den entsprechenden Frequenzen lassen sich leicht innerhalb eines Schreibprogrammes herstellen. Zur Anpassung der Pegel an den Kassettenrekorder ist dann nur noch das Poti P2 in Abb.11.13 erforderlich.

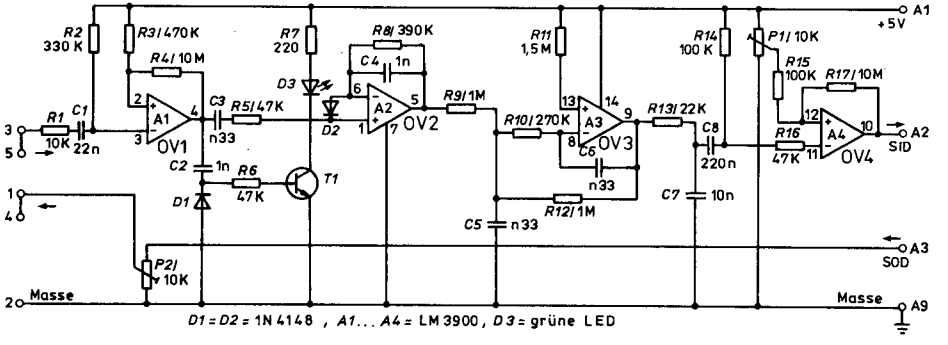


Abb.11.13: Schreib- und Leseschaltung für einen Audiokassettenrekorder im Kansas-City-Standard; über die seriellen Ein- und Ausgänge SID und SOD des 8085.
3/5: zum Kassettenrekorder-Ausgang, 1/4: zum Kassettenrekorder-Eingang.

Schreibprogramm

Das Abspeichern von Daten aus den internen Speichern des MICO 85 auf die Kassette übernimmt ein entsprechendes Programm, das ebenso wie das Einleseprogramm im Monitor (ROM) des Systems eingegliedert wurde.

Dieses Programm sorgt dafür, daß am SOD-Ausgang, an dem der Eingang des Kassettenrekorders angeschlossen ist, die Daten entsprechend dem Split-Phase-Verfahren seriell anstehen.

Entsprechend der Konventionen des MICO 85 wird das Schreibprogramm durch Auswahl der zuständigen Kommandokennzahl (41) aktiviert.

Beisp.11.10: Datenaufzeichnungsformat

Synchronisation SYN	Daten-Kennung niederw.	Daten-Kennung höherw.	Daten	Endkennung EDT
48 x 16H	1 Byte	1 Byte	n Bytes	10 x 04H

Zunächst werden die folgenden Angaben eingegeben:

1. eine aus 4 Hexadezimalzeichen bestehende Kennung für diesen Datenblock (Filekennung, Dateikennung),

2. Anfangsadresse des zu übertragenden Speicherbereiches,
3. Endadresse des zu übertragenden Speicherbereiches.

Nach der letzten Eingabe startet die Übertragung zum Kassettenrekorder. Hierbei wird das in Beisp.11.10 dargestellte Format benutzt.

Leseschaltung

Zum Einlesen der gespeicherten Daten wurde der serielle Eingang des 8085 benutzt (SID), Abb.11.13.
Da bedingt durch den Frequenzgang des Rekorders die Flanken der aufgezeichneten Rechtecksignale abflachen, müssen diese Signale wieder zu Rechtecken geformt werden. Dies geschieht durch die als Komparator arbeitende 1. Stufe (OV1). Im Ruhezustand fließt ein größerer Strom in den invertierenden Eingang als in den nicht invertierenden Eingang: Der Ausgang liegt auf Massepotential.
Bei einem ausreichend großen Eingangssignal vom Rekorder fließt ein größerer Strom in den nicht invertierenden Eingang: Der Ausgang liegt dann auf H-Potential.
OV2 bildet aus dem Komparatorsignal eine von der Signalfrequenz abhängige Spannung. Eine Flanke des Komparatorsignals erzeugt einen kurzen Stromimpuls am nicht invertierenden Eingang von OV2. Um einen Gleichgewichtszustand herzustellen, muß ein ebenso großer Stromimpuls in den invertierenden Eingang von OV2 fließen. Diese Stromimpulse laden C4 auf, hierdurch steigt die Ausgangsspannung von OV2 in gleichem Maße. Über R8 entlädt sich der Kondensator und somit sinkt auch die Ausgangsspannung von OV2 wieder. Je mehr Impulse pro Zeiteinheit eintreffen, desto höher wird die Ausgangsspannung sein; das Signal aus OV1 mit der Frequenz von 1200Hz erzeugt einen niedrigeren Spannungspegel als das Signal mit 2400Hz.
OV3 ist als aktives Tiefpaßfilter geschaltet. Die Eckfrequenz ist gleich der maximalen Signalfrequenz von 300Hz (300Hz entsprechen 300 Baud nach dem Split-Phase-Verfahren). Das Filter soll höherfrequente Störsignale abblocken.
Der Komparator OV4 sorgt für eine ausreichende Amplitude und Flankensteilheit des Signals.
Die Transistorstufe hinter OV1 zeigt mit Hilfe der LED an, ob die vom Rekorder gelieferte Signalspannung zur Decodierung ausreichend ist.

Leseprogramm

Da es sich um eine serielle Datenübertragung handelt, muß der Einlesevorgang synchronisiert werden, d.h., es muß dafür gesorgt werden, daß vom ersten Datenbyte auch das erste Bit eingelesen wird. Um dies sicherzustellen werdem beim Beschreiben der Kassetten zunächst Synchronisationszeichen ausgegeben.

Beim Einlesen muß mindestens ein Synchronisationszeichen erkannt werden, bevor die eigentliche Information eingelesen wird.

Damit Gleichlaufschwankungen des Rekorders keine Einlesefehler verursachen, muß das serielle Signal nachsynchronisiert werden. Diese Nachsynchronisation wird mit Hilfe des Split-Phase-Verfahrens realisiert. Hierbei wird ja das logische H durch einen 1200/2400-Hz-Wechsel und das logische L durch einen 2400/1200-Hz-Wechsel dargestellt, Abb.11.12.

Beim Einlesen wird durch die Schaltung nach Abb.11.13 aus einem 1200 Hz-Ton ein L und aus einem 2400-Hz-Ton ein H erzeugt. Das Programm erwartet deshalb, daß innerhalb der Zeit T (Abb.11.12) ein L/H oder ein H/L-Wechsel eintritt. Hierdurch wird für eine ständige Nachsynchronisation gesorgt, die auch größere Gleichlaufschwankungen, wie sie bei einfachen Rekordern auftreten können, ausgleicht.

Das Leseprogramm wartet zunächst auf die Synchronisationszeichen. Dann wird der eingelesene Programmname mit dem geforderten Programmnamen verglichen. Fällt dieser Vergleich negativ aus, wird auf ein neues Synchronisationszeichen gewartet. Wenn eingelesener und geforderter Programmname übereinstimmen, werden die Daten eingelesen, decodiert und beginnend bei der vorgegebenen Speicheranfangsadresse im RAM-Bereich des Systems abgespeichert. Ist das Programmende erreicht, gibt der MICO 85 eine Fertigmeldung auf seiner Anzeige aus. Wenn das zu speichernde Programm länger ist als der durch die Endadresse festgelegte Speicherbereich, bricht der MICO 85 den Einlesevorgang mit Erreichen der Endadresse ab und es erscheint eine Abbruchmeldung auf der Anzeige.

11.4 Die EPROM-Programmiereinheit

Wenn ein Mikrocomputersystem als Programm-Entwicklungssystem (Kap.5) Anwendung finden soll, ist es unerläßlich, eine Möglichkeit zum Programmieren von EPROMs zu schaffen. Die bereits in Kapitel 8.4 beschriebenen, elektrisch programmierbaren ROMs (EPROM) - mit der Möglichkeit zur Löschung des gesamten Speichers durch UV-Bestrahlung - werden heute in der Mikrocomputertechnik sehr viel eingesetzt. Deshalb soll hier etwas mehr über das Löschen und Programmieren dieser Bausteine ausgesagt und darüber hinaus ein Gerät beschrieben werden, welches - gesteuert durch einen Mikrocomputer - das Lesen und Programmieren von EPROMs ermöglicht.

Am meisten benutzt werden zur Zeit (1983) die EPROMs 2716 (Intel-Typ) oder 2516 (Texas Instr.-Typ); beide Bausteine sind pin- und funktionskompatibel und haben die Organisation 2048 x 8bit (2KB).

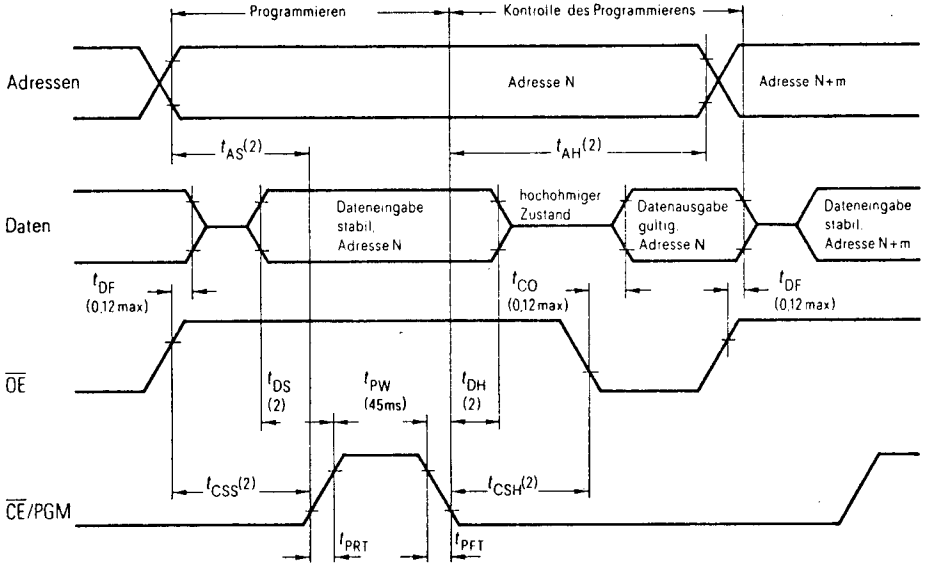
Die fortschreitende technische Entwicklung führt jedoch zu immer größeren (und preiswerteren) Speichern. So werden heute auch schon im erheblichen Umfang die EPROMs 2732 mit 4KB-Speicherkapazität eingesetzt.

Während diese EPROMs noch im 24-poligen Gehäuse unterzubringen sind, haben die neueren EPROMs, wie z.B. 2764 (8KB), 27128 (16KB) und 27256 (32KB), ein 28-poliges Gehäuse.

Im folgenden wird auf eine Platine eingegangen, die, angeschlossen an den MICOBUS (Abb.11.8), das Lesen und Programmieren der EPROMs 2716 und 2732 gestattet.

Löschvorschrift

$V_{PP}=25\ V\pm1\ V;\ V_{CC}=5\ V\pm5\%$



Alle in Klammern angegebenen Zeiten sind Mindestzeiten in µs, wenn nicht anders angegeben.

Abb.11.14: Impulsdiagramm zum Programmieren und Lesen des EPROMs 2716

Wie Abb.8.22 zeigt, verfügen EPROMs über ein Quarzfenster, durch das der Silizium-Chip mit ultraviolettem Licht bestrahlbar ist.

Empfohlen wird hierzu eine UV-Lampe, die Licht der Wellenlänge 253,7nm aussendet. Der Abstand zwischen der Lichtquelle und dem EPROM sollte ca. 2 bis 3cm betragen.

Hieraus ergibt sich dann - je nach Art der Leuchte und des Bausteines - eine Löschzeit von 10 bis 15 Minuten.
Exakt ausgedrückt bedeutet dies, daß auf das Fenster des Bausteines (2716) eine Strahlungsdosis (UV-Intensität x Belichtungszeit) von $15 Ws/cm^2$ - bei 253,7nm - auftreffen muß, um ihn zu löschen.
Das EPROM 2716 wird durch Anlegen eines TTL-Programmierimpulses von 50ms Dauer, an den Anschluß CE/PGM, programmiert, wobei der Eingang OE im H-Zustand ist und die Programmierspannung Vpp = 25V+-1V beträgt (Abb.11.14).

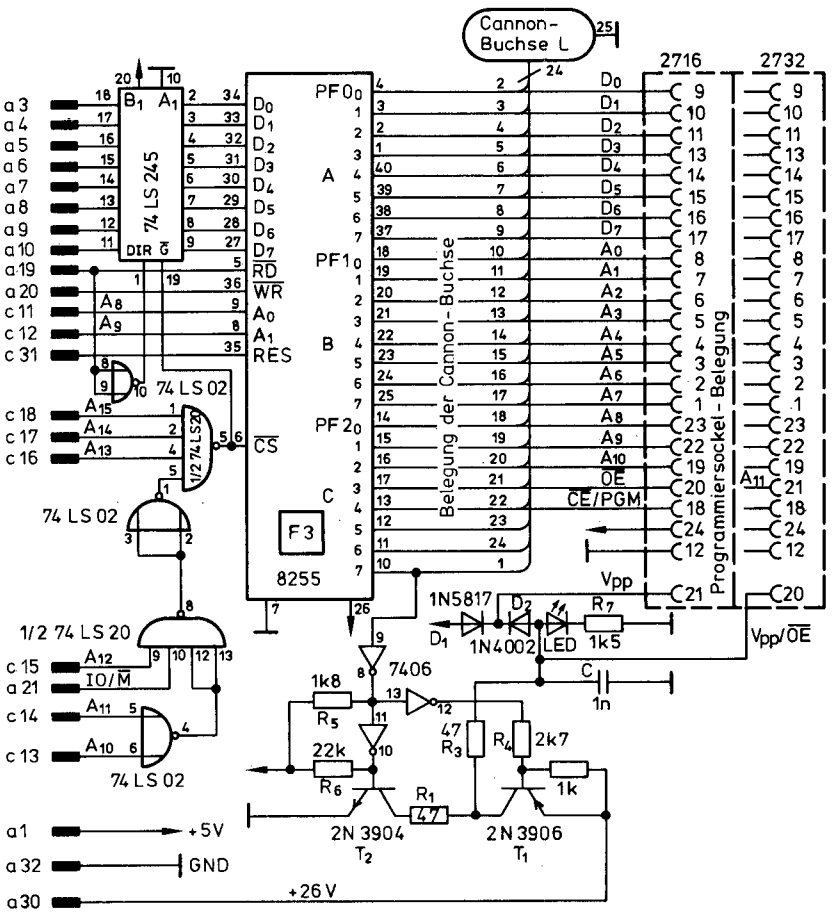


Abb.11.15: Programmier- und Leseschaltung für die EPROMs 2716 und 2732, zum Anschluß an den MICOBUS.
Die Schaltung ist auf einer Eurokarte aufgebaut (Abb.11.1b). Zusätzlich ist ein 25pol. Stecker vorhanden, um den Portbaustein auch für allgemeine Ein-/Ausgaben benutzen zu können.

Es läßt sich jeder Speicherplatz jederzeit programmieren - entweder einzeln, sequentiell oder beliebig. Die Programmierzeit für ein einzelnes Bit beträgt nur 50ms, für alle 16384bit ca. 100s.

Beispielschaltung

Die Abb.11.15 zeigt, wie die EPROMs 2716/32 über einen Portbaustein 8255 (Kap.9.1) als Interface zwischen dem Datenbus und den EPROMs programmiert und ausgelesen werden können.

Steht ein Mikrocomputer zur Verfügung, der schon über genügend programmierbare Ein-/Ausgabekanäle verfügt (22-Kanäle), so kann der 8255, einschließlich des Bustreibers 74LS245 und der Bausteinauswahllogik, entfallen und es braucht nur die Steuerschaltung für die Programmierspannung aufgebaut zu werden.

Die Steuerschaltung für die Programmierspannung (26V) besteht aus den zwei Transistoren, den Invertern und den Dioden. Wird auf dem Portkanal $PF2_5$ ein L ausgegeben, so schaltet der 2N 3906 durch und die Programmierspannung liegt am EPROM an; außerdem leuchtet die LED. Ein H auf $PF2_5$ dagegen schaltet den 2N 3904 durch und über die Schottky-Diode 1N 5817 erhält der Programmiereingang H-Potential.

Die Bausteinauswahllogik, bestehend aus den Bausteinen 74LS02 und 74LS20, weist dem 8255 in Verbindung mit den beiden Adreßleitungen A8 und A9 eindeutig die Ein-/Ausgabeadressen F0 bis F3 zu. Port A hat deshalb die Adresse F0 und das Steuerregister die Adresse F3.

Der Bustreiber (74LS245) ist erforderlich, um bei Leseoperationen einen ausreichenden Strom zu gewährleisten, wenn mehrere Einheiten am Datenbus angeschlossen sind oder Pull-Widerstände die Busleitungen - zwecks größerer Störsicherheit - bedämpfen.

Programmier- und Leseprogramm

Das Flußdiagramm in Abb.11.16 zeigt den Ablauf beim Programmieren und Lesen eines 2716 EPROMs. Nach jedem Programmier- oder Lesevorgang für ein Byte, erfolgt eine Kontrollesung. Bei der Kontrollesung wird der Inhalt des momentan adressierten EPROM-Speicherplatzes gelesen und mit dem Inhalt des RAM-Speicherplatzes verglichen, dessen Inhalt ins EPROM gebracht werden sollte (programmieren) oder der aus dem EPROM geladen wurde (lesen).
Sind beide Speicherplatzinhalte nicht gleich, so liegt ein Fehler vor, der anzuzeigen ist. Ein Fehler beim Programmieren kann z.B. bedeuten, daß das EPROM nicht lange genug gelöscht worden ist. Beim Lesen wird beispielsweise ein Fehler angezeigt, wenn der Speicherbereich in den der Inhalt des EPROMs gebracht werden soll, nicht mit RAMs bestückt ist (z.B. gar keine oder ROMs).

Zur Ausführung einer Programmier- oder Leseoperation sind dem Programm drei Adressen zu übergeben. Dies sind üblicher Weise die Anfangs- und Endadresse des beteiligten RAM-Speichers und die Startadresse im EPROM; wobei mit EPROM-Adresse eine relative Adresse des Bausteines - beginnend mit 0000 und endend mit 07FFH, bei einem 2KB-EPROM - gemeint ist.

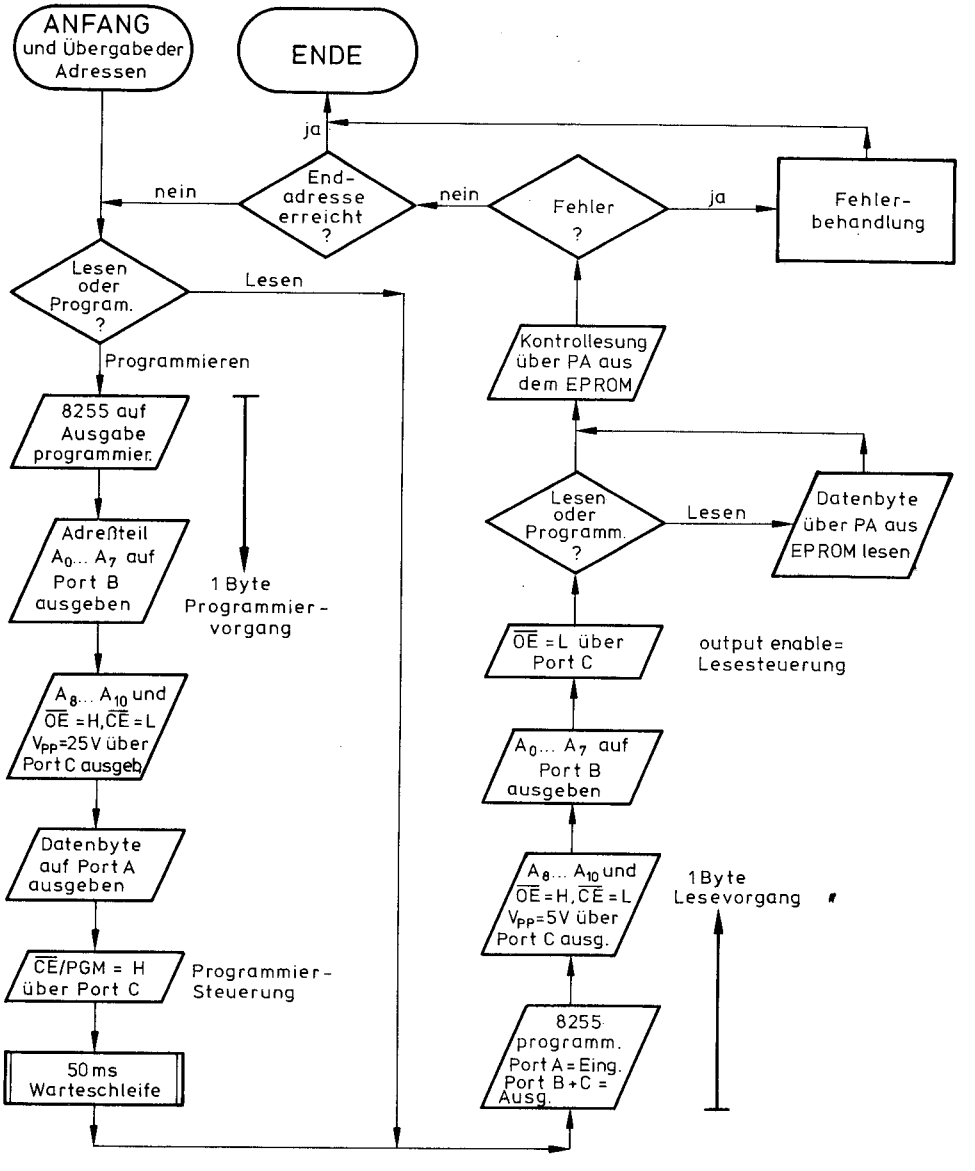


Abb.11.16: Programmier- und Leseprogramm für ein 2716 EPROM

Beisp.11.11 enthält ein Programm - unter Voraussetzung der Schaltung in Abb.11.15 - welches das Flußdiagramm aus Abb.11,16 konkretisiert.

Beispiel 11.11: Programmier- und Leseprogramm für ein EPROM 2716, gemäß Abb.11.15.

Über das Carry-Flag wird die Auswahl zwischen Programmieren (H) oder Lesen (L) durchgeführt.

Die Registerpaare HL und BC müssen die Anfangs- bzw. Endadresse des RAM-Bereiches enthalten und DE die Anfangsadresse des EPROMs.

```
EPROM:  DI
        PUSH   PSW
SP66:   POP    PSW     ;Flags laden
        PUSH   PSW     ;Flags sichern
        JNC    LESEN   ;Carry = L bedeutet Lesen
        MVI    A,80H   ;Steuerwort für den 8255: Port A+B+C=Ausgabe,
        OUT    0F3H    ;       ins Steuerregister schreiben.
        MOV    A,E     ;Adresse
        OUT    0F1H    ;       und
        MOV    A,D     ;       Steuersignale
        ORI    8       ;       ausgeben:
        OUT    0F2H    ;       Vpp=25V.
        MOV    A,M     ;Das zu programmierende
        OUT    0F0H    ;       Byte ausgeben.
        MOV    A,D
        ORI    18H     ;CE=OE='H'
        OUT    0F2H
        MVI    A,50
        CALL   ZA1MS   ;50 ms Warteschleife
LESEN:  MVI    A,90H   ;Steuerwort 8255, Port A=Ein, PB+PC=Aus
        OUT    0F3H
        MOV    A,D     ;Höherwertige Adresse ausgeben
        ORI    28H
        OUT    0F2H    ;OE='H',Vpp=5V,CE='L'
        MOV    A,E     ;Niederwertige Adresse ausgeben
        OUT    0F1H
        MOV    A,D
        ORI    20H     ;OE='L'
        OUT    0F2H
        POP    PSW     ;Flags laden und rücksichern
        PUSH   PSW
        JC     SP65    ;Sprung, wenn programmiert wird
        IN     0F0H    ;Datenbyte vom EPROM einlesen
        MOV    M,A     ;eingelesenes Datenbyte ins RAM schreiben.
SP65:   IN     0F0H
        CMP    M       ;Richtig geschrieben oder gelesen?
        JNZ    FEHLER  ;Sprung, wenn nicht (Fehler)
        PUSH   H
        DB     8;(DSUB) Enadr. erreicht ?
        POP    H
        INX    H
        INX    D
        JNZ    SP66    ;Sprung, wenn Ende nicht erreicht
        POP    PSW
        MVI    A,28H
        OUT    0F2H    ;Vpp=5V
        EI
        RET
ZA1MS   EQU    0BH     ;Service-Routine (Beisp.4.3) Zeitschleife
FEHLER  EQU    XXXX    ;Fehlerbehandlungs-Routine
        END
```

Wenn das Programm als Unterprogramm Anwendung findet, müssen im aufrufenden Programm noch vorher vom Benutzer die Programmieradressen erfragt werden. Weiterhin sind noch eine Warteschleife (Wartezeit = (A)*1ms, z.B Service-Routine ZA1MS) als Unterprogramm und ein Fehlerbehandlungs-Programm als Ergänzung erforderlich.

11.5 Anwendungsbeispiele

Hier werden einige Anwendungen der in den vorangegangenen Kapiteln beschriebenen Komponenten aufgezeigt.

a) Interrupts vom Timer

Benutzung des Timers im 8155 zur Erzeugung von Interrupts.

Die Programmierung des Timers wurde bereits anhand von Beisp.11.4 erläutert. Zur Erzeugung von Interrupts wird der Ausgang des Timers (TIMER-OUT) z.B. mit dem Eingang RST 7.5 (s.a.Abb.7.23) verbunden.

Jede ansteigende Flanke eines Timer-Impulses bewirkt dann einen Interrupt und somit einen UP-Sprung zur Adresse 003C (Abb.7.17). Auf dieser Adresse steht im Monitorprogramm (Beisp.4.3a) ein Sprungbefehl zur Adresse 2026 des Arbeitsspeichers. An dieser Stelle des Arbeitsspeichers sind drei Speicherplätze reserviert, um den HEX-Code eines weiteren Sprungbefehles - zum eigentlichen Interruptbedienprogramm hin - aufzunehmen.

Wenn ein Interrupt erfolgt, wird der gerade laufende Befehl zuende bearbeitet, die Adresse des folgenden Befehles über den Stapelzeiger im Arbeitsspeicher abgelegt und ein Sprung auf die Adresse 003C ausgeführt. Von dort erfolgt ein Sprung in den Arbeitsspeicher (Adr.2026H) und dann ein Sprung auf den Anfang des Interruptbedienprogrammes.

Am Ende dieses Programmes müssen durch den Befehl EI (enable interrupts) die weiteren Interrupts wieder zugelassen werden und über den Befehl RET (return) ein Rücksprung ins unterbrochene Programm erfolgen.

Mit den ersten 6 Befehlen des Programmes in Beisp.11.12 wird der Interrupt-Vektor auf die Adresse 1830H des Bedienprogrammes gelenkt: Interrupt RST 7.5 --→ 003C, (Monitor) --→ 2026 (Arb.Sp.) --→ 1830 (Bedienprogramm).

Die nächsten 6 Befehle programmieren und starten den Timer und die folgenden 3 Befehle aktivieren die Interruptannahme, (Abb.7.21).

Anstelle der zwei letzten Befehle des Hauptprogrammes - dort hält der Mikroprozessor jeweils an - kann auch ein Sprung in ein anderes Programm erfolgen.

Beispiel 11.12: Unterbrechungen im Sekundentakt und Anzeige einer dezimalen Sekundenzählung auf den 4 mittleren Anzeigen des MICO 85 (s.a. Beisp.11.4).

```
1800  212620    LXI  H,2026  ;Interrupt-Zieladresse im Arb.Speicher
1803  36C3      MVI  M,C3    ;Sprungbefehl (C3) zum
1805  23        INX  H       ;       Interrupt-Bedienprogramm
1806  3630      MVI  M,30    ;       (Adr. 1830H) in den
1808  23        INX  H       ;       Arbeitsspeicher
1809  3618      MVI  M,18    ;       bringen.
180B  3EEE      MVI  A,EE    ;Niederw.Timerbyte programmieren.
180D  D324      OUT  24
180F  3EC2      MVI  A,C2    ;Höherwertiges Timerbyte programmieren
1811  D325      OUT  25      ;       und Betriebsart HH wählen.
1813  3EC3      MVI  A,C3    ;Timer starten und Port A u. B auf
1815  D320      OUT  20      ;       Ausgabe, Port C auf Eingabe.
1817  3E18      MVI  A,18    ;Interrupt-Maske
1819  30        SIM          ;       setzen.
181A  FB        EI           ;Interrupts zulassen
181B  210000    LXI  H,00    ;Sekunden-Zählregister auf
181E  225020    SHLD 2050    ;       Null setzen.
1821  76        HLT          ;MP anhalten, bis Interr. erfolgt.
1822  C32118    JMP  1821    ;Rückspr. aus Bedienprogr.u. Sprung auf
                             ;       Halt.
      ;Interrupt-Bedienprogramm
1830  D5        PUSH D       ;Inhalte der benutzten Register retten
1831  E5        PUSH H
1832  F5        PUSH PSW
1833  2A5020    LHLD 2050    ;Sekunden-Zählregister laden
1836  7D        MOV  A,L
1837  3C        INR  A       ;       und niederw. Byte um 1 erhöhen.
1838  27        DAA          ;Dezimalkorrektur
1839  6F        MDV  L,A
183A  3E00      MVI  A,00    ;Evtl. Übertrag zum
183C  8C        ADC  H       ;       höherw. Byte hinzuaddieren.
183D  27        DAA
183E  67        MOV  H,A
183F  225020    SHLD 2050    ;Um 1 erhöhten Sekundenwert wegspeichern
1842  EB        XCHG
1843  CD4B00    CALL 004B    ;Sekundenwert anzeigen (ADRAN).
1846  F1        POP  PSW     ;Registerinhalte rückladen
1847  E1        POP  H
1848  D1        POP  D
1849  FB        EI           ;Interrupts erneut zulassen
184A  C9        RET          ;Zurück ins unterbrochene Progr.
```

Wachhund-Schaltung (watch dog):

Die hier dargestellte Technik kann z.B. dazu benutzt werden, den Mikroprozessor in regelmäßigen Intervallen zu unterbrechen und Testprogramme durchlaufen zu lassen. Auf diese Weise ist es möglich, fehlerhafte Arbeitsweisen oder defekte Baugruppen frühzeitig zu erkennen und entsprechende Gegenmaßnahmen einzuleiten.

b) Anschluß eines Druckers

Drucker werden entweder seriell über eine V24- oder RS232C-Schnittstelle oder parallel an einen Computer angeschlossen (s.a.Kap.10.4, Druckeranschluß).

Als Parallelschnittstelle wird meistens die sog. Centronics-Schnittstelle benutzt.

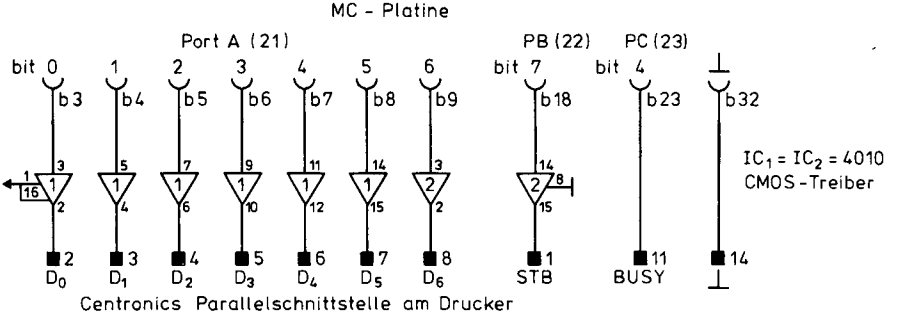


Abb.11.17: Treiberschaltung zum Anschluß eines Druckers an die MC-Platine.

Das Timing dieser Schnittstelle enthält Abb.10.26.
Während die V24- oder RS232C-Schnittstellen die Spannungen + und - 12V benötigen, arbeitet die Centronics-Schnittstelle mit TTL-Potentialen (+5V).

Da an den Ports der MC-Platine (8155) bereits die Anzeigeeinheit und die Tastatur angeschlossen ist, kann der Drucker nicht direkt angeschlossen werden. Die Belastbarkeit der Portausgänge würde damit überschritten. Abhilfe schafft hier die in Abb.11.17 gezeigte Treiberschaltung.

Diese Schaltung besteht nur aus zwei CMOS-Treiberbausteinen. Durch den hohen Eingangswiderstand der CMOS-Bausteine (Kap.6.2) stellen diese keine nennenswerte Lasterhöhung für die Ports dar. Andererseits liefern sie jedoch einen genügend hohen Ausgangsstrom zum Betrieb der Druckerschnittstelle.

Anders als in Abb.10.26 (Timing) gezeigt, wird hier nicht das Signal ACKNLG (Empfangsbestätigung) zur Syschronisation benutzt, sondern das Signal BUSY (nicht empfangsbereit). Eine Übertragung darf nur erfolgen, wenn dieser Druckerausgang den Zustand L hat.

Das Datenbit 7 kann im allgemeinen entfallen, da der zu übertragende ASCII-Code nur 7bit hat.

Das Beisp.11.13 zeigt ein Programm, welches unter Verwendung der Schaltung aus Abb.11.17 einzelne ASCII-Codes an einen Drucker sendet.

Wird ein Matrixdrucker angeschlossen, so ist zu beachten, daß der Ausdruck i.a. erst erfolgt, wenn mit dem Steuerzeichen 0A (LF=neue Zeile) die Übertragung einer Zeile abgeschlossen wird.

Der Kanal, welcher das Datenübergabesignal (STB=strobe) überträgt, wurde an Port B7 angeschlossen (und nicht etwa an Port A7), damit Aus-

gaben auf die Anzeigeeinheit - die ebenfalls Port A benutzt - nicht zu einem Ansprechen des Druckers führen.

Beispiel 11.13: Übertragungsprogramm für ein ASCII-Zeichen an den Drucker.

```
;Übergabe des ASCII-Codes im Register C
 DRUCK: PUSH PSW
 BUSY:  IN   PC     ;Port C (23),bit 4 = BUSY = L?
        RLC
        RLC
        RLC
        RLC
        JC   BUSY   ;Sprung, solange BUSY=H
        MOV  A,C    ;ASCII-Wert in den Akku
        OUT  PA     ;      und an Drucker weitergeben.
        IN   PB     ;Zustand von Port B (22) erfragen.
        ORI  80H    ;Port B, bit 7 = Strobe = H setzen
        OUT  PB
        MVI  A,1
        CALL ZA1MS  ;1ms warten
        IN   PB
        ANI  7FH
        OUT  PB     ;Port B, bit 7 = Strobe = L setzen
        POP  PSW
        RET
 ZA1MS  EQU  0BH    ;Zeitschleife (A)*1ms (Kap.4.2)
    PB  EQU  22H
    PC  EQU  PB+1
        END
```

c) Motorsteuerung

Im folgenden wird dargestellt, wie mit dem MICO 85 die Steuerung eines Gleichstrommotors realisierbar ist.

Abb.11.18 zeigt im oberen Teil einen Gleichstrommotor dessen Anschlüsse über jeweils 2 Schalttransistoren ($T_{1,3}$ bzw.$T_{2,4}$) an die positive Motorbetriebsspannung (U_M) und die Masse (Minuspol) angeschlossen werden kann.

Solange nur die beiden oberen oder nur die beiden unteren Transistoren durchgeschaltet sind, läuft der Motor nicht. Wenn dagegen zwei diagonal angeordnete Transistoren durchschalten, läuft der Motor in der einen oder anderen Drehrichtung (Vertauschung der Polarität am Motor).

Unzulässig ist es, zwei übereinander liegende Transistoren durchzuschalten, da dann ein Kurzschluß entsteht. Damit dies nicht eintreten kann und zur Erzeugung der nötigen Basissteuerströme wurde ein Interface - bestehend aus 4 invertierenden Treibern mit offenem Kollektor (Bild 3.7) - vorgesehen.

Auf diese Weise bleiben nur noch 2 Steuerleitungen übrig. Wenn beide Leitung H- oder L-Potential haben, steht der Motor. Hat dagegen eine Leitung den Zustand H und die andere L, so läuft der Motor entweder im Rechts- oder Linkslauf, je nachdem an welcher Leitung der logische H-Pegel anliegt.

Gesteuert wird der Motor in diesem Beispiel über die beiden Portkanäle B_4 und B_5 der MC-Platine.

Durch Veränderung der logischen Zustände (L oder H) auf den beiden Motorsteuerleitungen können vom Mikrocomputer die unterschiedlichsten Betriebszustände erreicht werden. Zur Kontrolle dieser Betriebszustände sind an andere Kanäle des Mikrocomputers noch entsprechende Sensoren anschließbar (Drehzahlgeber, Lichtschranken, Strommesser etc.).

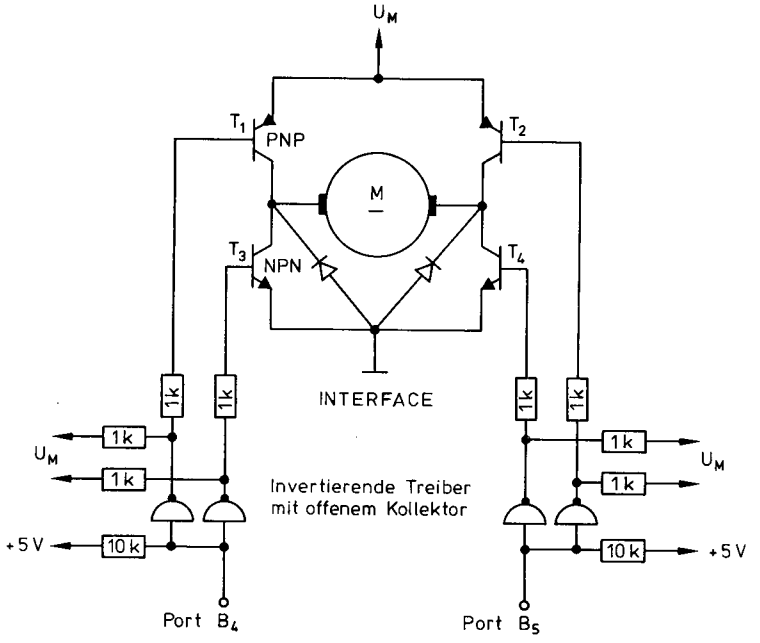


Abb.11.18: Motorsteuerung mit dem MICO 85
T1=T2= BD 680 (80V/4A), T3=T4= BD 679 (80V/4A),
Treiber = 7406 (30V/40mA).

Start des Motors:

Wenn die Schaltung aus Abb.11.18 eingesetzt wird, bewirkt das Programm aus Beisp.11.14, daß der Motor zu laufen beginnt.

Beispiel 11.14: Start des Motors

```
IN  22H    ; Zustände der Kanäle des Ports B in den Akku einlesen
ORI 10H    ; ODER-Verknüpfung des Akkus mit 10H. Ergebnis: Bit4=H
ANI ODFH   ; UND-Verkn. von A mit DFH. Ergebnis: Bit5=L
OUT 22H    ; Ausgabe auf Port B. Erg.:Der Motor läuft links herum
JMP 60H    ; Rücksprung in das Monitor-Programm.
```

Wenn das Programm aus Beisp.11.14 gestartet worden ist, erhält die Motorsteuerung über Port B4 ein H und über B5 ein L. Hierdurch werden die Transistoren T1 und T4 durchgesteuert und der Motor läuft links herum.

Drehrichtungsumkehr:

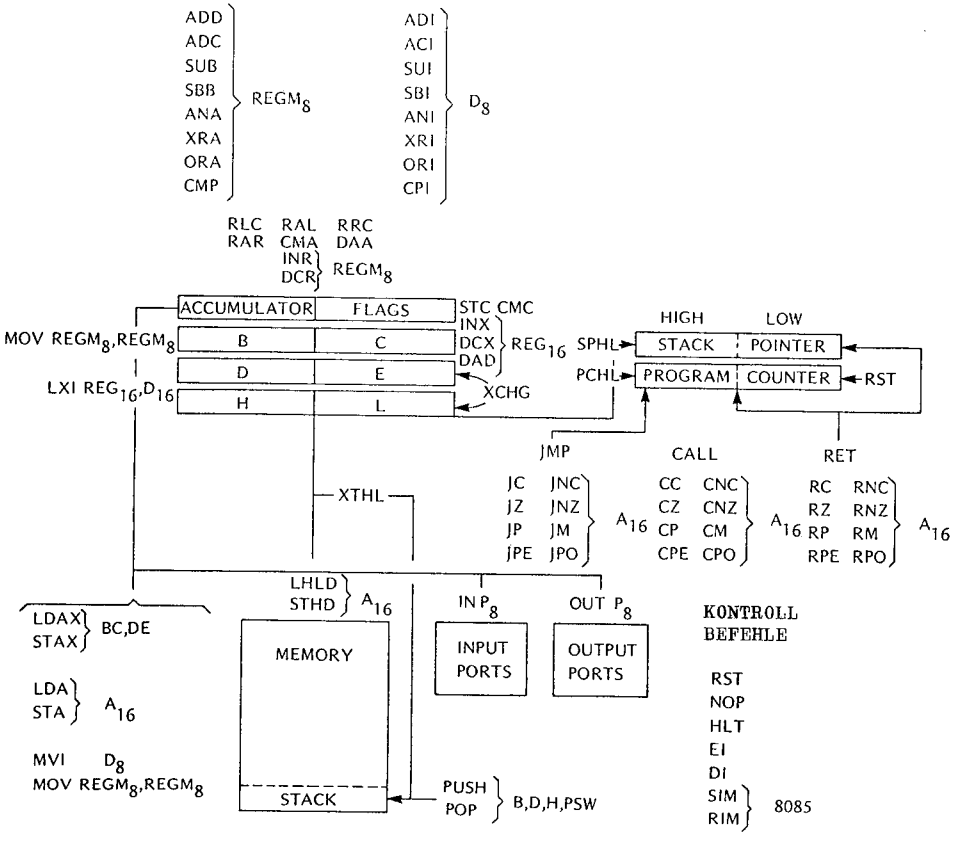
Das Programm in Beisp.11.15 zeigt, wie über die Tastatur des MICO 85 der Motor zum Rechts- und Linkslauf und zum Stillstand gebracht werden kann. Hierzu wird das Tastatur-Abfrageprogramm des Monitors (TASAB, Beisp.11.6) verwandt. Der Druck auf die Taste 0 heißt Linkslauf, auf 1 heißt Rechtslauf und jede andere Taste heißt Stillstand.

Beispiel 11.15: Drehrichtungsteuerung des Motors

```
1000  DB22     IN    22H    ; Port B einlesen, Ergebnis im Akku
1002  E6CF     ANI   0CFH   ; UND-Verkn.des Akkus mit CF: Bits 4 u.5=L
1004  47       MOV   B,A    ; Akkuinhalt in Register B sichern
1005  CD1B00   CALL  TASAB  ; Tastatur abfragen, Tastenwert im Akku.
1008  FEFF     CPI   0FFH   ; Keine Taste gedrückt? (Kennzeichen FF)
100A  CA0510   JZ    1005H  ; Tastatur erneut abfragen
100D  0E10     MVI   C,10H  ; Bit4=H für Linkslauf setzen
100F  FE00     CPI   00     ; Linkslauf ? (Taste 0)
1011  CA1D10   JZ    101DH  ; Wenn ja, Steuersignale geben
1014  0E20     MVI   C,20H  ; Bit5=H für Rechtslauf setzen
1016  FE01     CPI   01     ; Rechtslauf? (Taste 1)
1018  CA1D10   JZ    101DH  ; Wenn ja, Sprung!
101B  0E30     MVI   C,30H  ; Beide Steuerbits=H für Stopp
101D  78       MOV   A,B    ; Port-B-Wert zurück in den Akku
101E  81       ADD   C      ; Entsprechende Steuerbits H setzen
101F  D322     OUT   22H    ; Ausgabe auf Port B
1022  C30010   JMP   1000H  ; Sprung an den Anfang
```

D) Anhang

1. Zusammenfassende Darstellung der Funktionen der Befehle des 8080/85



Bedeutung der Abkürzungen

$REGM_8$	Der Operand spezifiziert eines der 8 Bit Register A...L oder einen Speicher M, welcher über das Registerpaar HL adressiert wird.
D_8	8 Bit Datenwort
A_{16}	16 Bit Adresse
P_8	8 Bit Port-Nummer
REG_{16}	16 Bit Registerpaar (BC,DE,HL,SP)
D_{16}	16 Bit Datenwort

2. Ergänzende Tabellen zu den Befehlen des 8080/85

a) Befehlsvorrat des MP8080/85 in numerischer Folge

Es bedeutet: D8 bzw. D16 = 8 bzw. 16bit Konstante.
Adr = 16bit Adresse.
* = 8085 Zusatzbefehle

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	2B	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	SHLX*
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM*	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SPD16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	JNX 5*
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	DSUB*	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAXB	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	LDS I*	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	ARHL*	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	LHLX*
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	RDEL*	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAXD	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DRC E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ Adr	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	RSTV*	F6	ORI D8
20	RIM*	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	JX 5*
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	LDHI*	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

Listen für die Mikroprozessoren Z80 und NSC800 enthält das Buch: "HANNEMANN, Programmierung von Mikroprozessoren I, Verlag W.Girardet".

2b) Bedeutung der Mnemoniks

ACI	Add immediate to accu with carry
ADC	Add to accu with carry
ADD	Add to accu
ADI	Add immediate to accu
ANA	And accu
ANI	And immediate with accu
ARHL	Arithmetic right shift of HL
C	Carry-flag
CALL	Call
CC	Call on carry
CM	Call on minus
CMA	Complement accu
CMC	Complement carry
CMP	Compare with accu
CNC	Call on no carry
CNZ	Call on no zero
CP	Call on positiv
CPE	Call on parity even
CPI	Compare immediate with accu
CPO	Call on parity odd
CZ	Call on zero
DAA	Decimal adjust accu
DAD	Add to H&L
DCR	Decrement
DCX	Decrement register pair
DI	Disable interrupts
DSUB	Double subtract
EI	Enable interrupts
H	Half-carry-flag
HLT	Halt
IN	Input
INR	Increment
INX	Increment register pair
JC	Jump on carry
JM	Jump on minus
JMP	Jump
JNC	Jump on no carry
JNX5	Jump on no X5
JNZ	Jump on no zero
JP	Jump on positiv
JPE	Jump on parity even
JPO	Jump on parity odd
JX5	Jump on X5
JZ	Jump on zero
LDA	Load accu direct
LDAX	Load accu indirect
LDHI	Load DE with HL + immediate byte
LDSI	Load DE with SP + immediate byte
LHLD	Load H&L direct
LHLX	Load HL indirect through DE
LXI	Load register pair immediate
M	Memory
MOV	Move
MVI	Move immediate
NOP	No operation
ORA	Or with accu
ORI	Or immediate with accu
OUT	Output
P	Parity-flag
PCHL	H&L to program counter
POP	Pop
PSW	Processor status word
PUSH	Push
RAL	Rotate accu left
RAR	Rotate accu right
RC	Return on carry
RDEL	Rotate DE left through carry
RET	Return
RIM	Read interrupt Mask
RLC	Rotate accu left through carry
RM	Return on minus
RNC	Return on no carry
RNZ	Return on no zero
RP	Return on positiv
RPE	Return on parity even
RPO	Return on parity odd
RRC	Rotate accu right through carry
RST	Restart
RSTV	Restart V on overflow
RZ	Return on zero
S	Sign-flag
SBB	Subtract from accu with borrow
SBI	Subtract immediate from accu with borrow
SHLD	Store H&L direct
SHLX	Store HL indirect through DE
SIM	Set interrupt Mask
SPHL	H&L to stackpointer
STA	Store accu direct
STAX	Store accu indirect
STC	Set carry
SUB	Subtract from accu
SUI	Subtract immediate from accu
XCHG	Exchange
XRA	Exclusive Or with accu
XRI	Exclusive Or immediate with accu
XTHL	Exchange top of stack H&L
Z	Zero-flag

3. ASCII – Tabelle

* Graphische Darstellung der Steuerzeichen (DIN 66213)

Hex.	Dez.	ASCII	Bedeutung englisch	CTRL +	Bedeutung deutsch	*
00	0	NUL	NULL	@	Null,Nichts	
01	1	SOH	START OF HEADING	A	Kopfzeilenbeginn	
02	2	STX	START OF TEXT	B	Textanfangzeichen	
03	3	ETX	END OF TEXT	C	Textendezeichen	
04	4	EOT	END OF TRANSMISSION	D	Ende der Übertragung	
05	5	ENQ	ENQUIRY	E	Aufford.zur Datenübertrag.	
06	6	ACK	ACKNOWLEDGE	F	Positive Rückmeldung	
07	7	BEL	BELL	G	Klingelzeichen	
08	8	BS	BACK SPACE	H	Rückwärtsschritt	
09	9	HT	HORIZONT.TABULATION	I	Horizontal Tabulator	
0A	10	LF	LINE FEED	J	Zeilenvorschub	
0B	11	VT	VERTICAL TABULATION	K	Vertikal Tabulator	
0C	12	FF	FORM FEED	L	Seitenvorschub	
0D	13	CR	CARRIAGE RETURN	M	Wagenrücklauf	
0E	14	SO	SHIFT OUT	N	Dauerumschaltungszeichen	
0F	15	SI	SHIFT IN	O	Rückschaltungszeichen	
10	16	DLE	DATA LINK ESCAPE	P	Datenübertragungsumschalt.	
11	17	DC1	DEVICE CTRL 1 (X-ON)	Q	Gerätesteuerzeichen 1	
12	18	DC2	DEV.CONTROL 2 (TAPE)	R	Gerätesteuerzeichen 2	
13	19	DC3	DEVIC.CTRL 3 (X-OFF)	S	Gerätesteuerzeichen 3	
14	20	DC4	DEVICE CTRL 4 (TAPE)	T	Gerätesteuerzeichen 4	
15	21	NAK	NEGATIVE ACKNOWLEDG.	U	Negative Rückmeldung	
16	22	SYN	SYNCHRONOUS IDLE	V	Synchronisierung	
17	23	ETB	END OF TRANSM.BLOCK	W	Ende des Übertragungsblocks	
18	24	CAN	CANCEL	X	Ungültig	
19	25	EM	END OF MEDIUM	Y	Ende der Aufzeichnung	
1A	26	SUB	SUBSTITUTE	Z	Substitution	
1B	27	ESC	ESCAPE	[	Umschaltung	
1C	28	FS	FILE SEPARATOR	/	Hauptgruppentrennzeichen	
1D	29	GS	GROUP SEPARATOR	]	Gruppentrennzeichen	
1E	30	RS	RECORD SEPARATOR	^	Untergruppentrennzeichen	
1F	31	US	UNIT SEPARATOR	_	Teilgruppentrennzeichen	
20	32	SP	SPACE		Leerzeichen	
21	33	!	EXCLAMATION MARK		Ausrufzeichen	
22	34	"	QUOTATION MARK		Anführungszeichen	
23	35	#	NUMBER SIGN		Nummerzeichen	
24	36	$	DOLLAR SIGN		Dollarzeichen	
25	37	%	PERCENT SIGN		Prozentzeichen	
26	38	&	AMPERSAND		Kommerzielles UND-Zeichen	
27	39	'	APOSTROPHE		Hochkomma	
28	40	(	OPENING PARENTHESIS		Runde Klammer (offen)	
29	41	)	CLOSING PARENTHESIS		Runde Klammer (geschlossen)	
2A	42	*	ASTERISK		Stern	
2B	43	+				
2C	44	,				
2D	45	-				
2E	46	.				
2F	47	/				
30	48	0				
31	49	1				
32	50	2				
33	51	3				
34	52	4				
35	53	5				
36	54	6				
37	55	7				
38	56	8				
39	57	9				
3A	58	:				
3B	59	;				
3C	60					
3D	61	=				
3E	62					
3F	63	?				
40	64	@ (§)				
41	65	A				
42	66	B				
43	67	C				
44	68	D				
45	69	E				
46	70	F				
47	71	G				
48	72	H				
49	73	I				
4A	74	J				
4B	75	K				
4C	76	L				
4D	77	M				
4E	78	N				
4F	79	O				
50	80	P				
51	81	Q				
52	82	R				
53	83	S				
54	84	T				
55	85	U				
56	86	V				
57	87	W				
58	88	X				
59	89	Y				
5A	90	Z				
5B	91	[(Ä)				
5C	92	\ (Ö)				
5D	93	] (Ü)				
5E	94	^				
5F	95	_				
60	96	`				
61	97	a				
62	98	b				
63	99	c				
64	100	d				
65	101	e				
66	102	f				
67	103	g				
68	104	h				
69	105	i				
6A	106	j				
6B	107	k				
6C	108	l				
6D	109	m				
6E	110	n				
6F	111	o				
70	112	p				
71	113	q				
72	114	r				
73	115	s				
74	116	t				
75	117	u				
76	118	v				
77	119	w				
78	120	x				
79	121	y				
7A	122	z				
7B	123	{ (ä)				
7C	124	| (ö)				
7D	125	} (ü)				
7E	126	~ (ß)				
7F	127	DEL				
└─	*	///				

4. Monitorprogramm für den MICO 85

```
                ;******  Monitorprogramm fuer den MICO 85  ******

                ;Arbeitsspeicheraufteilung:
                ;Anzeigepuffer Anzeige 1...8 : 2000...2007H
                ;Interruptvermittelung RST1...RST7.5 : 2008...2028
                ;ARSP+     (ARSP = 2029H)
                ; +0           I Befehlsspeicher
                ; +1 Anf.Adr.  I fuer die
                ; +2 RAM-Test. I direkte Ein- und
                ; +3           I Ausgabe auf
                ; +4           I Ports.
                ; +5
                ; +6  Anfang Arbeitsspeicher des Testsystemes
                ; +18 Ende        "                 "

0000 310021      LXI  SP,STACK ;Stackpointer laden
0003 C36F00      JMP  INITI    ;INT- und UP-Verzweigung ueberspringen

                ;Interruptvektoren auf den Arbeitsspeicher richten und
                ;Einspruenge in die Service-Routinen (SR) festlegen.
0006             DS   2        ;Adresse um 2 erhoehen
0008 C30820      JMP  INT      ;Sprung in den Arbeitsspeicher bei RST1
000B C34103      JMP  ZA1MS    ;SR: Zeitschleife (A)*1ms
000E             DS   2
0010 C30B20      JMP  INT+3    ;RST2
0013 C38503      JMP  TASDR    ;SR: Warten auf Tastendruck.
0016             DS   2        ;Der Tastenwert wird in A uebergeben.
0018 C30E20      JMP  INT+6    ;RST3
001B C39F03      JMP  TASAB    ;SR: Tastaturabfrage, Tastenwert in A.
001E             DS   2        ;(A) = FF : keine Taste gedrueckt.
0020 C31120      JMP  INT+9    ;RST4
0023 00          NOP
0024 C31420      JMP  INT+12   ;TRAP
0027 00          NOP
0028 C31720      JMP  INT+15   ;RST5
002B 00          NOP
002C C31A20      JMP  INT+18   ;RST5.5
002F 00          NOP
0030 C31D20      JMP  INT+21   ;RST6
0033 00          NOP
0034 C32020      JMP  INT+24   ;RST6.5
0037 00          NOP
0038 C32320      JMP  INT+27   ;RST7
003B 00          NOP
003C C32620      JMP  INT+30   ;RST7.5

                ;Weitere Service-Routinen (Einspruenge in UPs)
003F C35702      JMP ANZ12 ;(B) : Anzeigestelle 1.+2.
0042 C35E02      JMP ANZ34 ;(B) :       "       3.+4.
0045 C36502      JMP ANZ56 ;(B) :       "       5.+6.
0048 C36C02      JMP ANZ78 ;(B) : Anzeigestelle 7.+8.
004B C38802      JMP ADRAN ;(DE) : Anzeigestelle 3.-6.
004E C31B03      JMP NULL  ;Anzeigepuffer FFFFFFFF setzen
0051 C3D903      JMP ANZEI ;Anzeigepuffer zur Anzeige bringen
0054 C3B502      JMP DALES ;1Byte von Tast.(B) und anzeigen (1.+2.Anz.)
0057 C3D002      JMP ADLE  ;Adresse von Tast.einlesen (DE) und anzeigen
005A C35003      JMP KOPIE ;Copiere Speicherbereich (HL)-(BC) : (DE)
005D C3FE02      JMP ANLES ;Anzeige lesen:(B)=Nr.d.Anz.,: (B)=1Anz.Byte
0060 C3F600      JMP ANFA  ;Warmstart-Adresse, FFFFFFFF : Anzeige
0063 C38E00      JMP ANFA1 ;Einsprung in den Kommandobereich
0066 C39E02      JMP GDRT0 ;1.+2.Anz.um eine Stelle nach links+ (B):1.
0069 C39302      JMP GDRT1 ;3.-5.Anz.um eine Stelle nach links+ (B):3.
006C C32C03      JMP KOKE  ;Kommandokennzahl holen/anzeigen : (A)
```

```
                    ;Initialisierungen
006F 3E03           INITI: MVI    A,3       ;Steuerwort laden (PA+B=Aus,PC=Ein)
0071 D320                  OUT    SR        ;Portbaustein 8155 programmieren
0073 210820                LXI    H,INT     ;Arsp.Adr. zur Interruptverzweigung
0076 3E21                  MVI    A,33      ;Den Speicherbereich, auf den die
0078 36C9           BERTA: MVI    M,0C9H    ;Interrupt-Vektoren gerichtet
007A 23                    INX    H         ;wurden, mit C9 (RET) laden.
007B 3D                    DCR    A
007C C27800                JNZ    BERTA

                    ;Welcher RAM-Bereich steht zur Verfuegung?
007F 7F                    MOV    A,A       ;Flag-Befehl
0080 210007                LXI    H,700H    ;Anfangsadresse RAM-Test
0083 222A20                SHLD   ARSP+1    ;Teststartadresse sichern
0086 CDEB01                CALL   RAMTS     ;RAM-Test
0089 3E00                  MVI    A,0
008B 323B20                STA    ARSP+18 ;Kennung "kein Einspr.vom Testpro."

                    ;Kommandoannahme und Verzweigung
008E CD1B03         ANFA1: CALL   NULL      ;Anzeigepuffer FFFFFFFF setzen
0091 CD8503                CALL   TASDR     ;Warte auf Tastendruck : A
0094 F5                    PUSH   PSW       ;Hauptkommando-Kennzahl sichern
0095 07                    RLC              ;Kennzahl in hoeherwertige
0096 07                    RLC              ;Tetrade bringen.
0097 07                    RLC
0098 07                    RLC
0099 F60F                  ORI    0FH       ;Unterkomm.Kennzahl = F setzen
009B 47                    MOV    B,A       ;Hauptkommando-Kennzahl
009C CD6C02                CALL   ANZ78     ;auf die Anzeige 8 bringen
009F 3E40                  MVI    A,40H     ;SOD auf L setzen und damit die
00A1 30                    DB     30H       ;Dezimalpunkte (Anz.) einschalten.

00A2 F1                    POP    PSW       ;Hauptkomm.Kennz. zurueck holen
00A3 FE0A                  CPI    0AH       ;- Speicher-Inhalt anzeigen/
00A5 CA4E01                JZ     RAMAN     ;  aendern.
00A8 FE0D                  CPI    0DH       ;- Direktes Lesen und Ausgabe
00AA CA8601                JZ     PORTA     ;  auf Ports.
00AD FE0F                  CPI    0FH       ;- RAM-Bereich fuellen.
00AF CA3201                JZ     RAMSE
00B2 FE01                  CPI    1         ;- Einschreiben von Programmen
00B4 CADB01                JZ     EIN       ;  oder Daten.
00B7 FE0B                  CPI    0BH       ;- Starten eines Programmes.
00B9 CA2901                JZ     START
00BC FE00                  CPI    0         ;- Weiteren RAM-Bereich suchen.
00BE CCEB01                CZ     RAMTS
00C1 FE0C                  CPI    0CH       ;- RAM-Inhalt copieren.
00C3 CA1701                JZ     RAMVE
00C6 FE0E                  CPI    0EH       ;- Einzelschritt Testsystem.
00C8 CAFA03                JZ     SINGL
00CB FE02                  CPI    2         ;- Programmstart auf 4000H.
00CD CA0040                JZ     4000H
00D0 FE03                  CPI    3         ;- Programmstart auf 1000H.
00D2 CA0010                JZ     1000H     ;  bzw.Einspr.ASCII-Tast.+Bildsch.
00D5 FE04                  CPI    4         ;- Audiokassette lesen oder
00D7 CA0008                JZ     800H      ;  beschreiben.
00DA FE05                  CPI    5         ;- Relativierbares Programm
00DC CA0308                JZ     803H      ;  Verschieben.
00DF FE06                  CPI    6         ;- EPROMs (2716/32) lesen
00E1 CA0608                JZ     806H      ;  oder programmieren.
00E4 FE09                  CPI    9         ;- Speicherinhalte vergleichen.
00E6 CA0908                JZ     809H
00E9 FE07                  CPI    7         ;- Programmstart auf 1800H
00EB CA0018                JZ     1800H
00EE FE08                  CPI    8         ;- Speicherinhalt ausdrucken,
00F0 CA0310                JZ     1003H     ;  anzeigen, disassemblieren.
00F3 C38E00                JMP    ANFA1
```

```
                ;Ruecksprungvermittelung

00F6 CD1B03   ANFA:  CALL    NULL     ;FFFFFFFF : Anzeigepuffer
00F9 3A3B20          LDA     ARSP+18  ;Kennung Testprogr.oder nicht
00FC FE01            CPI     1
00FE C20E01          JNZ     MARTA
0101 3E00            MVI     A,0      ;Kennung ruecksetzen
0103 323B20          STA     ARSP+18
0106 06E0            MVI     B,0E0H
0108 CD6C02          CALL    ANZ78
010B C3FD03          JMP     SINGL+3  ;Sprung ins Testprogramm
010E CDD903   MARTA: CALL    ANZEI    ;F : Anzeige
0111 3EC0            MVI     A,0C0H   ;SOD auf H setzen
0113 30              DB      30H      ;SIM
0114 C38E00          JMP     ANFA1


                ;KOMMANDO  C  Speicherinhalt copieren *****************

0117 CDD002   RAMVE: CALL    ADLE     ;Adresse einlesen : DE
011A EB              XCHG             ;Anfangsadresse : HL
011B CDD002          CALL    ADLE
011E 4B              MOV     C,E      ;Endadresse : BC
011F 42              MOV     B,D
0120 CDD002          CALL    ADLE     ;Zieladresse : DE
0123 CD5003          CALL    KOPIE
0126 C3F600          JMP     ANFA


                ;KOMMANDO  B  Starten von Programmen ******************

0129 CDD002   START: CALL    ADLE     ;Startadresse lesen : DE
012C EB              XCHG
012D 46              MOV     B,M      ;Erstes Byte des Programmes : B
012E CD5702          CALL    ANZ12    ;auf die Anzeige (1.+2.) bringen
0131 E9              PCHL             ;Dort das Programm fortsetzen


                ;KOMMANDO  F  RAM-Bereich fuellen *********************

0132 CDD002   RAMSE: CALL    ADLE     ;Adresse lesen und anzeigen
0135 EB              XCHG             ;RAM-Anfangsadresse : HL
0136 CDD002          CALL    ADLE
0139 42              MOV     B,D      ;RAM-Endadresse : BC
013A 4B              MOV     C,E
013B E5              PUSH    H
013C C5              PUSH    B
013D CDB502          CALL    DALES    ;2 Tetraden einlesen
0140 78              MOV     A,B      ;Datenbyte : A
0141 C1              POP     B
0142 2B              DCX     H
0143 23       SP10:  INX     H
0144 77              MOV     M,A      ;Fuellbyte : RAM
0145 E5              PUSH    H
0146 08              DB      8        ;DSUB: Endadr. erreicht ?
0147 E1              POP     H
0148 CAF600          JZ      ANFA
014B C34301          JMP     SP10


                ;KOMMANDO  A  RAM-Inhalt anzeigen/aendern *************

014E CDD002   RAMAN: CALL    ADLE     ;Speicheradresse : DE
0151 1A              LDAX    D        ;Speicherinhalt : (A)
0152 47              MOV     B,A
```

```
0153 CD5702          CALL    ANZ12
0156 CD2C03   NOCH:  CALL    KOKE    ;Unterkomm.-Kennzahl holen:A
0159 FE00            CPI     00
015B CA7201          JZ      OHNE    ;Sprung, wenn ohne Aenderung
015E FE01            CPI     01
0160 CA7E01          JZ      MITA    ;Sprung, wenn mit Aenderung
0163 FE02            CPI     2
0165 CA7001          JZ      ZURU    ;Einen Speicherplatz zurueck
0168 FE03            CPI     3
016A CAF600          JZ      ANFA
016D C35601          JMP     NOCH
0170 1B       ZURU:  DCX     D
0171 1B              DCX     D
0172 13       OHNE:  INX     D       ;Speicheradresse + 1
0173 CD8802          CALL    ADRAN   ;Neue Speicheradresse anzeigen
0176 1A       ANNA:  LDAX    D       ;Speicherplatzinhalt : A
0177 47              MOV     B,A
0178 CD5702          CALL    ANZ12   ;Speicherinhalt zur Anzeige
017B C35601          JMP     NOCH
017E CDB502   MITA:  CALL    DALES   ;zwei Hexzeichen einlesen : (B)
0181 78              MOV     A,B
0182 12              STAX    D       ;Datenbyte ins RAM
0183 C37601          JMP     ANNA

                     ;KOMMANDO  D  Direkte Ein- und Ausgabe auf Ports  *****

0186 CDD002   PORTA: CALL    ADLE    ;Portadresse einlesen : DE
0189 1600     HANNA: MVI     D,0
018B CD8802          CALL    ADRAN   ;(DE) : Anz. 3.-6.
018E 212920          LXI     H,ARSP  ;IN- oder OUT-Befehl ablegen
0191 36DB            MVI     M,0DBH  ;IN
0193 23              INX     H
0194 73              MOV     M,E     ;Portadresse
0195 23              INX     H
0196 36C3            MVI     M,0C3H  ;Ruecksprung
0198 23              INX     H
0199 01D901          LXI     B,PORT+1 ;Ruecksprungadresse laden
019C 0A              LDAX    B
019D 77              MOV     M,A
019E 23              INX     H
019F 03              INX     B
01A0 0A              LDAX    B
01A1 77              MOV     M,A
01A2 C32920          JMP     ARSP    ;Befehlsausfuehrung  IN
01A5 47       PORTS: MOV     B,A
01A6 CD5702          CALL    ANZ12   ;Eingelesenes Byte zur Anzeige
01A9 CD9F03   FALS:  CALL    TASAB
01AC FEFF            CPI     0FFH    ;Taste gedrueckt?
01AE CA8901          JZ      HANNA
01B1 CD2C03          CALL    KOKE    ;Unterk.-Kennz. holen/anz.:A
01B4 FE03            CPI     3
01B6 CAF600          JZ      ANFA
01B9 FE01            CPI     1
01BB CACE01          JZ      INA     ;aendern
01BE 1D              DCR     E       ;Portadr. - 1
01BF FE02            CPI     2
01C1 CA8901          JZ      HANNA
01C4 1C              INR     E       ;Portadr.+1
01C5 1C              INR     E
01C6 FE00            CPI     0
01C8 CA8901          JZ      HANNA
01CB C3A901          JMP     FALS
01CE CDB502   INA:   CALL    DALES   ;neues Datum einlesen : B
01D1 78              MOV     A,B
```

```
01D2 212920          LXI     H,ARSP
01D5 36D3            MVI     M,0D3H  ;OUT
01D7 E9              PCHL    ;Befehlsausfuehrung
01D8 C3A501    PORT: JMP     PORTS

               ;KOMMANDO  1  Einschreiben von Programmen oder Daten  *

01DB CDD002    EIN:  CALL    ADLE    ;Anfangsadresse einlesen
01DE EB        HERA: XCHG
01DF CDB502          CALL    DALES   ;Zwei Zeichen einlesen : B
01E2 70              MOV     M,B
01E3 23              INX     H
01E4 EB              XCHG
01E5 CD8802          CALL    ADRAN   ;Neue Adresse anzeigen
01E8 C3DE01          JMP     HERA

               ;------------------ Unterprogramme -----------------

               ;RAM - Test
01EB C5        RAMTS: PUSH   B
01EC D5              PUSH    D
01ED E5              PUSH    H
01EE F5              PUSH    PSW
01EF 3EC0            MVI     A,0C0H
01F1 30              DB      30H     ;SIM :  SOD=H : Dez.Pkt aus
01F2 2A2A20          LHLD    ARSP+1  ;Beginn der Suche
01F5 3E01            MVI     A,1     ;Kennung:" Beginn RAM Test"
01F7 322920          STA     ARSP
01FA 23        OBEN: INX     H
01FB 3E55            MVI     A,55H
01FD CD2302          CALL    PRUEF
0200 C23E02          JNZ     FEHL
0203 3EAA            MVI     A,0AAH
0205 CD2302          CALL    PRUEF
0208 C23E02          JNZ     FEHL
020B 3A2920          LDA     ARSP    ;Kennung laden
020E FE01            CPI     1       ;Anfang?
0210 C2FA01          JNZ     OBEN
0213 44              MOV     B,H     ; RAM-Anfangsadresse
0214 CD6C02          CALL    ANZ78   ; zur Anzeige bringen
0217 45              MOV     B,L
0218 CD6502          CALL    ANZ56
021B 3E00            MVI     A,0     ;Kennung ruecksetzen
021D 322920          STA     ARSP
0220 C3FA01          JMP     OBEN
0223 2F        PRUEF: CMA
0224 2B              DCX     H
0225 46              MOV     B,M
0226 77              MOV     M,A
0227 23              INX     H
0228 23              INX     H
0229 4E              MOV     C,M
022A 77              MOV     M,A
022B 2F              CMA
022C 2B              DCX     H
022D 56              MOV     D,M
022E 77              MOV     M,A
022F 23              INX     H
0230 2F              CMA
0231 77              MOV     M,A
0232 2B              DCX     H
0233 5E              MOV     E,M
0234 23              INX     H
```

```
0235 71              MOV     M,C
0236 2B              DCX     H
0237 72              MOV     M,D
0238 2B              DCX     H
0239 70              MOV     M,B
023A 23              INX     H
023B 2F              CMA
023C BB              CMP     E
023D C9              RET
023E 3A2920    FEHL: LDA     ARSP    ;Kennung laden
0241 FE01            CPI     1       ;Anfang suchen?
0243 CAFA01          JZ      OBEN
0246 222A20          SHLD    ARSP+1  ;RAM-Endadresse sichern
0249 2B              DCX     H
024A 44              MOV     B,H     ; RAM-Endadresse
024B CD5E02          CALL    ANZ34   ; zur Anzeige bringen
024E 45              MOV     B,L
024F CD5702          CALL    ANZ12
0252 F1              POP     PSW
0253 E1              POP     H
0254 D1              POP     D
0255 C1              POP     B
0256 C9              RET

                ;(B) zur Anzeige bringen

0257 E5        ANZ12: PUSH   H
0258 210020           LXI     H,ANPU  ;Adr. des Anzeigepuffers
025B C37402           JMP     LOLA
025E E5        ANZ34: PUSH   H
025F 210220           LXI     H,ANPU+2
0262 C37402           JMP     LOLA
0265 E5        ANZ56: PUSH   H
0266 210420           LXI     H,ANPU+4
0269 C37402           JMP     LOLA
026C E5        ANZ78: PUSH   H
026D 210620           LXI     H,ANPU+6
0270 C37402           JMP     LOLA
0273 00               NOP
0274 F5        LOLA:  PUSH    PSW
0275 78               MOV     A,B
0276 E60F             ANI     0FH     ;niederw.Tetrade aussieben
0278 77               MOV     M,A
0279 78               MOV     A,B
027A E6F0             ANI     0F0H    ;hoeherw.Tetrade aussieben
027C 0F               RRC             ;und in niederw. schieben
027D 0F               RRC
027E 0F               RRC
027F 0F               RRC
0280 23               INX     H
0281 77               MOV     M,A
0282 CDD903           CALL    ANZEI   ;Anzeigepuffer zur Anzeige
0285 F1               POP     PSW
0286 E1               POP     H
0287 C9               RET

                ;Adresse (DE) anzeigen auf 3.bis 6. Anzeigestelle

0288 C5        ADRAN: PUSH    B
0289 42               MOV     B,D
028A CD6502           CALL    ANZ56
028D 43               MOV     B,E
028E CD5E02           CALL    ANZ34
0291 C1               POP     B
0292 C9               RET
```

```
                ;Anzeigeninhalt nach links schieben und rechts den
                ;Inhalt von B (niederw.) einfuegen.
0293 F5         GDRT1: PUSH   PSW       ;3.-5. Anzeigenstelle
0294 E5                PUSH   H
0295 C5                PUSH   B
0296 210220            LXI    H,ANPU+2 ;Anzeigepuffer (DS3)
0299 0E04              MVI    C,4      ;Vier Ziffern : Anzeigepuffer
029B C3A602            JMP    ANSCH    ;3.-6.Anzeigest.von rechts laden
029E F5         GDRTO: PUSH   PSW      ;1.+2. Anzeigenstelle
029F E5                PUSH   H
02A0 C5                PUSH   B
02A1 210020            LXI    H,ANPU   ;Anz.-puffer 1.+2. von rechts lad.
02A4 0E02              MVI    C,2

                ;Ziffern nach links schieben und rechts eine neue anfueg.
02A6 7E         ANSCH: MOV    A,M      ; Alter Pufferinhalt : A
02A7 70                MOV    M,B      ; Neuen Pufferinhalt laden
02A8 47                MOV    B,A      ; Alten Wert aus A nach B
02A9 23                INX    H        ; Pufferadresse +1
02AA 0D                DCR    C        ; Anzeigestellenzaehler - 1
02AB C2A602            JNZ    ANSCH    ; Sprung,bis alle durchgeschoben
02AE CDD903            CALL   ANZEI    ; Anzeigepuffer zur Anzeige
02B1 C1                POP    B
02B2 E1                POP    H
02B3 F1                POP    PSW
02B4 C9                RET

                ;Datum einlesen (Anzeige 1.+2.) : B
02B5 D5         DALES: PUSH   D
02B6 E5                PUSH   H
02B7 F5                PUSH   PSW
02B8 1602              MVI    D,2
02BA CD8503     RUTH:  CALL   TASDR    ;Warten auf Tastendruck
02BD 47                MOV    B,A
02BE C5                PUSH   B
02BF CD9E02            CALL   GDRTO    ;in ANZ12 einschieben
02C2 C1                POP    B
02C3 15                DCR    D
02C4 C2BA02            JNZ    RUTH
02C7 0601              MVI    B,1
02C9 CDFE02            CALL   ANLES    ;Anzeige 1 u.2 lesen : B
02CC F1                POP    PSW
02CD E1                POP    H
02CE D1                POP    D
02CF C9                RET

                ;Adresse (DE) von Tastatur holen und anzeigen
02D0 E5         ADLE:  PUSH   H
02D1 C5                PUSH   B
02D2 F5                PUSH   PSW
02D3 CD8503            CALL   TASDR    ;Warten auf Tastendruck
02D6 06FF              MVI    B,0FFH
02D8 CD5E02            CALL   ANZ34    ;FF in die Anzeige bringen
02DB CD6502            CALL   ANZ56
02DE 0E04              MVI    C,4      ;4 mal auf Tastendruck warten
02E0 C3E602            JMP    RIKI
02E3 CD8503     NICHT: CALL   TASDR    ;Auf Tastendruck warten
02E6 47         RIKI:  MOV    B,A      ;Tastenwert : B
02E7 CD9302            CALL   GDRT1    ;Anzeigen
02EA 0D                DCR    C
02EB C2E302            JNZ    NICHT    ;Sprung, wenn nicht 4. Zeichen
02EE 0603              MVI    B,3      ; Anzeige 3 + 4 lesen
02F0 CDFE02            CALL   ANLES
02F3 58                MOV    E,B
```

```
02F4 0605           MVI     B,5     ;Anzeige 5 + 6 lesen
02F6 CDFE02         CALL    ANLES
02F9 50             MOV     D,B
02FA F1             POP     PSW
02FB C1             POP     B
02FC E1             POP     H
02FD C9             RET

                    ;Anzeige lesen, 2 Werte vom Anzeigepuffer : B
                    ;Aufruf: Nummer der niederw.Anzeige (Nr.) in B
                    ;Rueckgabe: (B) = Anzeigenwert von Nr.+1 und von Nr.
02FE F5         ANLES: PUSH     PSW
02FF E5                PUSH     H
0300 C5                PUSH     B
0301 05                DCR      B       ;Nr.-1
0302 48                MOV      C,B
0303 0600              MVI      B,0
0305 210020            LXI      H,ANPU  ;Adr. des Anz.-puffers fuer Nr.1
0308 09                DAD      B       ;ANPU + Nr. - 1
0309 C1                POP      B
030A 7E                MOV      A,M     ;1. Zeichen : A
030B E60F              ANI      0FH     ;Hoeherwertige Tetrade = 0
030D 47                MOV      B,A
030E 23                INX      H
030F 7E                MOV      A,M     ;2. Zeichen : A
0310 07                RLC              ;Zeichen in die
0311 07                RLC              ;hoeherwertige
0312 07                RLC              ;Tetrade schieben.
0313 07                RLC
0314 E6F0              ANI      0F0H    ;Niederwertige  Tetrade = 0
0316 80                ADD      B       ;Beide Zeichen zusammenfuegen
0317 47                MOV      B,A     ;und ins Reg.B bringen.
0318 E1                POP      H
0319 F1                POP      PSW
031A C9                RET

                    ;FFFFFFFF in den Anzeigepuffer (loeschen)
031B C5         NULL:  PUSH     B
031C E5                PUSH     H
031D 210020            LXI      H,ANPU  ;Anzeigepuffer-Adresse
0320 0608              MVI      B,8     ;Pufferlaenge
0322 360F       SP1:   MVI      M,0FH
0324 23                INX      H
0325 05                DCR      B
0326 C22203            JNZ      SP1
0329 E1                POP      H
032A C1                POP      B
032B C9                RET

                    ;Unterkommando-Kennzahl holen/anzeigen : A
032C C5         KOKE:  PUSH     B
032D CD8503            CALL     TASDR   ;Warten auf Tastendruck
0330 4F                MOV      C,A     ;Tastenwert sichern
0331 0607              MVI      B,7     ;Anzeigestelle Nr.7 + 8
0333 CDFE02            CALL     ANLES   ;lesen und in B uebergeben.
0336 78                MOV      A,B
0337 E6F0              ANI      0F0H
0339 81                ADD      C
033A 47                MOV      B,A
033B CD6C02            CALL     ANZ78   ;Haupt- und Unterkomm.(B) anzeigen
033E 79                MOV      A,C
033F C1                POP      B
0340 C9                RET
```

```
                        ;Zeitschleife: Aufruf: (A)= Anzahl der ms
0341 F5                 ZA1MS: PUSH    PSW
0342 C5                        PUSH    B
0343 06D6               OBER:  MVI     B,0D6H
0345 05                 LAUF:  DCR     B
0346 C24503                    JNZ     LAUF
0349 3D                        DCR     A
034A C24303                    JNZ     OBER
034D C1                        POP     B
034E F1                        POP     PSW
034F C9                        RET


                        ;Anfangsadresse des zu kopierenden Bereiches in HL
                        ;Endadresse  des zu  kopierenden  Bereiches in  BC
                        ;           Anfangsadresse des Zielbereiches in  DE
0350 E5                 KOPIE: PUSH    H       ;Wenn die
0351 C5                        PUSH    B       ;Zieladresse
0352 D5                        PUSH    D       ;kleiner ist
0353 C1                        POP     B       ;als die
0354 08                        DB      8;(DSUB);Anfangsadresse
0355 C1                        POP     B       ;des zu kopierenden
0356 E1                        POP     H       ;Bereiches,
0357 D26503                    JNC     KLEIN   ;dann springe.
035A F5                        PUSH    PSW
035B C5                        PUSH    B
035C E5                        PUSH    H
035D C5                        PUSH    B       ;Berechnung
035E E1                        POP     H       ;der Endadresse
035F C1                        POP     B       ;des Zielbereiches.
0360 08                        DB      8;(DSUB);
0361 19                        DAD     D
0362 EB                        XCHG
0363 E1                        POP     H
0364 F1                        POP     PSW
0365 F5                 KLEIN: PUSH    PSW     ;Carry Flag sichern
0366 7E                        MOV     A,M     ;Erstes Byte
0367 12                        STAX    D       ;kopieren und
0368 EB                        XCHG            ;vergleichen ob
0369 BE                        CMP     M       ;richtig angekommen, sonst
036A C28203                    JNZ     ENDE    ;Abbruch : Speicherfehler!
036D EB                        XCHG
036E E5                        PUSH    H       ;Ist das Ende
036F 08                        DB      8;(DSUB);des zu kopierenden
0370 E1                        POP     H       ;Bereiches erreicht,
0371 CA8203                    JZ      ENDE    ;dann Sprung zum Ende.
0374 F1                        POP     PSW     ;Carry-Flag laden
0375 D27D03                    JNC     HOCH    ;Sprung, wenn zu einer klein.
0378 2B                        DCX     H       ;Adresse hin kopiert wird.
0379 1B                        DCX     D
037A C36503                    JMP     KLEIN   ;Naechstes Byte kopieren
037D 23                 HOCH:  INX     H
037E 13                        INX     D
037F C36503                    JMP     KLEIN   ;Naechstes Byte kopieren
0382 33                 ENDE:  INX     SP
0383 33                        INX     SP
0384 C9                        RET


                        ;Warten auf Tastendruck. Rueckgabe: Tastenwert in A
0385 C5                 TASDR: PUSH    B
0386 CD9F03             HILDE: CALL    TASAB   ;Tastenabfrage
0389 FEFF                      CPI     OFFH    ;Keine Taste gedrueckt?
038B CA8603                    JZ      HILDE
038E 47                        MOV     B,A     ;Tastenwert sichern
```

```
038F 3E64           MVI     A,100   ;100MS
0391 CD4103         CALL    ZA1MS   ;Prellen abwarten
0394 CD9F03 WARTE:  CALL    TASAB
0397 FEFF           CPI     0FFH
0399 C29403         JNZ     WARTE   ;Weiter wenn Taste losgelassen
039C 78             MOV     A,B     ;Tastenwert laden
039D C1             POP     B
039E C9             RET

                    ;Tastenabfrage, Die hoeherw.Tetrade des Ports 22 bleibt
                    ;erhalten.Rueckgabe: (A): Tastenwert, keine Taste: (A)=FF
039F C5     TASAB:  PUSH    B
03A0 D5             PUSH    D
03A1 0603           MVI     B,3     ;Tastenwert rechts unten
03A3 DB22           IN      PB      ;Spaltenport lesen
03A5 E6F0           ANI     0F0H    ;Hoeherw. Tetrade separieren
03A7 57             MOV     D,A     ;und speichern.
03A8 C601           ADI     1       ;1.Spalte H setzen
03AA C5     SPSCH:  PUSH    B       ;Tastenwert sichern
03AB D322           OUT     PB      ;Spalte H setzen
03AD E60F           ANI     0FH     ;Hoeherw.Tetrade=0
03AF F5             PUSH    PSW     ;Spalten-Wert sichern
03B0 DB23           IN      PC      ;Zeilen-Wert laden
03B2 0E04           MVI     C,04H   ;Schleifenzaehler= 4
03B4 0F     ZESCH:  RRC             ;Akku-Bit 0,1,2,3 ins Carry
03B5 DAD303         JC      TASGE   ;Sprung,wenn H gefunden
03B8 04             INR     B       ;B um 4 erhoehen, gleich
03B9 04             INR     B       ;Wert der darunter
03BA 04             INR     B       ;liegenden Taste.
03BB 04             INR     B
03BC 0D             DCR     C       ;Schleifenzaehler - 1
03BD C2B403         JNZ     ZESCH   ;Sprung,bis alle 4 Zeilen abgefr.
03C0 F1             POP     PSW     ;Spaltenzustand laden
03C1 C1             POP     B       ;Tas.-Wert rueckladen
03C2 05             DCR     B       ;Tas.-Wert links daneben
03C3 FE08           CPI     8       ;Letzte Spalte?
03C5 CACE03         JZ      GABI
03C8 87             ADD     A       ;H um 1 Spalte nach links
03C9 82             ADD     D       ;Hoeherw.Tetrade addieren
03CA C3AA03         JMP     SPSCH   ;naechste  Spalte H setzen
03CD 00             NOP
03CE 3EFF   GABI:   MVI     A,0FFH  ;Kennung:keine Tas.-gedr.
03D0 D1             POP     D
03D1 C1             POP     B
03D2 C9             RET
03D3 F1     TASGE:  POP     PSW     ;SP um 4 erhoehen
03D4 F1             POP     PSW
03D5 78             MOV     A,B     ;Tas.-Wert in Akku laden
03D6 D1             POP     D
03D7 C1             POP     B
03D8 C9             RET

                    ;Anzeigepuffer zur Anzeige bringen
03D9 C5     ANZEI:  PUSH    B
03DA E5             PUSH    H
03DB F5             PUSH    PSW
03DC 210020         LXI     H,ANPU  ;Anfangsadresse des Anzeigepuffers
03DF 06B0           MVI     B,0B0H  ;Steuertetrade (DSC1) laden
03E1 7E     AZB2:   MOV     A,M     ;Zeichentetrade : A
03E2 E60F           ANI     0FH     ;Hoeherwertige Tetrade Null setzen
03E4 80             ADD     B       ;Steuer- und Zeichentetrade addieren
03E5 D321           OUT     PA      ;Steuer-+Zeichentetrade : Anzeige
03E7 F6C0           ORI     0C0H    ;Bausteinauswahl zurueecknehmen
03E9 D321           OUT     PA      ;CS1=CS2=H
```

```
03EB 78              MOV     A,B     ;Steuer/Zeichenwort : A
03EC D610            SUI     10H     ;Steuerwort fuer die naechste Anzeige
03EE 47              MOV     B,A     ;links daneben erzeugen.
03EF 23              INX     H       ;Adresse fuer naechstes Zeichen im Puff.
03F0 7D              MOV     A,L
03F1 FE08            CPI     8       ;Kontrolle ob Pufferende (=Anzeigeende)
03F3 C2E103          JNZ     AZB2    ;Sprung wenn nicht Pufferende
03F6 F1              POP     PSW
03F7 E1              POP     H
03F8 C1              POP     B
03F9 C9        ENDE1: RET

03FA 49              MOV     C,C     ;Tabellenanfang
03FB 7F              MOV     A,A
03FC 2000            DW      32
03FE 4D4F4E49        DB      'MONI4/1-0683 '
0402 342F312D
0406 30363833
040A 20
040B 50726F66        DB      'Prof.Dr.D.Hannemann'
040F 2E44722E
0413 442E4861
0417 6E6E656D
041B 616E6E
041E 5B              MOV     E,E     ;Ende
041F 0000            DW      0       ;Startadresse

2000           ANPU  EQU     2000H   ;Anzeigepuffer Anfangsadresse
2008           INT   EQU     2008H   ;Interruptvermittelung
2100           STACK EQU     2100H   ;Stapelzeigergrundstellung
2029           ARSP  EQU     2029H   ;Anf.Adr.allgemeiner Arbeitsspeich.
03FA           SINGL EQU     ENDE1+1 ;Einsprung ins Testsystem
0020           SR    EQU     20H     ;Steuerregisteradresse des 8155
0021           PA    EQU     SR+1    ;Adresse von Port A
0022           PB    EQU     SR+2    ;Adresse von Port B
0023           PC    EQU     SR+3    ;Adresse von Port C

0000                 END
```

SYMBOL TABLE

ADLE	02D0	ADRAN	0288	ANFA	00F6	ANFA1	008E
ANLES	02FE	ANNA	0176	ANPU	2000	ANSCH	02A6
ANZ12	0257	ANZ34	025E	ANZ56	0265	ANZ78	026C
ANZEI	03D9	ARSP	2029	AZB2	03E1	BERTA	0078
DALES	02B5	EIN	01DB	ENDE	0382	ENDE1	03F9
FALS	01A9	FEHL	023E	GABI	03CE	GDRT0	029E
GDRT1	0293	HANNA	0189	HERA	01DE	HILDE	0386
HOCH	037D	INA	01CE	INITI	006F	INT	2008
KLEIN	0365	KOKE	032C	KOPIE	0350	LAUF	0345
LOLA	0274	MARTA	010E	MITA	017E	NICHT	02E3
NOCH	0156	NULL	031B	OBEN	01FA	OBER	0343
OHNE	0172	PA	0021	PB	0022	PC	0023
PORT	01D8	PORTA	0186	PORTS	01A5	PRUEF	0223
RAMAN	014E	RAMSE	0132	RAMTS	01EB	RAMVE	0117
RIKI	02E6	RUTH	02BA	SINGL	03FA	SP1	0322
SP10	0143	SPSCH	03AA	SR	0020	STACK	2100
START	0129	TASAB	039F	TASDR	0385	TASGE	03D3
WARTE	0394	ZA1MS	0341	ZESCH	03B4	ZURU	0170

5. Lösungen zu den Übungsaufgaben

Bei Aufgaben, deren Lösung hier nicht angegeben ist, kann der Leser über das Stichwortverzeichnis entsprechende Erklärungen in den einzelnen Kapiteln finden.

1. 1: HLLHHLLL + HHHLHH = HHLHLLHH

1. 2: HLLH*HLL = HLLHLL, da HLL=2^2, wird das Komma nur um zwei Stellen nach rechts verschoben.

1. 3: HHLL - HLLH = L HHLL + H LHHH(2-Komplement) = L LLHH

1. 4: HLLH / HLL = HL,LH, da HLL=2^2 wird das Komma nur um zwei Stellen nach links verschoben

1. 5:

2048	1024	512	256	128	64	32	16	8	4	2	1
2378			330		74			10		2	
-2048			-256		-64			- 8		-2	
330			74		10			2		0	
H	L	L	H	L	H	L	L	H	L	H	L

1. 6: HLLHLHHL = 128 + 16 + 442 = 150

1. 7: LHHHHLHL = LHHH HLHL = 7A

1. 8: 178D=a*16^1+b*16^0=a*16+b=11*16+2=B2H, oder 178:16=11 Rest 2

1. 9: CD = 12 * 16 + 13 = 205D

1.10: HLLH,HLHHLH = 8+1+0,5+0,125+0,0625+0,015625 = 9,703125

1.12: 2^8 = 256

1.14: a) HLLHLLLHLHHL = LLLH HLLH LHHL = 196D
b) 196D = 128+64+4 = HHLLLHLL Zur Tetradendarstellung der Zahl 196D werden 4 Bits mehr gebraucht.

1.15: 1 Byte hat 8bit.

1.16: 8-4-2-1 -Code oder Tetradendarstellung.

1.17: 2^7 = 128

1.18: 3bit

1.19: 67H

1.21: Siehe Abb.Ü1.21

2. 3: Der Adreßbus besteht aus 4 Tetraden und es sind 2^{16}= 65535 verschiedene Adressen darstellbar.

2. 5: 63H (Abb.2.2)

2. 6: 8 Zellen

2. 7: 6 allgemeine Register

2. 8: 16bit

2. 9: Entweder die um 1 erhöhte Adresse des letzten Befehlsbytes oder eine Sprungzieladresse.

2.10: Arithmetische und logische Operationen

2.11+2.12: Nach einem Reset sendet der MP 8080 die Adresse 0000 aus und holt das erste Befehlsbyte ein.

2.13: 2MHz/(5 Takte) = 400 000/s

3. 3: Adresse oder Konstante

3. 4: C3DEBA

3. 6: 05; 86; C6BA; E600; 321012

3. 7: Adreßteil eines Befehles oder ADI 78

3. 8: Vorher die Adresse von M in HL laden.

3. 9: BC,DE,HL

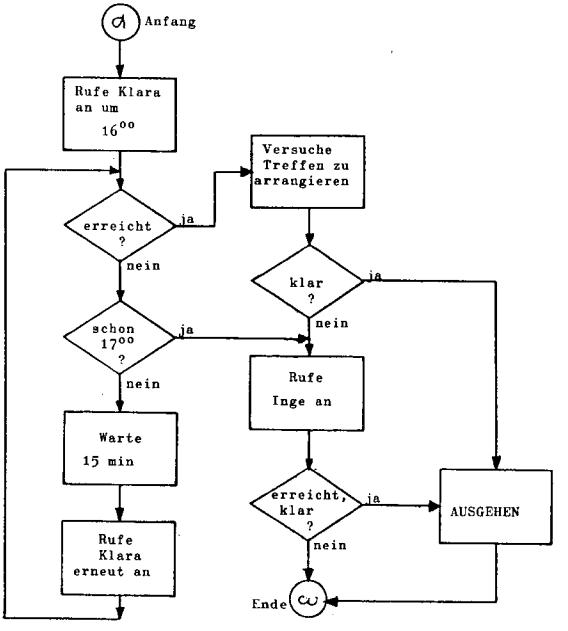


Abb. Ü 1.21: Flußdiagramm zur Lösung des Problems einer Rendevous-Vereinbahrung

3.10: In der Befehlszeile mit der Adresse B steht 8C,0,1C,L,H,H,L. Aufgabe dieses Programmes ist es 28*5 zu errechnen. Das Ergebnis steht im Akku (8C=140D). Die Multiplikation wird hier durch eine wiederholte Addition ersetzt

3.11: a) Direkte-, b) Register-, c) Implizite-, d) Unmittelbare-Adressierung.

3.12: 1. Befehlsadresse, 2. Hex- oder Objektcode, 3. Mnemonic- oder Operationscode, 4. Operanden, 5. Kommentar.

3.13: Eine Adresse, die um 1 und 2 erniedrigt, zwei freie RAM-Speicherplätze kennzeichnet.

3.14: a) 1025, b) 000A

3.15: Falls das Registerpaar HL nicht anderweitig gebraucht wird, kann der RET-Befehl durch POP H und PCHL ersetzt werden.

3.16: Das hängt nur von der Größe des Stapelspeichers ab.

3.17: Ein von außen erzwungener Unterprogrammsprung.

3.18:

Befehl:	IN	MOV	IN	ANA	OUT
Takte:	10	5	10	4	10

$39T/(2*10^6 T/s) = 19{,}5us$

4. 1: Siehe Beisp.4.8 und 4.9

4. 3: Wenn keine Taste gedrückt wird, übergibt das Dienstprogramm im Akku den Wert FF. Die Akku-Rotation ändert nichts an diesem Wert.

4. 4:

```
    MVI  A,FF ;  7T
    OUT  PORT ; 10T , PORT=H, nach Bef. Ausf.
    MVI  A,1  ;  7T                                  7T
    CALL AA   ; 18T                                 18T
    MVI  A,0  ;  7T                                  7T
    OUT  PORT ; 10T , PORT=L                        10T

AA:PUSH PSW   ; 12T                                 12T
BB:MVI  D,FF  ;  7T                                  7T
CC:DCR  D     ;  4T     : 255*4T=1020T:          1020T
   JNZ  CC    ;  7 o.10T: 254*10T+7T=2547T:       2547T
   DCR  A     ;  4T                                 4T
   JNZ  BB    ;  7 oder 10T (Bed.erfüllt)           7T
   POP  PSW   ; 10T                                 10T
   RET        ; 10T                                 10T
                                                  ------
                                                  3659T
```

Taktfrequenz f = 3MHz, Taktdauer T = 1/f.
Impulsdauer t = T * Anzahl der Takte (N) = N/f
t = 3659 / 3MHz = 1,2196ms

4. 6: Das Testprogramm läuft bis zur maximalen Adresse (FFFF), dann geht es mit der Adresse 0 weiter bis zur angegebenen Endadresse.

4. 7: Siehe nach im Monitorprogramm (Anhang 4).

5. 1: Dieser Code (Objektcode oder Maschinencode genannt) wird in einem Speicher des Mikrocomputers gebracht, für den das Programm geschrieben wurde.

5. 2: Ein Cursor ist ein bewegliches Zeichen auf einem Bildschirm (oder einem anderen Anzeigefeld), welches anzeigt, wo das nächste Zeichen erscheint.

5. 3: Ein Kommando wird erteilt, durch die Eingabe eines oder mehrerer Kommando-Zeichen (Kommando-Name), evtl. unter Hinzufügung weiterer Kommandozeichen und -Daten, und ein Kommando-Abschlußzeichen (Kommando-Ausführungszeichen).

5. 4: Ein Monitor verwaltet die Datenströme, stellt die Ein- und Ausgabe-Treiber-Routinen zur Verfügung und gestattet die Benutzung dieser Fähigkeiten über die Eingabe von Kommandos.

5. 5: Der Schalter 5 muß in die Stellung 1 gebracht werden.

```
MVI A,LLHLLLLL ; Bring 20H in den Akku
STA 2803H      ; Schalterbyte setzen
JMP ANFA       ; Zurück in den Monitor (Warmstart)
```

5. 6: Es wird das Tastaturprogramm des BIOS (Beisp.5.5) TASDR benutzt. Der Einsprung hierzu steht immer auf der Adresse "Warmstartadresse"+6. Die Warmstartadresse wiederum steht immer auf den Speicherplätzen mit der Adresse 1 und 2.

```
TASDR: LHLD 0001   ;Warmstart Einsprung-Adresse
       LXI  B,6
       DAD  B      ;(HL)=(HL)+6
       PCHL        ;Sprung auf TASDR des BIOS
```

5. 7: Ein Komma trennt die Operanden.

5. 8: MVI A,5*2 : 3E0A.

5. 9: Die Adresse PUFER hat den Wert 1003H, 1003H + 32D = 1023H. Ergebnis: 212310.

5.10: 414243 (ASCII-Code der Buchstaben A,B,C)

7. 1: Um einen direkten Speicherzugriff machen zu können (DMA).

7. 2: Prinzipiell unterschiedliche Informationen werden zeitlich versetzt auf dem selben Bus transportiert, z.B. AD-Bus des MP 8085.

7. 3: Um zwei gemultiplexte Informationen zu entkoppeln.

7. 4: Ein Signal, welches die Datenflußrichtung auf dem Datenbus und einem angeschlossenen Speicher- oder Ein-/Ausgabe-Bausteinen angibt.

7. 5: Ein Maschinenzyklus dient zur Einholung oder Ausgabe von Befehls- oder Datenbytes auf dem Datenbus und zur Abarbeitung von Befehlen.
Ein Befehlszyklus besteht beim MP 8080/85 aus 1 bis 5 Maschinenzyklen und umfaßt die Zeit von der Ausgabe der ersten Adresse bis zur vollständigen Abarbeitung des Befehles.

7. 6: Zur Einholung des Operationsteiles eines Befehles (opcode fetch)

7. 7: Indem Wartezyklen (Abb.7.8) eingefügt werden.

7. 8: T = 1 / 3MHz = 0,333us
a) T1-T3: 3*T=1us, b) T4: 0,333us, c) T5+T6: 2*T=0,666us.

7. 9: C_{ges} = C(6*2716)+C(8*2114)+C(4*8255)+C(MP-8085)+C(Leitung)
C_{ges}=(36+40+40+20+120)pF=256pF, tv=(256-150)pF*0,30ns/pF=31,8ns

7.10: 2^{20} = 1.048.576

7.11: ca. T/2

7.12: 2^{9} = 512 = 1/2K

7.13: Im 3. Maschinenzyklus wird die Portadresse 0707 ausgesandt (Abb.7.7) und IO/M hat den Pegel H (Peripheriezugriff). Daraus folgt: Ao=H, A1=L, A2=L, E1=H, E2=L, E3=H.

7.14: Ao an A10, A1 an A11, A2 an A12, E3 an A13, E1 an IO/M und A14+A15 'ODER verknüpft' auf E2.

7.15: Die ODER-Verknüpfung bewirkt, daß der Datenbustreiber im hochohmigen Zustand ist, wenn ALE oder HLDA den Zustand H hat, d.h. wenn eine Adresse ausgegeben oder der Mikroprozessor angehalten wird.

7.16: Wenn die Busse den hochohmigen Zustand annehmen, sind sie, und damit auch die Eingänge des 8205, in einem indifferenten Zustand. Der pull-up-Widerstand an E1 sorgt dafür, daß alle Ausgänge im H-Zustand bleiben.

7.17: Der Mikroprozessor kann noch andere Aufgaben erledigen.

8. 1: Nichtflüchtige (nonvolatile) und flüchtige Speicher.

8. 2: Massenspeicher.

8. 3: Halbleiterspeicher

8. 4: 2048 * 8 = 16384

8. 5: 8 Datenleitungen + 10 Adreßleitungen + Lese-/Schreibauswahl + Bausteinauswahl + Versorgungsspannung + Masse = 22

8. 6: Weil von den 64 Zellen einer Zeile immer 4 zu einem Wort zusammengefaßt werden. 64/4 = 16 = 2^{4}.

8. 7: Zum einen aus einem Flip-Flop und zum anderen aus einem Kondensator.

8. 8: Nachdem die Adresse und die Steuersignale ausgesandt worden sind, vergeht die Zugriffszeit, bis der Inhalt des Speicherplatzes auf dem Datenbus erscheint. Dieser Zustand bleibt solange erhalten, bis das READY-Signal zurückgenommen wird.

8. 9: 5101: 0,01mW(Abb.8.6); I=P/U=5uA;
t = 450mAh / 20uA = 22500h = sa.2,5 Jahre

8.10: Ein ROM kann nur vom Hersteller und ein PROM auch vom Anwender programmiert werden. Gemeinsamkeiten: nicht umprogrammierbar, permanent, wahlfreier Zugriff.

8.11: Durch die Auf- oder Entladung eines sog. schwebenden Gates.

9. 1: z.B.
```
        LXI  H,4000H
        MVI  B,100
   LES: IN   20H     ; 10 Takte
        MOV  M,A     ;  7
        INX  H       ;  6
        DCR  B       ;  4
        JNZ  LES     ; 10
                       37 Takte
```
Übertragungsrate = $6{,}144*10^6/(2*37)$s = 83027,027 Bytes/s

9. 2: Durch das Einfügen von Füllbefehlen, mit zusammen 27 Takten, verlängert sich die Schleife auf insgesamt 64 Taktzyklen.

Übertragungsrate = $6{,}144\ 10^6/(2*64)$s = 48000 Bytes/s
Füllbefehle: z.B. die Befehle CPI+IN+IN hinter dem INX-Befehl einfügen, dort stören sie den Ablauf nicht.

9. 3: Die Bausteinauswahl erfolgt immer dann, wenn IO/M = H, A13 = H A10 = A11 = A12 = A14 = A15 = L, d.h. bei einem IN- oder OUT-Befehl mit den Adressen 20, 21, 22, oder 23H. Die unterschiedlichen Adressen ergeben sich durch Variation der Adreßleitungen A8 und A9; hiermit werden die internen Register des 8255 selektiert. Gemäß Abb.9.5 hat das Steuerregister dann die Adresse 23H. Die übrigen Adressen sind: PA = 20H, PB = 21H, PC = 22H.
```
     MVI  A,10001010B  ; = 8AH
     OUT  23H
     MVI  A,55H
     CMA
     OUT  20H
     IN   21H
```

10.1: Tastenabfrage
```
TASTAB:PUSH  B
       MVI   B,03H   ;Tastenwert rechts unten
       MVI   A,10H   ;DB 4 = H (H auf erste Spalte rechts)
SPSCHL:PUSH  B       ;Tastenwert sichern
       OUT   22H     ;Spalten H (Port C) ausgeben
       PUSH  PSW     ;Port C höherwertig sichern (PCH)
       IN    22H     ;Zeilen abfragen (Port C niederwertig, PCL)
       MVI   C,04    ;Schleifenzähler= 4
ZSCHL: RRC           ;Akkubit 0,1,2,3 ins Carry schieben
       JC    ENDE    ;Sprung wenn H gefunden
       INR   B       ;B um 4 erhöhen,
       INR   B       ;     dies ist gleich
       INR   B       ;     dem Wert der
       INR   B       ;     darüber liegenden Taste.
       DCR   C       ;Schleifenzähler -1
       JNZ   ZSCHL   ;Alle 4 Zeilen abfragen
       POP   PSW     ;PCH Zustand rückladen
       POP   B       ;Tastenwert der untersten Zeile rückladen.
       DCR   B       ;Tastenwert links daneben
       ADD   A       ;Spalten H um eine Stelle nach links
       JNZ   SPSCHL  ;Nächste Spalte aktivieren
       MVI   A,0FFH  ;Kennzeichen:Keine Taste gedrückt
       POP   B
       RET
ENDE:  POP   PSW
       POP   PSW     ;Stackpointer um 4 erhöhen
       MOV   A,B     ;Tastenwert in A laden
       POP   B
       RET
       END
```
Siehe auch Beisp.11.6

10. 2: Software: Wenn ein Programm gemäß Aufg.10.1 benutzt wird, erkennt dieses nur den Tastenwert, der am weitesten rechts steht.
Hardware: Werden zwei Tasten in einer Reihe gleichzeitig gedrückt, so wird der H-Ausgang des einen Portkanales mit dem L-Ausgang des anderen kurzgeschlossen. Dies ist nicht zulässig (Ein 8255 'überlebt' dies jedoch einige Zeit). Zum Schutz hiervor, können in die Spaltenleitungen Dioden eingefügt werden.

10. 3: LLLHHLH, dies ist das Steuerzeichen für den Wagenrücklauf eines Druckers.

10. 4: 88H oder 08H, je nachdem welchen Wert PA7 haben soll.

10. 5:
```
ANFA: MVI  A,88H ; Code der Zahl A
      OUT  20H   ; Ausgabe auf Port A
      MVI  A,1   ; Anzeige 1 einschalten
      OUT  21H   ; PB0 = H
      MVI  A,0   ; Anzeige 1 ausschalten
      OUT  21H   ; PB = L
      MVI  A,0F9H; Code der Zahl 1
      OUT  20H   ; Ausgabe auf Port A
      MVI  A,2   ; Anzeige 2 einschalten
      OUT  21H   ; PB1 = H
      MVI  A,0   ; Anzeige 2 ausschalten
      OUT  21H   ; PB = L
      JMP  ANFA  ; Zurück zum Anfang
```

10. 6: LHLLLHL, dieses Ergebnis erhält man unter Berücksichtigung der Tatsache, daß die beiden linken Zeichen der Zahl 432H den ASCII-Code darstellen und damit ein C bedeuten (Abb.10.14). Die rechte Stelle gibt die Nummer der zur Darstellung des Zeichens notwendigen Zeile (hier die 3.) an. Der Abb.10.10 kann man entnehmen, daß zur Darstellung des Buchstaben C in einer 5x7 Matrix in der 2. sichtbaren Zeile zwei Punkte zu erzeugen sind.

10. 7:
```
      LXI  H,3000H  ; Anfangsadr. Videoram
      LXI  B,3600H  ; Endadresse Videoram
ANFA: MVI  M,20H    ; Leerzeichen einschreiben
      INX  H        ; Nächster Speicherplatz
      PUSH H
      DSUB          ; (HL) = (HL) - (BC)
      POP  H
      JNZ  ANFA     ; Sprung, solange die Endadresse nicht
                      erreicht wurde.
```

10. 8: Bandgeschwindigkeit v = 4,75cm/s, N = 4 Schwingungen/bit, f = 1200Hz.
X = (4/bit)*v/f = 0,1583mm/bit
C = f*60min/((4/bit)8bit/Byte) = 135kB

10. 9: Die schnellste Änderung erfolgt bei der Einfügung eines Phasenflußwechsels (Abb.10.27) mit fo = (2/bit)*6kbit/s = 12kHz.
Anzahl der Taktzyklen = 3,072MHz/12kHz = 256

10.10: n = 6/s, t = 1/n = 1s/6

6. Literaturangaben

(1) Blomeyer-Bartenstein H.P.,"Personal Computer", Verlag Markt & Technik, München 1980.

(2) Dworatschek S.,"Einführung in die Datenverarbeitung", Walter de Gruyter Verlag, Berlin.

(3) Hannemann D.,"Software-Entwicklung für 8080/8085", Markt & Technik Nr.3, S.31, München 1981;

(4) "Probleme sind die Ausbildung und Einarbeitung", die computer zeitung, S.9, 31.3.82;

(5) "Wie erlernt man den Umgang mit Mikroprozessoren", elektro anzeiger 13, S.12ff, 1982;

(6) Pszolla R.,Weßling N.,"Video-A/D-Umsetzer mit MC-Interface", Elektronik 24, S.69ff, München 1982;

(7) "Testhilfe für Mikrocomputerschaltungen", Elektronik Applikation 12, S.47ff, Essen 1982;

(8) Weidner H.,"Bildaufnahme mit Mikrocomputer und Fotodiodenzeile", Elektronik Applikation 2 + 3, 15Jg, Essen 1983;

(9) Frank J.,"Audio-Kassettenrekorder als Massenspeicher", Elektronik Applikation 7, S.42ff, Essen 1983;

(10) Haßiepen J.,"Erweiterung eines 8086-Mikrocomputers um den Arithmetikprozessor 8087", Elektronik Applik.10, Essen 1983.

(11) "Mikroelektronik-Innovation in einem mittelständischen Unternehmen des Maschinenbaus", ITZ-Schriftenreihe, Duisburg 1983;

(12) et al,"Elektronische Bauelemente und Schaltungen in der Energietechnik", VDE-Verlag, Berlin/Offenbach 1984;

(13) "Programmierung von Mikroprozessoren I, Die 8bit-Mikroprozessoren 8080, 8085, Z80, C800, Verlag W.Girardet, Essen 1984;

(14) "Programmierung von Mikroprozessoren II, 16bit-Mikroprozessoren 8086/87/88 80186 80286, Verlag W.Girardet, Essen i. V.;

(15) "Praxis der Mikrocomputertechnik", Verlag W.Girardet, Essen, in Vorbereitung.

(16) Harper C.A., "Handbook of Components for Electronics", Mc Graw-Hill, New York, 1977.

(17) Hitachi,"81 Semiconductor Data Book, Memories", 1981.

(18) Intel, Datenbücher und Application Notes.

(19) Lesea A., Zaks R., "Mikroprozessor Interface Techniken", M.+R. Nedela, Postf.1122, D-7778 Markdorf, 1979.

(20) Pol B., "Vom Umgang mit CP/M", IWT-Verlag, 1983

(21) Siemens, Datenbücher und andere Schriften

(22) Tietze U.,Schenk Ch., "Halbleiter-Schaltungstechnik", Springer-Verlag, Berlin 1980.

(23) -------, "Pocket Mikrocomputer Lexikon", SYBEX-Verlag, Düsseldorf 1983.

(24) Weber W.J.,"Mikrocomputer Grundbegriffe von A-Z", Verlag W.Girardet, Essen, 1984.

7. Abbildungsnachweis

Einige Abbildungen dieses Buches wurden dem Verfasser von verschiedenen Firmen zur Verfügung gestellt und unverändert, oder in modifizierter Form übernommen.

Cherry:	Abb. 10.4
Data Modul:	Abb. 10.10
Intel:	Abb. 7.2, 7.3, 7.7, 7.8, 8.5, 8.8, 8.22, 8.24
Philips:	Abb. 10.28
Siemens:	Abb. 10.12, 11.13
Texas Instrum.:	Abb. 10.7b, 10.8

8. Bezugsquellennachweis

Fairchild, Daimlerstr.15, 8046 Garching-Hochbrück: MP-F8-3850; Speicher, 10414, 10470.

Hitachi, Hans-Pinsel-Str.3, 8013 Haar: Speicher, 4816, 4864, 6116, 6148, 25089, 48016.

Intel, Oberrather Str.2, 4 Düsseldorf: MP-Bausteine, 8021...8080... 8755; Speicher, 2101...2185, 2308...2364, 2708...27256, 2808...2816, 5101...5115, 6300.

Intersil, Bavariaring 38, 8 München 2: 7212.

ITT, Östliche 132, 7530 Pforzheim: Personal-Computer 3030 (Z80).

Lutronik, Am Schornacker 6, 4230 Wesel: Personal-Computer, Drucker.

Mitel, Atlantik Elektronik, Hoffmannstr.20, 8 München 70: MP-MD46802.

Mostek, Freischützstr.92, 8 München 81: MP-Z80; Speicher, 4118.

Motorola, Arabellastr.17, 8 München 81: MP-6800...09; Speicher, 10155

National Semiconductor, Industriestr.10, 8080 Fürstenfeldbruck: MP-SC/MP, MP-NSC800.

NEC, Karlstr.123, 4 Düsseldorf: MP und Speicherbausteine.

PTFE, Hermann-Löns-Str.6, 465 Gelsenkirchen 2: MICO 85, Mikrocomputer auch als Bausätze und Zubehör (Abb.11.1).

RCA, Justus-von-Liebig-Ring 10, 2085 Quickborn-Hamburg: MP-CDP1802; Speicher, 5101.

Rockwell, Fraunhofer Str.11A, 8033 Martinsried: MP-6500.

Siemens, Wittelsbacherplatz 2, 8 München 2: alle Intel-Bausteine.

Texas Instruments, Haggerty Str.1, 805 Freising: MP-TMS9980; Speicher, 4164, 2516.

Triumpf Adler, Veilhofstr.6D, 85 Nürnberg: alphatronik Personal-Computer (MP-8085).

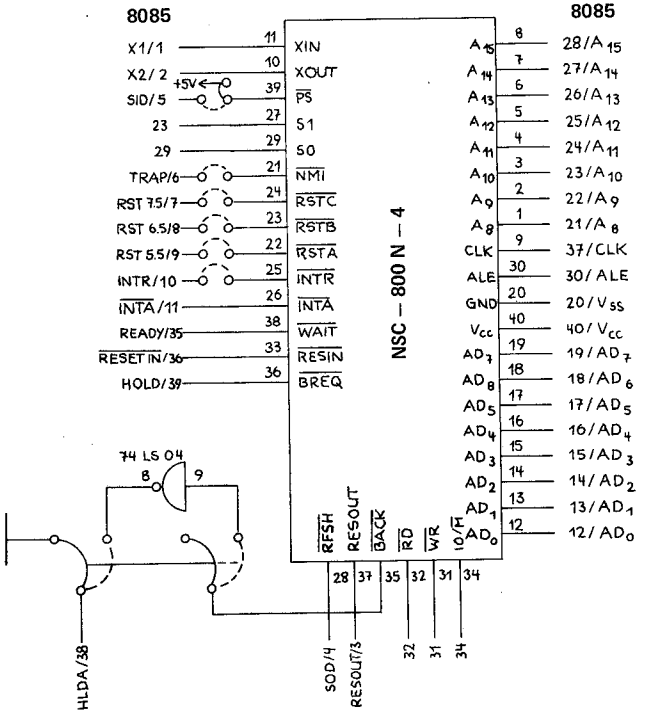
Valvo, Buchhardstr.19, 2 Hamburg 1: MP-2650; Speicher, 2616.

Zilog, Eschenstr.8, 8028 Taufkirchen: MP-Z80.

Dieser Nachweis erhebt keinen Anspruch auf Vollständigkeit, er soll lediglich dem Anfänger helfen, sich auf dem Markt der Mikrocomputerbausteine etc. etwas zurecht zu finden.

9. C800-Mikroprozessor mit Z80-Befehlssatz

Der Mikroprozessor C800 von der Firma National Semiconductor wird in CMOS-Technologie hergestellt (NSC800) und hat im wesentlichen die Busstruktur des 8085 (Intel) und den Befehlssatz des Z80 (Zilog). Der Befehlssatz des Z80 enthält alle Befehle des 8080 und viele mehr. Dies bedeutet, daß Programme geschrieben für den 8080 auch vom C800 abgearbeitet werden. Nähere Angaben hierzu enthält das Buch:"HANNEMANN, Programmierung von Mikroprozessoren I" erschienen im selben Verlag.
Wer auf den erweiterten Befehlssatz des Z80 zurückgreifen möchte, oder wen die besonderen Möglichkeiten eines CMOS-Mikroprozessors interessieren, der kann z.B. den im Kapitel 11 beschriebenen MICO 85 (Abb.11.2) mit einem C800 Mikroprozessor ausrüsten. Hierzu braucht im Minimum nur die in der Abb. gezeigte andere Belegung des C800 berücksichtigt, und der PS-Eingang (C800) mit +5V verbunden zu werden, sowie der HLDA-Ausgang (8085) Massepotential zu bekommen.
Zu beachten ist hierbei jedoch, daß in diesem Fall kein DMA-Zugriff möglich ist (wegen HLDA=L) und die Interrupts - anders als beim 8085 - auf ein L-Signal reagieren.
Die Bezeichnungen und Pinnummern am Baustein entsprechen denen des C800 und die Bezeichnungen außerhalb denen des 8085.
Die Befehle SIM und RIM des 8085 können nicht benutzt werden da diesem Code beim Z80 eine andere Bedeutung zukommt. Sie müssen deshalb im Monitorprog.durch ein NOP ersetzt werden.



Stichwortregister

Diese „Einführung in die Mikrocomputer-Technik" ist in erster Linie als Hilfsmittel für Studierende der Hochschulen und für Teilnehmer in Weiterbildungseinrichtungen gedacht. Sie verbindet die Vorzüge eines Skriptes mit denen eines Lehrbuchs.

Das Werk ist auf der Grundlage einschlägiger Unterrichtserfahrungen entstanden und entsprechend didaktisch aufbereitet, stellt aber keine spezialisierte Ausarbeitung einer einzelnen Lehrveranstaltung dar. Vielmehr vermittelt es – bei sinnvoller Beschränkung des Umfanges – den Standardstoff, der im allgemeinen zu einer Einführung in das Thema Mikrocomputer-Technik gezählt wird.

Die nebenstehende Inhaltsübersicht zeigt auf, inwieweit dabei Hardware-, Software- und Anwendungsfragen berücksichtigt und miteinander verknüpft sind.

PROGRAMMIERUNG
Grundlagen, Grundeinheiten eines Mikrocomputers, Programmerstellung und Abarbeitung, Personal-Computer, Betriebssysteme, Programmbeispiele

SCHALTUNGSTECHNIK
Integrierte Schaltungen, Mikroprozessor, Speicher, Ein-/Ausgabe-Einheiten, Peripherie-Geräte

ANWENDUNG
MP 8085-Mikrocomputer-System, Selbstbau-Computer, Kassettenspeicher, EPROM-Programmiergerät, Motorsteuerung

Mit Übungsaufgaben und einem Lösungsteil, zusammenfassenden Übersichten über die Befehle des MP 8080/85 und einem ausführlichen Stichwortregister

Girardet 10246