

ITT MP- Experimenter

Bedienungsanleitung

Copyright 1976 by
Standard Elektrik Lorenz Aktiengesellschaft
Unternehmensgruppe Rundfunk Fernsehen Phono
7530 Pforzheim, Östliche 132
Postfach 1570, Telefon (07231) 59-2391
1., 2. und 3. Auflage, Juli 1977

Druck: Druckerei Seiter, 7535 Königsbach-Stein

ITT MP-Experimentier

	Inhalt	Seite
1.	Funktionsbeschreibung	1
2.	Technische Daten	2
3.	Bedienungsanleitung	3
3.1	Allgemeine Hinweise zu den Experimenten	3
3.2	Addierer/Subtrahierer (SYSTEM 0)	4
3.3	Codierte ALU (SYSTEM 1)	5
3.4	Akkumulator (SYSTEM 2)	7
3.5	Akkumulator mit Speicher (SYSTEM 3)	8
3.6	Vereinfachter Rechner (SYSTEM 4)	10
3.7	Hypothetischer Mikrorechner (SYSTEM 5)	12
3.8	Mikrorechner-System 8080 (SYSTEM 6)	19
3.9	Erweitertes 8080-System (SYSTEM 7)	24
4.	Stromlaufpläne und Anschlußbelegung	25

1. Funktionsbeschreibung

Der MP-Experimenter ist ein Mikrocomputer basierend auf dem MP-System 8080.

Er ist ein Lehrsystem zur praktischen Einführung in den Gebrauch und die Arbeitsweise von Mikrocomputern. Der MP-Experimenter besteht aus der Stromversorgung, der Prozessorplatine und dem Frontpanel mit diversen Schablonen und Code-Karten.

Die Stromversorgung benötigt zum Betrieb lediglich eine Steckdose des 220-V-Netzes.

Der separat gekapselte Netztransformator gibt zum Zwecke einer Zweiweggleichrichtung $2 \times 8,5 V_{\text{eff}}$ ab. Die an ihm über eine 3polige Steckverbindung angeschlossene Netzteilplatine enthält 3 geregelte Stromversorgungen für die Betriebsspannungen +5 V, +12 V und -5 V gegen Masse. Je 2 Dioden eines Brückengleichrichters arbeiten als Zweiweggleichrichter für den +5-V- bzw. -5-V-Regler. Der zweite Brückengleichrichter arbeitet als Zweiweg-Villard-Gleichrichter für den +12-V-Regler. Die Ausgänge aller Regler sind kurzschlußfest.

Der Mikrocomputer enthält als CPU den Mikroprozessor 8080 A und die Ergänzungsbausteine 8224 (Taktgenerator) und 8228 (System-Controller und Datenbus-Treiber). Ein Quarz von 8,867 MHz erzeugt eine Taktfrequenz von ca. 1 MHz, so daß der Prozessor mit einer Zykluszeit von ca. 1 μ s arbeitet.

Das eingebaute Memory besteht aus einem maskenprogrammierten ROM (8308) von 1 k x 8 bit, welches das Systembetriebsprogramm (Monitor) enthält und einem statischen RAM (2 x 8111) von 1/4 k x 8 bit als STACK, Daten- und Programmspeicher für Anwenderprogramme.

Dem ROM ist der Adreßbereich von 0 0 0 0₁₆ bis 0 3 F F₁₆ und dem RAM der Adreßbereich von 0 4 0 0₁₆ bis 0 4 F F₁₆ durch den Adreßdecoder (2/6 74 LS 04, 6/6 74 LS 05 und 3/4 74 LS 32) zugeordnet. Der verbleibende Adreßbereich von 0 5 0 0₁₆ bis F F F F₁₆ (also 62 3/4 k = 64 256 Adressen) steht ohne Einschränkung für Erweiterungen zur Verfügung.

Weiterhin enthält die Computerplatine insgesamt 5 8-bit-Parallel-I/O-Bausteine (5 x 8212), von denen 3 als Input-Ports und 2 als Output-Ports geschaltet sind. Sie sind in der „isolierten I/O-Adressierung“ mit den Adressen 0 0 0 1₁₆, 0 0 0 2₁₆ und 0 0 0 4₁₆ für die Input-Ports und 0 0 0 1₁₆ und 0 0 0 2₁₆ für die Output-Ports adressiert.

Da keine vollständige Adreßcodierung angewendet wurde, sondern Adreß-bit-Adressierung, stehen für weitere I/O-Adressen nur noch 0 0 0 8₁₆, 0 0 1 0₁₆, 0 0 2 0₁₆, 0 0 4 0₁₆ usw. für Inputs und 0 0 0 4₁₆, 0 0 0 8₁₆, 0 0 1 0₁₆ usw. für Outputs zur freien Verfügung.

Am System-RESET (RESET-Eingang des 8224) liegt eine RCD-Kombination zum automatischen RESET beim Einschalten der Betriebsspannung.

Die System-Steuersignale HOLD, INT und BUS EN sind über Inverter geführt, die am Eingang einen 1-k Ω -Pull-up-Widerstand tragen. Als äußere Eingangssignale stehen sie somit als active-low-Signale ($\overline{\text{HOLD}}$, $\overline{\text{INT}}$, BUS EN) zur Verfügung, d.h. ein äußeres Low-Signal an $\overline{\text{HOLD}}$ steuert den 8080 in den HOLD-Zustand, ein Low-Signal an $\overline{\text{INT}}$ gibt an den 8080 eine Unterbrechungs-Anforderung (Interrupt-Request) und ein Low-Signal an BUS EN steuert die Datenbus-Ausgänge sowie die Steuerbus-Ausgänge vom System-Controller (8228) in den Tri-State-Zustand. Als weitere Eingänge sind noch $\overline{\text{RES IN}}$ und RDY IN vorhanden. Ein äußeres Low-Signal an $\overline{\text{RES IN}}$ bewirkt einen RESET, ein Low-Signal an RDY IN bringt den Prozessor in den Warte-Zustand.

An Steuersignal-Ausgängen stehen RESET (pos. Impuls) zum Rücksetzen externer Elemente, $\overline{\text{STSTB}}$ (neg. Impuls) zum Abfragen der Statusinformation, WAIT (pos. Impuls) zur Anzeige des Warte-Zustandes, HLDA (pos. Impuls) als Quittierung einer HOLD-Anforderung und INTE, das anzeigt, ob ein Interrupt ein- (INTE = H) oder ausgeschaltet (INTE = L) ist, zur Verfügung.

Herausgeführt sind weiterhin die 5 Signale des Steuerbusses $\overline{\text{MEM R}}$ (neg. Impuls für Speicher Lesen), $\overline{\text{MEM W}}$ (dto. für Speicher Schreiben), $\overline{\text{I/O R}}$ (dto. Eingangs-/Ausgangs-Port Lesen), $\overline{\text{I/O W}}$ (dto. Eingangs-/Ausgangs-Port Schreiben) und $\overline{\text{INTA}}$ (Interrupt-Acknowledge = Interrupt-Bestätigung), der wegen fester Verbindung über 1 k Ω an +12 V einen automatischen RST 7-Befehl erzeugt.

Die Prozessor-Platine ist – von oben verdeckt – unter die Oberplatine (Front-Panel) gesteckt.

Das Front-Panel trägt einmal die Steckleisten zum Anschluß der Stromversorgung, zum Anschluß anderer Ein-/Ausgaben und für Systemerweiterungen und zum anderen die diversen Eingabe-Schalter und LED-Anzeigen zur Ausgabe.

Der mit „SYSTEM“ gekennzeichnete Codierschalter mit den Zahlen 0 bis 9 dient zur Auswahl der entsprechenden Monitorprogrammteile, die die einzelnen Experimentierschritte simulieren. Die Stellungen 0 bis 6 sind den Systemen „Addierer/Subtrahierer“ (0), „Codierte ALU“ (1), „Akkumulator“ (2), „Akku mit Speicher“ (3), „Vereinfachter Rechner“ (4), „Hypothetischer Rechner“ (5) und „Prozessorsystem 8080“ (6) zugeordnet. Die Stellung 7 dient für Erweiterungen mit zusätzlichen Betriebsprogrammen, während die Stellungen 8 und 9 unbenutzt bleiben.

Die binären Schiebeschalter C_4 bis C_0 dienen als Funktionsschalter unterschiedlicher Bedeutung in den verschiedenen Systemen. Auch die mit A_7 bis A_0 bzw. B_7 bis B_0 gekennzeichneten binären Schiebeschalter haben in den verschiedenen Systemen unterschiedliche funktionelle Bedeutung. Sie dienen in erster Linie zur Eingabe von Daten oder Adressen.

Als Ausgabe- bzw. Anzeigeelemente sind 2×8 LEDs mit der Bezeichnung L_7 bis L_0 und R_7 bis R_0 angeordnet. Sie gestatten das Ablesen von Adressen oder Daten im Maschinencode (Binärcode, d.h. leuchtende LED = log. 1, nichtleuchtende LED = log. 0). Ihre funktionelle Bedeutung ist unterschiedlich und wird von den Schablonen festgelegt.

Die mit „RESET“ bezeichnete Taste bringt den Prozessor zum Programmstart, d.h. nach deren Betätigung beginnt der Prozessor das Monitorprogramm ab Adresse $0\ 0\ 0\ 0_{16}$ abzuarbeiten. Die RESET-Taste hat keine „Clear-Funktion“, d.h. es werden keine Register- oder Speicherinhalte gelöscht! Sie ist jeweils nach der Veränderung des Systemwahlschalters zu betätigen, damit ein eindeutiges Einlaufen in das gewählte Systemprogramm gewährleistet ist.

Die farblich gekennzeichneten und mit der System-Nr. versehenen Schablonen weisen den Bedienungselementen der Frontplatte ihre systemspezifische Funktion zu. Die jeweils in der gleichen Farbe vorhandenen Codekarten (DIN A 6) bzw. Befehlslisten (195×175 mm) dienen zur Kurzinformation über Funktions- bzw. Displaycodes (Anzeigecodes).

2. Technische Daten

MP-System	8080 A
mit Taktgenerator	8224 (SN 74LS 424)
und System-Controller	8228 (SN 74LS 428)
Quarzfrequenz	8,867 MHz
Zykluszeit	ca. 1 μ s
Memory	ROM 1 k x 8 bit (Typ 8308) mit System-Betriebsprogramm RAM 1/4 k x 8 bit (2 x Typ 8111) für Anwenderprogramme
Memory-Erweiterung	möglich ab Adresse $0\ 5\ 0\ 0_{16}$ bis Adresse $F\ F\ F\ F_{16}$
Input	3 x 8 bit parallel mit 3 x I/O-Port 8212 (SN 74LS 412) für C-, B- und A-Schalter, bit-weise I/O-Adresse mit Adreß-bit 0, 1 bzw. 2
Input-Erweiterung	möglich mit den Adreß-bits 3, 4, 5, 6 und 7
Output	2 x 8 bit parallel mit 2 x I/O-Port 8212 für L- und R-LEDs, bit-weise I/O-Adresse mit Adreß-bit 0 und 1
Output-Erweiterung	möglich mit den Adreß-bits 2, 3, 4, 5, 6 und 7

fan-out an der Systemleiste links (1 LE = $1 \times \text{TTL} \triangleq 1,6 \text{ mA}_L$, $40 \mu\text{A}_H$)

Adreßbus	$AB_0 \dots AB_{15}$	1	LE
Datenbus	$DB_0 \dots DB_7$ (bidirektional)	5	LE
Steuerbus	MEM R, MEM W, I/O R, I/O W, INTA	5	LE
HLDA	(Halt-Quittung)	1	LE
INTE	(Unterbrechungs-Freigabe)	1 1/4	LE
RESET	(Rückstellung)	1 1/4	LE
$\overline{\text{STSTB}}$	(Zustandsübernahme)	1	LE
WAIT	(Wartezustandsquittung)	1 1/4	LE

fan-in an der Systemleiste links

Datenbus	DB ₀ . . . DB ₇ (bidirektional)	1/4 LE
BUS EN	(BUS-Freigabe-Steuerung)	3 1/4 LE
HOLD	(HALT-Anforderung)	3 1/2 LE
INT	(Unterbrechungs-Anforderung)	3 1/2 LE
RDY IN	(Bereit-Eingang)	3 1/2 LE
RES IN	(Rücksetz-Eingang)	1/4 LE (+ 1 µF)

fan-out an der Systemleiste rechts (in TTL-LE)

L ₇ . . . L ₀	(I/O-Port mit Adr. 0 0 0 2 ₁₆)	10	LE
R ₇ . . . R ₀	(I/O-Port mit Adr. 0 0 0 1 ₁₆)	10	LE

(U_{OH} = 3,2 V, da LED angeschlossen)

fan-in an der Systemleiste rechts (in TTL-LE)

C ₇ . . . C ₀	(I/O-Port mit Adr. 0 0 0 4 ₁₆)	1	LE
B ₇ . . . B ₀	(I/O-Port mit Adr. 0 0 0 1 ₁₆)	1	LE
A ₇ . . . A ₀	(I/O-Port mit Adr. 0 0 0 2 ₁₆)	1	LE

Als Eingänge nur verwendbar, wenn C₄ . . . C₀, B₇ . . . B₀ und A₇ . . . A₀ in Stellung „1“ und Systemschalter in Stellung „0“ oder „8“ stehen (wired OR)!

(U_{IHmax} = 5,0 V, U_{ILmin} = -0,5 V)

Stromversorgung

Kompaktnetztransformator	220 V/2 x 8,5 V, EI 66
Regelnetzteil	+5 V, 1,5 A
(kurzschlußfest)	+12 V, 250 mA
	-5 V, 100 mA
Strombedarf des MP-Experimenters	+5 V, 1 A typ.
	+12 V, 100 mA typ.
	-5 V, 2 mA typ.
Stromreserve für Erweiterungen	+5 V, 500 mA _{max}
	+12 V, 150 mA _{max}
	-5 V, 100 mA _{max}

3. Bedienungsanleitung**3.1 Allgemeine Hinweise zu den Experimenten**

Das Experimentiersystem enthält einen vollständigen Rechner. Im ROM sind 7 Programme zur Simulation von verschiedenen Systemen fest abgespeichert. Welches Programm ablaufen soll, kann mit dem SYSTEM-Schalter (BCD-Schalter auf der linken Seite) festgelegt werden. Den 7 Programmen sind die Nummern 0 bis 6 zugeordnet. Die Stellung 7 des SYSTEM-Schalters ist für eine eventuelle Erweiterung des Systems vorgesehen, die Stellungen 8 und 9 werden nicht verwendet.

SYSTEM 0:	8-bit-Parallel-Addierer/Subtrahierer
SYSTEM 1:	Codierte 8-bit-ALU
SYSTEM 2:	8-bit-Akkumulator
SYSTEM 3:	8-bit-Akku mit 16 x 8-bit-Datenspeicher
SYSTEM 4:	Vereinfachter 8-bit-Ein-Adreßrechner
SYSTEM 5:	Hypothetischer 8-bit-Mikrorechner (didaktischer Modell-Rechner)
SYSTEM 6:	8-bit-Mikrorechner mit 8080-Mikroprozessor
SYSTEM 7:	für andere Mikrorechner mit dem 8080 und Erweiterungen

Ein Programm wird mit der RESET-Taste gestartet. Diese Taste entspricht in etwa der Lösch Taste eines Taschenrechners und **muß am Anfang jedes Experimentes gedrückt werden**. Mit den restlichen Schiebeschaltern können Daten und Steuerinformationen eingegeben werden.

Die Schaltergruppe C_4 bis C_0 wird zur Steuerung des Experimentierablaufes benötigt. Die Schaltergruppen A_7 bis A_0 und B_7 bis B_0 werden bis auf einige Spezialfälle für die Daten- oder Programmeingabe benutzt.

Bei allen Experimenten gelten folgende Festlegungen:

Schalter oben = logisch 1
Schalter unten = logisch 0

Die 2 mal 8 Leuchtdioden dienen zur Anzeige der Rechnerergebnisse sowie zur Anzeige interner Schaltzustände. Hier gelten folgende Festlegungen:

Leuchtdiode leuchtet = logisch 1
Leuchtdiode dunkel = logisch 0

Alle Schalter, die bei einem bestimmten Experimentiervorgang nicht benötigt werden, sollten auf log. 0 geschaltet werden.

Es ist zu empfehlen, daß zu Beginn eines Experimentes alle Schalter auf log. 0 geschaltet werden, bevor die RESET-Taste gedrückt wird. Ausnahmen hiervon werden bei den einzelnen Experimenten angegeben.

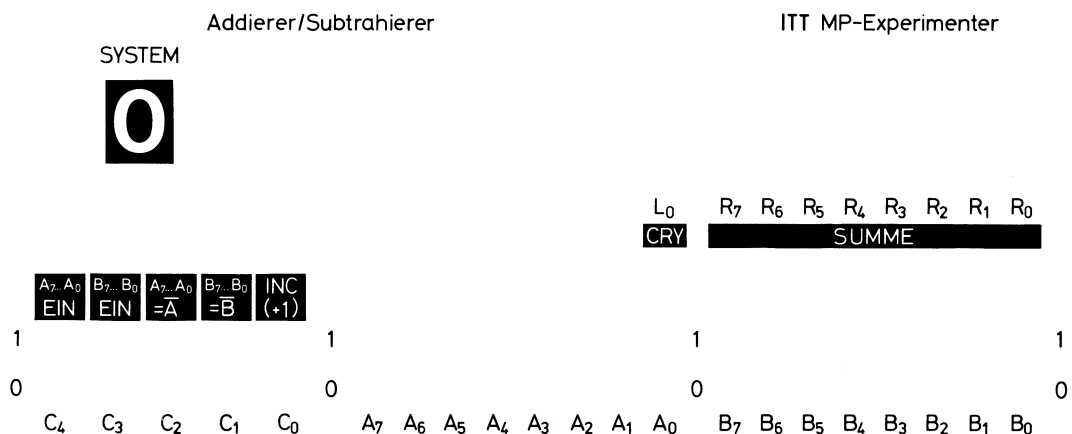
Die grundsätzliche Experimentiervorbereitung ist folgende:

1. Die in der Experimentieranweisung angegebene Schablone auflegen
2. SYSTEM-Schalter auf das verlangte Programm einstellen
3. Alle Schiebeschalter auf Null (unten) stellen
4. RESET-Taste drücken

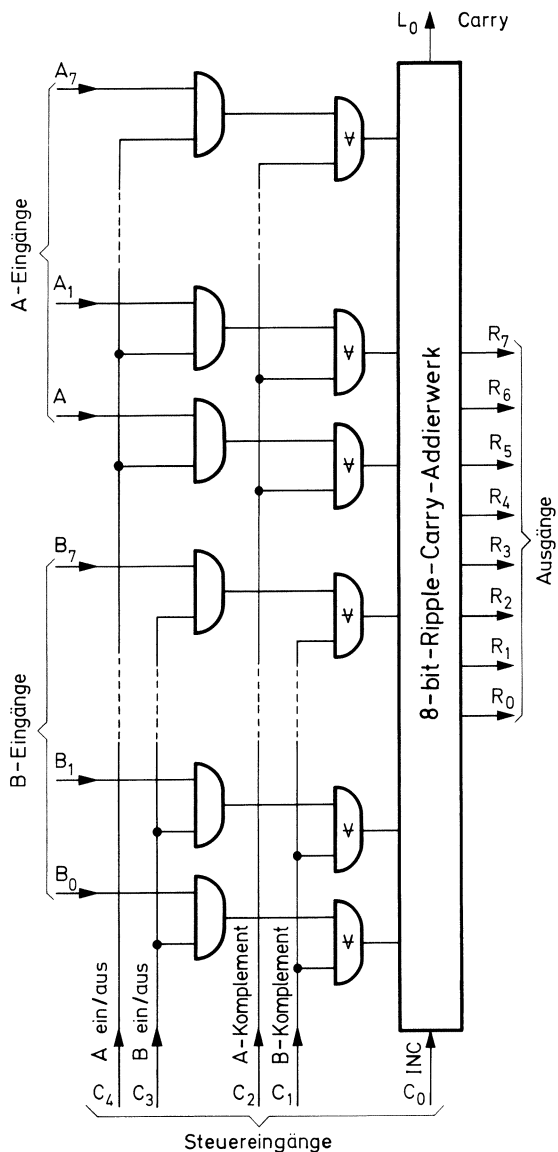
Bei falscher Schalterbetätigung können grundsätzlich keine Schäden am Experimentiersystem entstehen. Allerdings können dadurch die selbst eingegebenen Programme und Daten verändert werden, so daß ein falsches Ergebnis entsteht. Bei umfangreichen und komplizierten Experimenten kann eine falsche Betätigung viel Zeit kosten.

3.2 Addierer/Subtrahierer (SYSTEM 0)

Im System 0 läuft auf dem Rechner ein Programm, das ein 8-bit-Parallel-Addier-/Subtrahierwerk simuliert. Eine solche Schaltung ist schaltungstechnisch ein rein kombinatorisches Netzwerk mit statischer Betriebsweise (Bild). Es hat zweimal 8 Dateneingänge (A_7 bis A_0 = Operand A und B_7 bis B_0 = Operand B) und 5 Steuereingänge (C_4 bis C_0), die die in der Tabelle aufgeführten Funktionsmöglichkeiten ergeben.



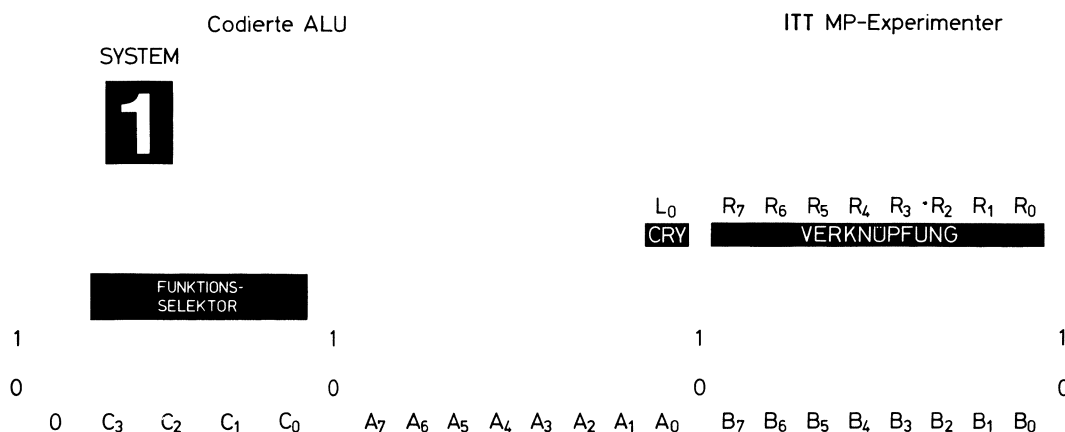
Nach der Experimentiervorbereitung Schalter C_4 und C_3 auf 1. Damit können die an den Schaltern A_7 bis A_0 und B_7 bis B_0 eingestellten Informationen in das System gelangen. Mit den Schaltern $C_2 = \bar{A}$ und $C_1 = \bar{B}$ können die eingegebenen Informationen komplementiert werden (Einerkomplement). Der Schalter $C_0 = \text{INC}$ legt bei 1 eine 1 auf den INC-Eingang. Das Ergebnis der Addition erscheint in den rechten 8 LEDs (R_7 bis R_0). Die LED L_0 in der linken Lampengruppe zeigt einen Übertrag (Carry) an. Bei diesem System haben die LEDs L_7 bis L_1 keine Bedeutung.



C ₄	C ₃	C ₂	C ₁	C ₀	Ausgangs- funktion
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	-1
0	0	0	1	1	0
0	0	1	0	0	-1
0	0	1	0	1	0
0	0	1	1	0	-2
0	0	1	1	1	-1
0	1	0	0	0	B
0	1	0	0	1	B + 1
0	1	0	1	0	-B - 1 = $\overline{B}$
0	1	0	1	1	-B
0	1	1	0	0	B - 1
0	1	1	0	1	B
0	1	1	1	0	-B - 2
0	1	1	1	1	-B - 1 = $\overline{B}$
1	0	0	0	0	A
1	0	0	0	1	A + 1
1	0	0	1	0	A - 1
1	0	0	1	1	A
1	0	1	0	0	-A - 1 = $\overline{A}$
1	0	1	0	1	-A
1	0	1	1	0	-A - 2
1	0	1	1	1	-A - 1 = $\overline{A}$
1	1	0	0	0	A + B
1	1	0	0	1	A + B + 1
1	1	0	1	0	A - B - 1
1	1	0	1	1	A - B
1	1	1	0	0	B - A - 1
1	1	1	0	1	B - A
1	1	1	1	0	-A - B - 2
1	1	1	1	1	-A - B - 1

3.3 Codierte ALU (SYSTEM 1)

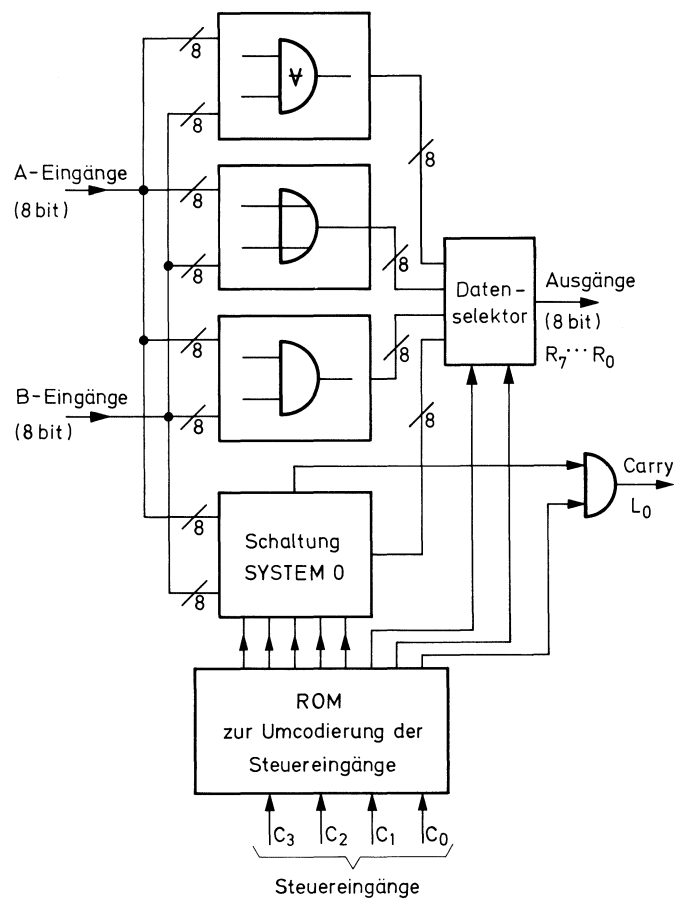
Im System 1 wird vom Rechner eine 8-bit-ALU simuliert. Sie ist schaltungstechnisch durch Erweiterung des Addierwerkes (System 0) um log. Funktionen, einen Datenselektor und eine Umcodierungsschaltung (ROM) entstanden.



Die Funktionen werden mit den Schaltern C_3 bis C_0 festgelegt. Diese Tabelle ist auch auf der Karte „Codierte ALU“ zu finden.

Die Funktionen sind alle ausreichend bekannt und bedürfen daher keiner weiteren Erläuterung.

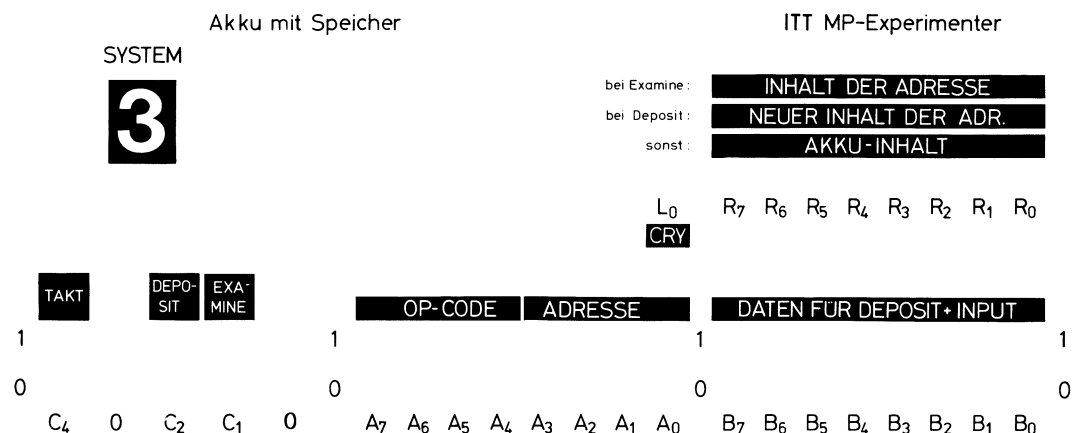
C_3	C_2	C_1	C_0	Funktion
0	0	0	0	A
0	0	0	1	1
0	0	1	0	$\bar{A}$
0	0	1	1	B
0	1	0	0	0
0	1	0	1	$A + 1$
0	1	1	0	$A - 1$
0	1	1	1	$A + B$
1	0	0	0	$A - B$
1	0	0	1	$A \wedge B$
1	0	1	0	$A \vee B$
1	0	1	1	$A \nabla B$
1	1	0	0	-1
1	1	0	1	} für Ausbau
1	1	1	0	
1	1	1	1	



Mit diesem Programm wird ein Akkumulator simuliert. Die in der Tabelle gezeigten Funktionen werden mit den Schaltern C₃ bis C₀ ausgewählt. Der Schalter C₄ dient als Taktschalter. Durch einmaliges Hin- und Herschieben wird ein Ergebnis in das Register übernommen und zur Anzeige gebracht. Die A-Schalter werden in diesem Beispiel nicht gebraucht, weil die A-Eingänge der im Akkumulator enthaltenen ALU mit den Ausgängen des Registers verbunden sind. Das Ergebnis bzw. der momentane Inhalt des Akkus wird wieder in R₇ bis R₀ angezeigt, ein Übertrag in L₀.

U ₃	U ₂	U ₁	U ₀	Abkürzung	Funktion	Übertrags-Flag
0	0	0	0	NOP	Keine Operation	ja
0	0	0	1	SP1	Setze Akku = 1	ja
0	0	1	0	CMA	Komplementiere Akku	nein
0	0	1	1	LDA	Lade B in den Akku	nein
0	1	0	0	CLA	Lösche Akku	nein
0	1	0	1	INC	Incrementiere Akku	ja
0	1	1	0	DEC	Decrementiere Akku	ja
0	1	1	1	ADD	Addiere B in den Akku	ja
1	0	0	0	SUB	Subtrahiere B von Akku	ja
1	0	0	1	AND	Akku UND B in den Akku	ja
1	0	1	0	IOR	Akku ODER B in den Akku	ja
1	0	1	1	XOR	Akku EXCLUSIV-ODER in den Akku	ja
1	1	0	0	SM1	Setze Akku = -1	ja
1	1	0	1	–	–	nein
1	1	1	0	–	–	nein
1	1	1	1	–	–	nein

3.5 Akkumulator mit Speicher (SYSTEM 3)

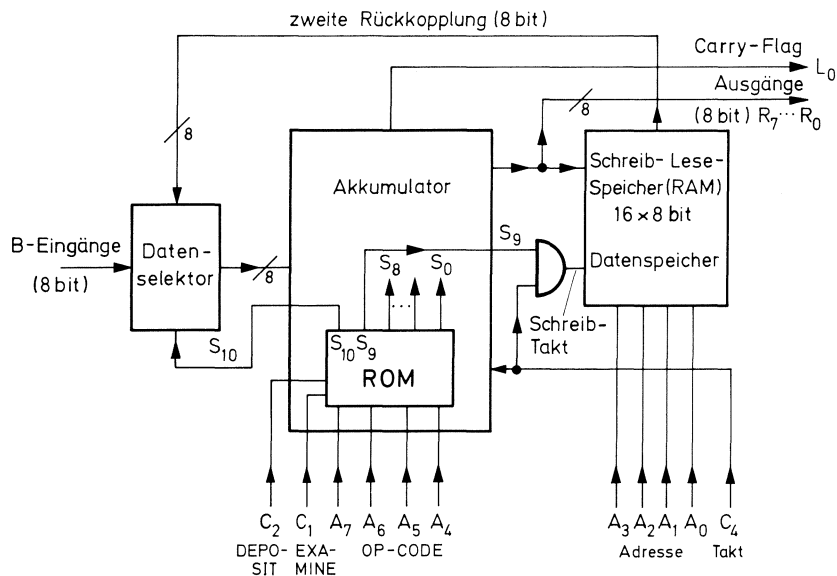


Das System 3 enthält grundsätzlich die gleichen Funktionen wie das System 2. Der Unterschied besteht darin, daß entsprechend dem Bild die Daten nicht mehr von den B-Schaltern kommen sondern von einem RAM. Mit dem neuen Befehl STA (Speichere Akku-Inhalt in Adresse a a a a b), können die Daten in das RAM zurückgeschrieben werden. Mit dem Befehl INP (Lade B-Eingänge in den Akku) werden jetzt die Daten an B₇ bis B₀ in den Akkumulator eingelesen (entspricht Befehl LDA in System 2). Bei allen Befehlen, die den Speicher nutzen, muß jetzt eine bestimmte Adresse spezifiziert werden. Der hier verwendete Speicher hat eine Kapazität von 16 Wörtern à 8 bit. Damit jedes dieser 16 Wörter spezifiziert bzw. adressiert werden kann, werden 4 bit benötigt. Damit besteht ein Befehl jetzt aus insgesamt 8 bit. Hiervon legen 4 bit die Funktion fest, die ausgeführt werden soll. Sie bilden den sog. OP-Code (Operation-Code). Die anderen 4 bit bestimmen die Speicheradresse. Aus diesem Grunde werden jetzt die A-Schalter für die Befehlseingabe benutzt.

Aus der Tabelle geht hervor, daß es Befehle gibt, die unbedingt die Angabe einer Adresse benötigen (a a a a), und andere, die ohne spezielle Adresse auskommen (x x x x).

Bevor Befehle, die Daten aus dem Speicher unter einer bestimmten Adresse benötigen, benutzt werden können, müssen die entsprechenden Daten in den Speicher geladen werden. Das Laden einer bestimmten Speicheradresse erfolgt mit dem Schalter C₂ DEPOSIT (Laden). Wird dieser Schalter betätigt, d.h. auf 1 und dann wieder auf 0 geschaltet, werden die Daten, die an B₇ bis B₀ liegen, im Speicher bei der Adresse abgespeichert, die von den Schaltern A₃ bis A₀ spezifiziert ist.

Mit dem Schalter C₁ EXAMINE (Abfragen) kann der Speicherinhalt, der mit A₃ bis A₀ spezifizierten Adresse in R₇ bis R₀ abgebildet werden.

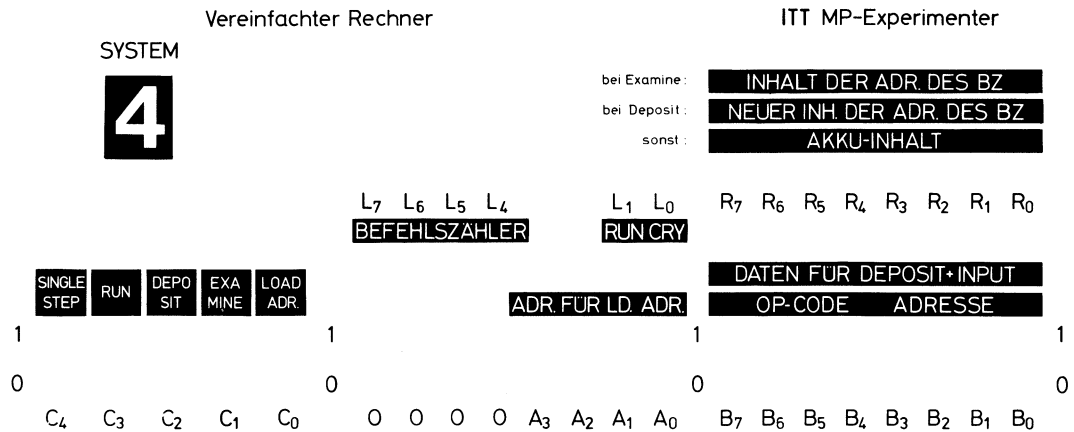


U ₃	U ₂	U ₁	U ₀	a ₃	a ₂	a ₁	a ₀	Abkürzung	Funktion	Übertrags-Flag
0	0	0	0	x	x	x	x	NOP	Keine Operation	ja
0	0	0	1	x	x	x	x	SP1	Setze Akku = 1	ja
0	0	1	0	x	x	x	x	CMA	Komplementiere Akku	nein
0	0	1	1	a	a	a	a	LDA	Lade Inhalt Adresse a a a a	nein
0	1	0	0	x	x	x	x	CLA	Lösche Akku	nein
0	1	0	1	x	x	x	x	INC	Incrementiere Akku	ja
0	1	1	0	x	x	x	x	DEC	Decrementiere Akku	ja
0	1	1	1	a	a	a	a	ADD	Addiere Inhalt Adresse a a a a	ja
1	0	0	0	a	a	a	a	SUB	Subtrahiere Inhalt Adresse a a a a	ja
1	0	0	1	a	a	a	a	AND	Akku UND Inhalt Adresse a a a a	ja
1	0	1	0	a	a	a	a	IOR	Akku ODER Inhalt Adresse a a a a	ja
1	0	1	1	a	a	a	a	XOR	Akku EXCLUSIV- ODER Adresse a a a a	ja
1	1	0	0	x	x	x	x	SM1	Setze Akku = -1	ja
1	1	0	1	x	x	x	x	INP	Lade B-Eingänge in den Akku	nein
1	1	1	0	a	a	a	a	STA	Speichere Akku in Adresse a a a a	nein
1	1	1	1	x	x	x	x	—	—	—

a a a a = eine Datenspeicheradresse

x x x x = „don't care“-Zustand, d.h. beliebig

3.6 Vereinfachter Rechner (SYSTEM 4)



Bei diesem System wird ein vereinfachter, aber kompletter Rechner simuliert. Er hat denselben Befehlsvorrat wie der Akkumulator mit Datenspeicher im System 3. Zusätzlich hat er einen HALT-Befehl (HLT), damit der Rechner am Ende eines Programms angehalten werden kann. Im Gegensatz zum System 3 werden im 16-Wort-Speicher nicht nur Daten sondern auch das Programm angespeichert. Das Programm und die Daten werden mit dem DEPOSIT-Schalter C_2 in den Speicher geladen. Damit ein Programm automatisch ablaufen kann, enthält der simulierte Rechner einen Befehlszähler (BZ). Welche der 16 Adressen gerade selektiert ist, wird durch die LEDs L_7 bis L_4 angezeigt. Die Funktionsweise des Befehlszählers können Sie wie folgt kontrollieren:

- Stellen Sie alle Schalter außer C_4 auf 0. Bei $C_4 = 1$ arbeitet das System im Single-Step-Betrieb, d.h., der Befehlszähler kann mit Schalter C_3 (RUN) in Einzelschritten getaktet werden.
- In L_7 bis L_4 erscheint jetzt eine beliebige Adresse von 0 0 0 0 bis 1 1 1 1.
- Takten Sie das System mit RUN. An L_7 bis L_4 können Sie sehen, daß der Befehlszähler mit jedem Takt um einen Schritt höher springt.

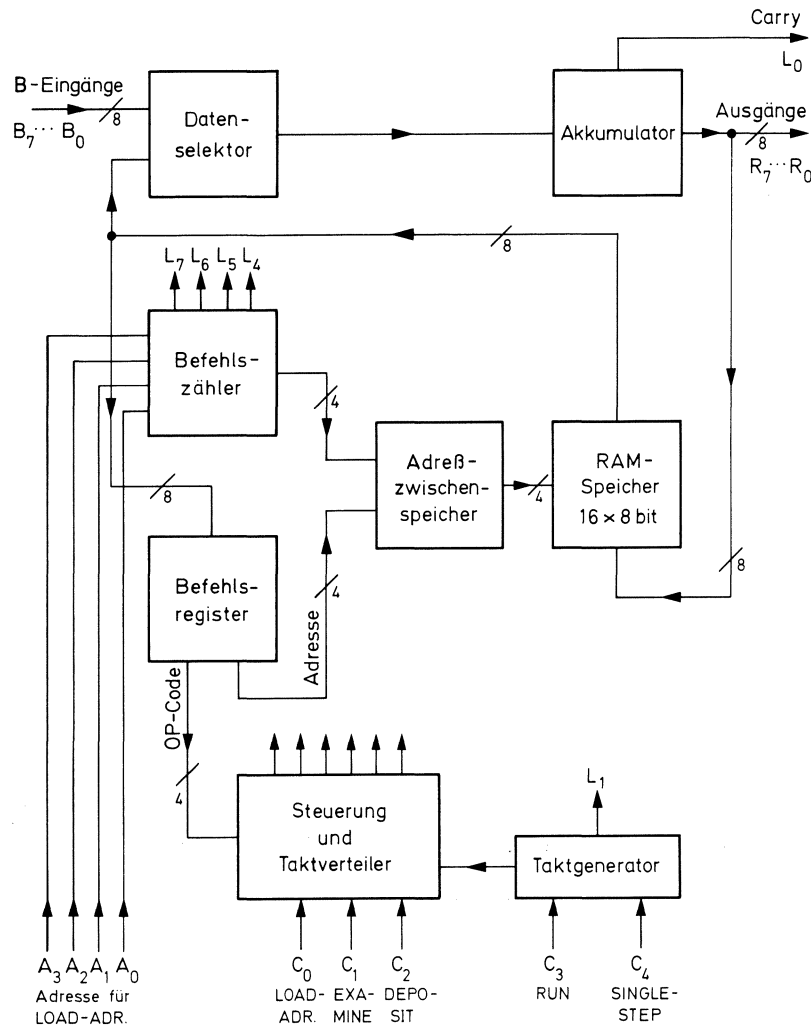
Mit den Schaltern A_3 bis A_0 können Sie den Befehlszähler auf eine bestimmte Adresse laden.

Wenn Sie z.B. A_3 bis A_0 auf 0 1 1 0 einstellen und den Schalter LOAD-ADRESS (C_0) betätigen, wird der Befehlszähler auf diese Adresse gesetzt (Anzeige durch L_7 bis L_4). Wenn Sie jetzt mit dem RUN-Schalter weitertakten, zählt der Zähler von dieser Stellung weiter.

Mit den Schaltern B_7 bis B_0 können OP-Code und Adresse eingegeben werden. Hierbei ist unbedingt zu berücksichtigen, daß es sich um einen **Befehl** handelt, der in einer bestimmten Adresse abgespeichert wird. Wenn Sie z.B. B_7 bis B_0 auf 0 1 1 1 0 0 1 0 einstellen und den Schalter DEPOSIT C_2 takten, wird dieser Befehl in der Adresse abgespeichert, die gerade vom Befehlszähler selektiert ist. Der Befehl 0 1 1 1 0 0 1 0 besagt laut Tabelle: Addiere den Inhalt der Adresse 0 0 1 0 zum Inhalt des Akkus. Dies bedeutet – und das ist unbedingt zu beachten – daß bei der Befehlszählerstellung, bei der dieser Befehl eingegeben wurde, diese Rechenoperation durchgeführt wird.

Die abzuarbeitende Folge von Steuerwörtern oder Befehlen (das Programm) wird zunächst in den Programmspeicher geladen. Dabei ist natürlich die Reihenfolge der einzelnen Befehle wichtig. Die Befehle werden deshalb im Programmspeicher mit **steigenden aufeinanderfolgenden** Adressen gespeichert. Wenn dann über einen Zähler die Programmspeicheradressen automatisch erzeugt werden, erscheinen die Befehle in der richtigen Reihenfolge und können nacheinander ausgeführt werden. Bevor man allerdings ein solches System benutzen kann, muß das Programm zunächst in den Programmspeicher geladen werden.

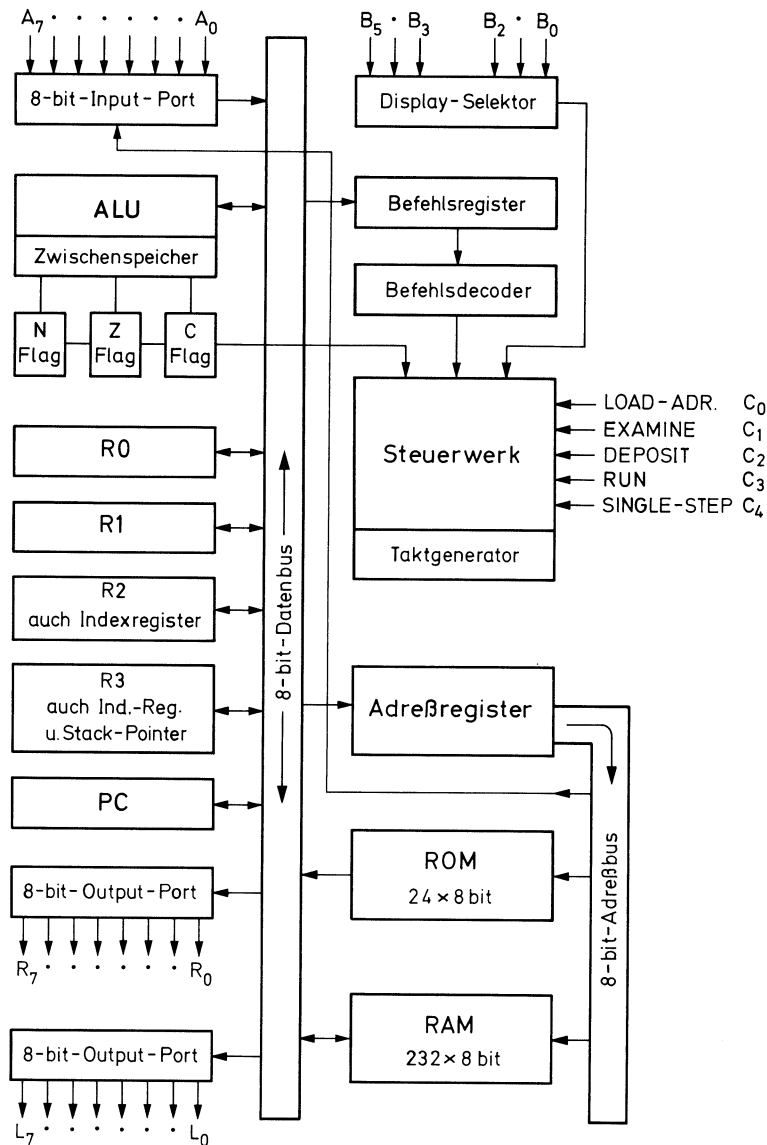
Damit der Rechner anhält, wenn das Programm abgearbeitet worden ist, muß am Ende eines Programms ein HALT-Befehl den Ablauf stoppen. Ohne diesen Befehl hätte das Programm kein Ende. Der Rechner würde auch die Daten ausführen und am Ende des Speichers wieder von vorne beginnen. Dem HALT-Befehl ist der OP-Code 1 1 1 1 zugeordnet.



Zur Steuerung des Rechenablaufes wird ein **Steuerwerk** benötigt. Ein solches Steuerwerk ist recht kompliziert, und wir werden uns deshalb im Rahmen dieses Lehrganges auf eine kurze Beschreibung beschränken. Die Aufgabe des Steuerwerkes ist es, die verschiedenen Taktimpulse und Steuerwörter für die einzelnen Stufen des Rechners zu erzeugen. Die zu erzeugenden Steuerimpulse hängen jeweils vom gerade auszuführenden Befehl ab. Der Befehlszähler zeigt an, welcher Befehl des Programms (z.B. Nr. 17 des Programms) ausgeführt werden soll. Der Befehlszählerinhalt wird also zuerst auf die Adreßeingänge des Speichers übertragen. Der auszuführende Befehl wird jetzt aus dem Speicher geholt und im Befehlsregister zwischengespeichert. Da der Befehl im allgemeinen aus dem Operationsteil (B₇ . . . B₄) und dem Adreßteil (B₃ . . . B₀) besteht, muß der Inhalt des Befehlsregisters in Operations- und Adreßteil aufgespalten werden. Aus dem Operationsteil (Op-Code) des Befehles erkennt das Steuerwerk durch eine entsprechende Logik, ob dieser Befehl eine Adresse benötigt oder nicht. Wenn nicht, veranlaßt das Steuerwerk direkt die entsprechende Operation (z.B. Op-Code = 0 0 0 1). Wenn ja, wird der Adreßteil über den Adreßzwischen-speicher auf die Adreßeingänge des Speichers gegeben. Das unter der angesprochenen Adresse liegende Datenwort gelangt aus dem Speicher zur Ausführung der Operation in den Akkumulator. Damit ist der Befehl ausgeführt, und der Rechner kann nach Erhöhen des Befehlszählers den nächsten Befehl der Programmliste durchführen. War dieser Befehl ein HALT-Befehl (letzter Befehl jedes Programms), stoppt das Steuerwerk den Rechenablauf.

3.7 Hypothetischer Mikrorechner (SYSTEM 5)

Der ebenfalls per Programm simulierte „Hypothetische Rechner“ (HR) ist im Gegensatz zum vereinfachten Rechner ein Rechner mit Bus-Struktur. Das Bild stellt ein Blockschaltbild dieses Rechners dar.



Wie das Blockschaltbild zeigt, besitzt der HR 4 Arbeitsregister (R0 bis R3), von denen R2 und R3 als Indexregister für besondere Adressierung und R3 zusätzlich als Stack-Pointer benutzt wird.

Zur Signalisierung der Registerzustände sind ein Negativ-Flag (N), ein Zero-Flag (Z) und ein Carry-Flag (C) vorhanden, die als Sprung-Conditions verwendet werden.

Der 8 bit breite Programmzähler (PC) kann $2^8 = 256$ Adressen spezifizieren. Im Adreßbereich 0 bis 24 (0_{16} bis 18_{16}) ist ein ROM untergebracht mit Betriebsprogramm. Dem Benutzer steht ein RAM mit 231 Plätzen im Adreßbereich von 25 bis 255 (19_{16} bis FF_{16}) zur Verfügung.

In diesen RAM-Plätzen ist bei Bedarf auch der Stack unterzubringen.

Die Adresse F_{16} ist als Speicher-Adresse auch nicht verfügbar, da sie für den Input-Port (A-Schalter) reserviert ist, der damit wie ein Speicherplatz angesprochen werden kann.

Das Befehlsformat des HR besteht aus 8 bit, davon 4 bit (MSD) als Op-Code und 4 bit (LSD) zur Adressierung. Aufgrund des 4-bit-Op-Codes ergeben sich 16 Grundbefehle, die die Tabelle zeigt.

Es gibt 1- und 2-Byte-Befehle, wobei das 2. Byte je nach Adreßmode eine Konstante (Daten) oder eine Adresse sein kann.

Befehl	Maschinen-Code OP-Code Adressierung	Byte- Anzahl	Funktion	Flags N Z C
HALT	0 0 0 0 0 0 0 0	1	hält Rechner an	- - -
NOP	0 0 0 0 1 1 1 1	1	keine Operation	- - -
MOVE	0 0 0 0 s s d d	1	(s s) → d d	- - -
ADDR	0 0 0 1 s s d d	1	(s s) + (d d) → d d	↑ ↑ ↑
SUBR	0 0 1 0 s s d d	1	(d d) - (s s) → d d	↑ ↑ ↑
IORR	0 0 1 1 s s d d	1	(s s) V (d d) → d d	↑ ↑ 0
XORR	0 1 0 0 s s d d	1	(s s) ∨ (d d) → d d	↑ ↑ 0
ANDR	0 1 0 1 s s d d	1	(s s) ∧ (d d) → d d	↑ ↑ 0
R . . .	0 1 1 0 e e s s	1	(f (s s)) → s s	*)
STAC	0 1 1 1 m m s s	1/2	(s s) → (m m)	- - -
LOAD	1 0 0 0 m m d d	1/2	(m m) → d d	- - -
ADDM	1 0 0 1 m m d d	1/2	(m m) + (d d) → d d	↑ ↑ ↑
SUBM	1 0 1 0 m m d d	1/2	(d d) - (m m) → d d	↑ ↑ ↑
IORM	1 0 1 1 m m d d	1/2	(m m) V (d d) → d d	↑ ↑ 0
XORM	1 1 0 0 m m d d	1/2	(m m) ∨ (d d) → d d	↑ ↑ 0
ANDM	1 1 0 1 m m d d	1/2	(m m) ∧ (d d) → d d	↑ ↑ 0
JMP . . .	1 1 1 0 m m c c	1/2	(m m) → PC Sprung	- - -
CAL . . .	1 1 1 1 m m c c	1/2	(PC) → (R3), (m m) → PC	- - -

Register-Adresse

OP-Code-Erweiterung e e

*)

s s/d d	Register	e e	Befehl	Reg.-Funktion	N Z C
0 0	R0	0 0	INCR	(s s) + 1 → s s	↑ ↑ -
0 1	R1	0 1	DECR	(s s) - 1 → s s	↑ ↑ -
1 0	R2	1 0	RACL	ROT s s u. C links	- - ↑
1 1	R3	1 1	RACR	ROT s s u. C rechts	- - ↑

- = Flag wird nicht beeinflusst

↑ = Flag wird beeinflusst, wird 0 oder 1

0 = Flag wird auf 0 gesetzt

Adreßmode m m

Befehls- gruppe	m m = 0 0	m m = 0 1	m m = 1 0	m m = 1 1
LOAD ADDM SUBM IORM XORM ANDM	# IMMEDIATE 2-Byte-Befehl 2. Byte = Daten	ABSOLUT oder DIREKT 2-Byte-Befehl 2. Byte = Adr.	@ R2 INDEXED über R2 1-Byte-Befehl R2 enth. Adr.	@ R3 ↑ AUTO INCRE- MENT INDEXED über R3 1-Byte-Befehl R3 enth. Adr.
STAC	@ ↓ R3 INDEXED AUTO DECREM. über R3 1-Byte-Befehl R3 enth. Adr.	ABSOLUT oder DIREKT 2-Byte-Befehl 2. Byte = Adr.	@ R2 INDEXED über R2 1-Byte-Befehl R2 enth. Adr.	@ R3 ↑ AUTO INCRE- MENT INDEXED über R3 1-Byte-Befehl R3 enth. Adr.

Fortsetzung der Tabelle auf Seite 14!

JMP . . .	ABSOLUT oder DIREKT 2-Byte-Befehl 2. Byte = Spr.-Adr.	@ INDIREKT 2-Byte-Befehl 2. Byte = Adr. d. Spr.-Adr.	@ R2 INDEXED über R2 1-Byte-Befehl R2 enth. Spr.-Adr.	@ R3 ↑ AUTO INCRE- MENT INDEXED über R3 1-Byte-Befehl R3 enth. Spr.-Adr.
CAL . . .	ABSOLUT oder DIREKT 2-Byte-Befehl 2. Byte = Spr.-Adr.	@ INDIREKT 2-Byte-Befehl 2. Byte = Adr. d. Spr.-Adr.	@ R2 INDEXED über R2 1-Byte-Befehl R2 enth. Spr.-Adr.	nicht verwendet

Sprung-Bedingungen c c

c c	Befehl	Sprung- u. Aufrufbedingung
0 0	JUMP, CALL	unbedingt
0 1	. . . Z	falls Z-Flag = 1
1 0	. . . N	falls N-Flag = 1
1 1	. . . C	falls C-Flag = 1

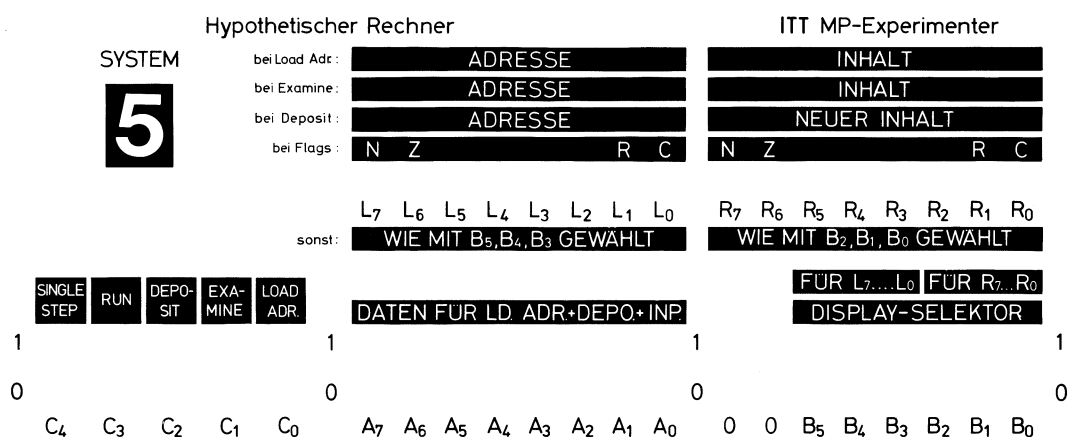
Der Befehl „R . . .“ (Registermanipulation) hat noch eine 2-bit-Op-Code-Erweiterung mit dem Code „e e“. Damit gibt es hier 4 Befehle, wie die Tabelle zeigt.

Auch bei den Befehlen „JMP . . .“ und „CAL . . .“ wird der Op-Code mit den beiden Bedingungs-bits „c c“ erweitert, so daß jeweils 4 verschiedene Sprungbefehle entstehen.

Mit der Adressierung „s s“ bzw. „d d“ wird je eins von den 4 vorhandenen Registern zum Quellregister (source, Code s s) bzw. zum Zielregister (destination, Code d d) ernannt.

Die Codierung „m m“ legt den eigentlichen Adressierungsmodus entsprechend der Tabelle fest. Damit können alle Befehle, die m m enthalten, je nach Wahl des Modes, 1- oder 2-Byte-Befehle werden.

Alle über R2 oder R3 indexierten Adressierungen sind 1-Byte-Befehle, alle immediate, absoluten (direkten) und indirekten ergeben 2-Byte-Befehle.



Bedeutung der Schalter:

Die Schaltergruppe C₄ bis C₀ hat die gleiche Bedeutung wie im System 4. Wie aus der aufgelegten Schablone zu erkennen ist, wird bei LOAD-ADR. = 1 die an den Schaltern A₇ bis A₀ eingestellte Adresse angewählt. Solange C₀ = 1 ist, erscheint in den LEDs L₇ bis L₀ die angewählte Adresse, der Inhalt dieser Adresse wird in R₇ bis R₀ angezeigt. Das gleiche gilt für die Betätigung von EXAMINE und DEPOSIT. Bei DEPOSIT = 1 erscheint in R₇ bis R₀ der mit dem Takt abgespeicherte neue Adresseninhalt. Werden die erwähnten Schalter wieder

in die Stellung 0 zurückgesetzt, erscheinen in den beiden Anzeigenreihen die Daten, die über die Schalter A₅ bis A₀ abgerufen werden. Hierbei kann mit den Schaltern B₅ bis B₃ die Anzeige L₇ bis L₀ und mit den Schaltern B₂ bis B₀ die Anzeige R₇ bis R₀ angewählt werden.

Die Schalter B₇ und B₆ haben keine Bedeutung. Bei gleicher Einstellung von B₅ bis B₃ und B₂ bis B₀ erscheinen in R₇ bis R₀ und L₇ bis L₀ die gleichen Daten. Der Anzeigecode (Display-Code) spezifiziert die einzelnen Anzeigemöglichkeiten:

B ₅ bis B ₃ bzw. B ₂ bis B ₀	Anzeige in L ₇ bis L ₀ bzw. R ₇ bis R ₀
0 0 0	Inhalt von R ₀
0 0 1	Inhalt von R ₁
0 1 0	Inhalt von R ₂
0 1 1	Inhalt von R ₃
1 0 0	Stellung des Befehlszählers PC
1 0 1	Zustände der Flags
1 1 0	Speicherwort, dessen Adresse vom Befehlszähler spezifiziert ist
1 1 1	Speicherwort, dessen Adresse mit A ₇ bis A ₀ spezifiziert ist

Die Codes 0 0 0 bis 0 1 1 bedürfen keiner weiteren Erklärung.

- Code 1 0 0 zeigt an, welche Adresse vom Befehlszähler gerade angewählt wird.
- Code 1 0 1 gibt Auskunft darüber, welche Zustände die 4 Flags gerade einnehmen.
- Code 1 1 0 gibt den Inhalt der Speicheradresse an, die vom Befehlszähler gerade angewählt ist.
- Code 1 1 1 zeigt den Inhalt einer Speicheradresse an, die mit den Schaltern A₇ bis A₀ frei wählbar ist.

Wesentlich ist, daß die Schalter B₅ bis B₀ keinen Einfluß auf den Funktionsablauf haben. Sie dienen lediglich zur Anzeige bestimmter Daten, wenn ein Programm abläuft.

Damit Sie mit diesen grundsätzlichen Eigenschaften des Rechners vertraut werden, führen Sie nachfolgendes Übungsprogramm Schritt für Schritt durch:

1. Alle Schalter auf Null stellen
 2. A₇ bis A₀ auf 4 0₁₆ einstellen
 3. LOAD-ADR. auf 1 stellen (nicht takten!)
- In L₇ bis L₀ erscheint die an A₇ bis A₀ eingestellte Adresse, deren momentaner Inhalt in R₇ bis R₀ angezeigt wird
4. LOAD-ADR. auf 0 stellen, in L₇ bis L₀ und R₇ bis R₀ erscheint der zufällige Inhalt von R₀, da mit den Schaltergruppen B₅ bis B₃ und B₂ bis B₀ jeweils der Code 0 0 0 eingestellt ist
 5. Schalter B₂ bis B₀ auf 1 0 0 einstellen. In R₇ bis R₀ erscheint die Adresse 4 0₁₆, die über LOAD-ADR. geladen wurde
 6. A₇ bis A₀ auf 1 0 0 0 0 1 0 0 einstellen und DEPOSIT auf 1 stellen. In L₇ bis L₀ erscheint die Adresse 4 0₁₆, in R₇ bis R₀ die an A₇ bis A₀ eingestellten Daten. Schalter DEPOSIT auf 0 zurückstellen. Bei B₂ bis B₀ = 1 0 0 erscheint jetzt in R₇ bis R₀ die nächste Adresse 4 1₁₆
 7. A₇ bis A₀ auf F E₁₆ einstellen und DEPOSIT takten
 8. A₇ bis A₀ auf 0 1₁₆ einstellen und DEPOSIT takten
 9. A₇ bis A₀ auf 1 1₁₆ einstellen und DEPOSIT takten
 10. A₇ bis A₀ auf 0 6₁₆ einstellen und DEPOSIT takten
 11. A₇ bis A₀ auf 0 0₁₆ einstellen und DEPOSIT takten
 12. A₇ bis A₀ auf 4 0₁₆ stellen und LOAD-ADR. takten

Wenn Sie alle Eingaben genau vorgenommen haben, ist der Rechner mit einem bestimmten Programm geladen und über Punkt 12 wieder auf die Anfangsadresse 4 0₁₆ zurückgestellt. Mit EXAMINE überprüfen Sie die Eingaben:

1. Alle Schalter auf 0
2. EXAMINE auf 1 stellen. In L₇ bis L₀ erscheint die Adresse 4 0₁₆, in R₇ bis R₀ deren Inhalt 8 4₁₆, also der erste Befehl des Programms. Hierbei handelt es sich um den Befehl LOAD R₀, F E₁₆ (lade Akkumulator R₀ mit dem Inhalt der Adresse F E₁₆). Diese **Datenadresse** ist in der nächsten Programmadresse abgespeichert

3. EXAMINE über 0 wieder auf 1 stellen. In L_7 bis L_0 erscheint die Adresse $4\ 1_{16}$, der Inhalt $F\ E_{16}$ erscheint in R_7 bis R_0
4. Punkt 3. wiederholen. In L_7 bis L_0 erscheint Programmadresse $4\ 2_{16}$. Der Inhalt dieser Adresse $0\ 1_{16}$ (Anzeige in R_7 bis R_0) entspricht dem Befehl MOVE R_1, R_0 . Dieser Befehl bewirkt, daß die Daten aus R_0 in R_1 transferiert werden
5. Punkt 4. wiederholen. In L_7 bis L_0 erscheint Adresse $4\ 3_{16}$. Der Inhalt $1\ 1_{16}$ entspricht dem Befehl ADDR R_1, R_0 . Hierbei werden die Daten des Registers R_0 mit dem Inhalt R_1 in R_1 addiert
6. Punkt 5. wiederholen. In der Adresse $4\ 4_{16}$ ist der Befehl $0\ 6_{16}$, d.h. MOVE R_2, R_1 , abgespeichert. Hierbei werden die Daten von R_1 in R_2 gebracht
7. Punkt 6. wiederholen. In der Adresse $4\ 5_{16}$ ist der HALT-Befehl abgespeichert

Damit dieses Programm mit definierten Daten ablaufen kann, speichern Sie in Adresse $F\ E_{16}$ die Zahl $1\ 0_{16}$:

1. A_7 bis A_0 auf $F\ E_{16}$ einstellen
2. Schalter LOAD-ADR. takten
3. A_7 bis A_0 auf $1\ 0_{16}$ einstellen
4. DEPOSIT takten

Jetzt stellen Sie über LOAD-ADR. das System wieder auf die Programmadresse $4\ 0_{16}$ zurück und schalten über C_4 SINGLE-STEP-Betrieb ein.

Adresse (hexadez.)	Inhalt Maschinencode	Befehl	Kommentar
4 0	1 0 0 0 0 1 0 0	LOAD $R_0, F\ E$	Datenadresse
4 1	1 1 1 1 1 1 1 0		
4 2	0 0 0 0 0 0 0 1	MOVE R_1, R_0	
4 3	0 0 0 1 0 0 0 1	ADDR R_1, R_0	
4 4	0 0 0 0 0 1 1 0	MOVE R_2, R_1	
4 5	0 0 0 0 0 0 0 0	HLT	
.			
.			
.			
F E	0 0 0 1 0 0 0 0	Daten	

Zur Beobachtung des Datenflusses stellen Sie B_2 bis B_0 auf $1\ 0\ 0$ (Stellung PC) und B_5 bis B_3 auf $0\ 0\ 0$ (Inhalt R_0). Mit Schalter RUN wird jetzt das System einmal getaktet. In L_7 bis L_0 erscheint $1\ 0_{16}$, also die Daten aus Adresse $4\ 1_{16}$, da hier (durch MODE $0\ 1$ im Maschinencode bedingt) kein neuer Befehl, sondern eine Datenadresse abgespeichert ist.

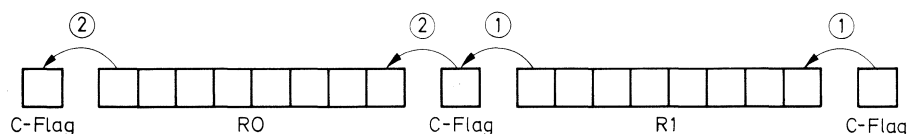
Schalter B_5 bis B_3 auf $0\ 0\ 1$ einstellen, und RUN erneut takten. Die Daten von R_0 erscheinen jetzt auch in R_1 . System mit RUN takten. In L_7 bis L_0 erscheint das Additionsergebnis $2\ 0_{16}$.

Schalter B_5 bis B_3 auf $0\ 1\ 0$ stellen und mit RUN takten. Die Daten von R_1 erscheinen auch in R_2 . Weiteres Takten hat keinen Einfluß mehr auf das System, da in Adresse $4\ 5_{16}$ ein HALT-Befehl programmiert ist.

Wie bereits erwähnt, ist im Adreßbereich $0\ 0_{16}$ bis $1\ 8_{16}$ des ROMs ein Betriebsprogramm untergebracht. Dieses Betriebsprogramm ist ein Multiplikationsprogramm, das als Unterprogramm benutzt werden kann:

Adresse	Inhalt	Befehl	Kommentar
0 1	7 5	STAC R1, F 0	speichere Multiplikand 1)
0 2	F 0		
0 3	4 5	XORR R1, R1	lösche Zwischensumme
0 4	8 2	LOAD R2, # 0 8	setze Schleifenzähler
0 5	0 8		
0 6	6 9	RACL R1	verschiebe rechte Hälfte Zwischensumme 2)
0 7	6 8	RACL R0	verschiebe linke Hälfte Zwischensumme
0 8	E 3	JMPC 0 C	prüfe Carry-Flag 3)
0 9	0 C		
0 A	E 0	JUMP 1 4	
0 B	1 4		
0 C	9 5	ADDM R1, F 0	C-Flag war 1, addiere
0 D	F 0		
0 E	E 3	JMPC 1 2	4)
0 F	1 2		
1 0	E 0	JUMP 1 4	
1 1	1 4		
1 2	9 0	ADDM R0, # 0 1	C-Flag war 1, addiere 5)
1 3	0 1		
1 4	6 6	DECR R2	erniedrige Schleifenzähler
1 5	E 1	JMPZ 1 9	Schleifenende?
1 6	1 9		
1 7	E 0	JUMP 0 6	
1 8	0 6		

1. Der Multiplikand steht in R1. Dieses Register wird später für die rechte Hälfte der Zwischensumme benötigt. Aus diesem Grunde muß der Multiplikand abgespeichert werden. Dies geschieht in unserem Beispiel unter der Adresse F 0.
2. Die Verschiebung der Zwischensumme erfolgt in 2 Schritten. Zuerst wird die rechte Hälfte (R1) verschoben. Das herausgeschobene bit kommt in das Carry-Flag. Danach wird die linke Hälfte (R0) verschoben. Der Inhalt des Carry-Flags wird dabei von rechts in R0 hineingeschoben, so daß insgesamt eine Verschiebung mit doppelter Wortlänge durchgeführt wird:



Dabei ist noch zu beachten, daß durch den Befehl RACL 2 1 der Inhalt des Carry-Flags von rechts in R1 hineingeschoben wird. Über das Programm muß deshalb sichergestellt werden, daß vor dieser Instruktion das Carry-Flag 0 enthält. Dies wird beim ersten Programmdurchlauf durch die Instruktion XORR R1, R1 garantiert. Dieser Befehl bewirkt nämlich ein Löschen des Carry-Flags. Auch in der Schleife muß diese Bedingung erfüllt sein.

3. und 4. Die folgenden Programmschritte (Addition) müssen übersprungen werden, wenn das Carry-Flag Null ist. Ein solcher Befehl ist im hypothetischen Rechner nicht vorhanden. Er muß daher durch einen bedingten und unbedingten Sprung realisiert werden, was natürlich zu einem längeren Programm führt.

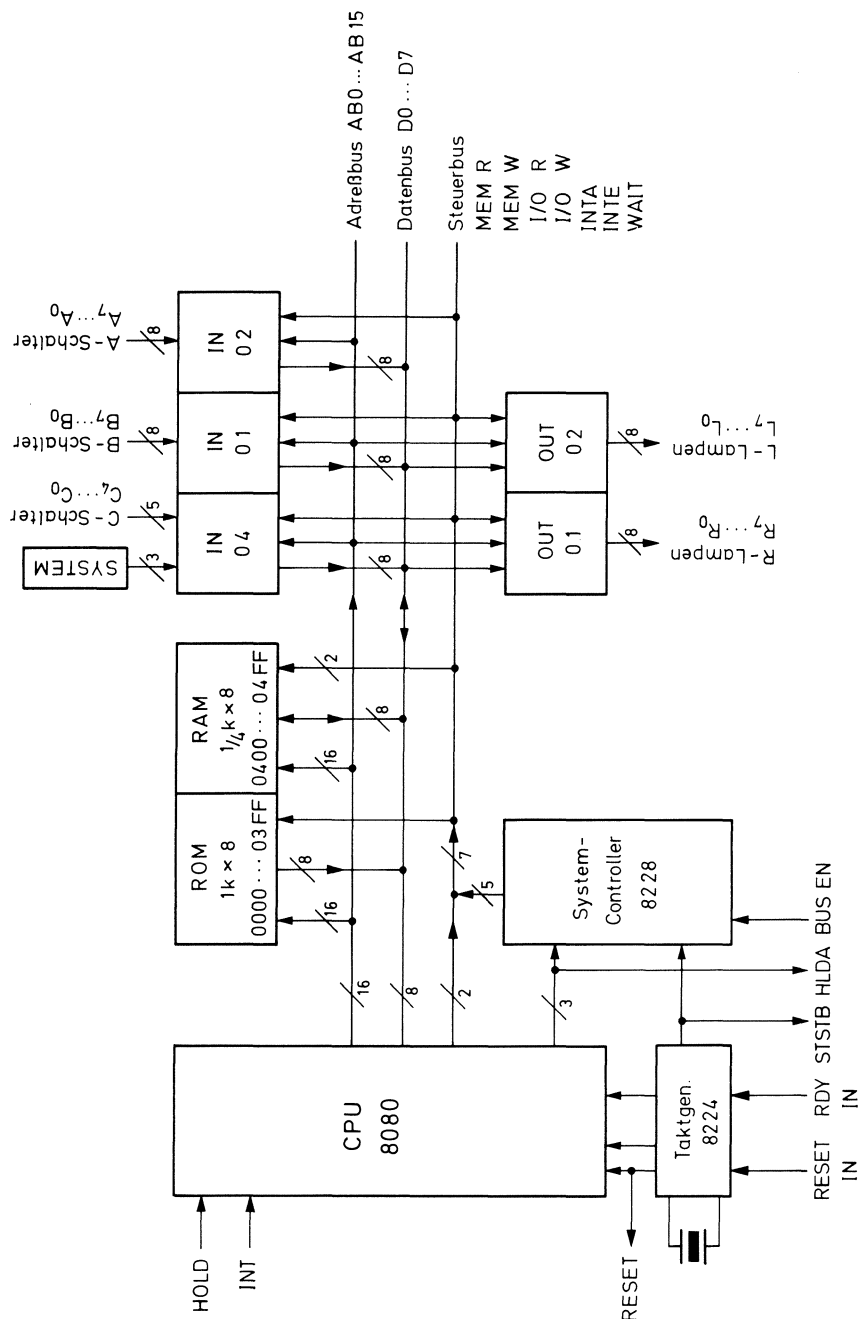
5. Dieser Befehl addiert 1 zur linken Hälfte der Zwischensumme und löscht gleichzeitig das Carry-Flag, da sich bei dieser Addition kein Übertrag ergeben kann. Damit ist sichergestellt, daß im nächsten Schleifendurchlauf mit dem Befehl RACL R1 eine Null von rechts in die Zwischensumme geschoben wird. Zum Erhöhen von R0 könnte auf den ersten Blick auch der Befehl INCR R0 verwendet werden. Dies ist nicht möglich, da dieser Befehl das Carry-Flag nicht beeinflusst.

Dieses Unterprogramm kann, wie das folgende Beispiel zeigt, in ein Gesamtprogramm für Multiplikationen eingebaut werden:

Adresse	Inhalt	Befehl	Kommentar	
4 0	8 3	LOAD R3, # F F	Initialisiere Stack-Pointer	
4 1	F F			
4 2	F 0	CALL MULTA		
4 3	2 6			
4 4	5 0	ADR F 1		Adresse von Faktor 1
4 5	5 1	ADR F 2		Adresse von Faktor 2
4 6	5 2	ADR P		Produktadresse
4 7	0 0	HALT		
2 6	0 E	MOVE R2, R3	Unterprogramm	
2 7	8 A	LOAD R2, @ R2		
2 8	8 A	LOAD R2, @ R2		
2 9	8 8	LOAD R0, @ R2		
2 A	0 E	MOVE R2, R3		
2 B	8 A	LOAD R2, @ R2		
2 C	6 2	INCR R2		
2 D	8 A	LOAD R2, @ R2		
2 E	8 9	LOAD R1, @ R2		
2 F	E 0	JUMP 0 1		
3 0	0 1	JUMP 0 1		
0 1		Multiplikations- programm im ROM-Bereich		
0 2				
.				
.				
.				
.				
1 7				
1 8				
1 9	0 E	MOVE R2, R3		
1 A	8 A	LOAD R2, @ R2		
1 B	9 2	ADDM R2, @ 0 2		
1 C	0 2			
1 D	8 A	LOAD R2, @ R2		
1 E	7 9	STAC R1, @ R2		
1 F	6 2	INCR R2		
2 0	7 8	STAC R0, @ R2		
2 1	8 E	LOAD R2, @ R3 ↑		
2 2	9 2	ADDM R2, # 0 3		
2 3	0 3			
2 4	7 2	STAC R2, @ ↓ R3		
2 5	E C	JUMP @ R3 ↑		

3.8 Mikrorechner-System 8080 (SYSTEM 6)

Das nachstehende Blockschaltbild zeigt den Aufbau des Mikrorechners mit der CPU 8080.



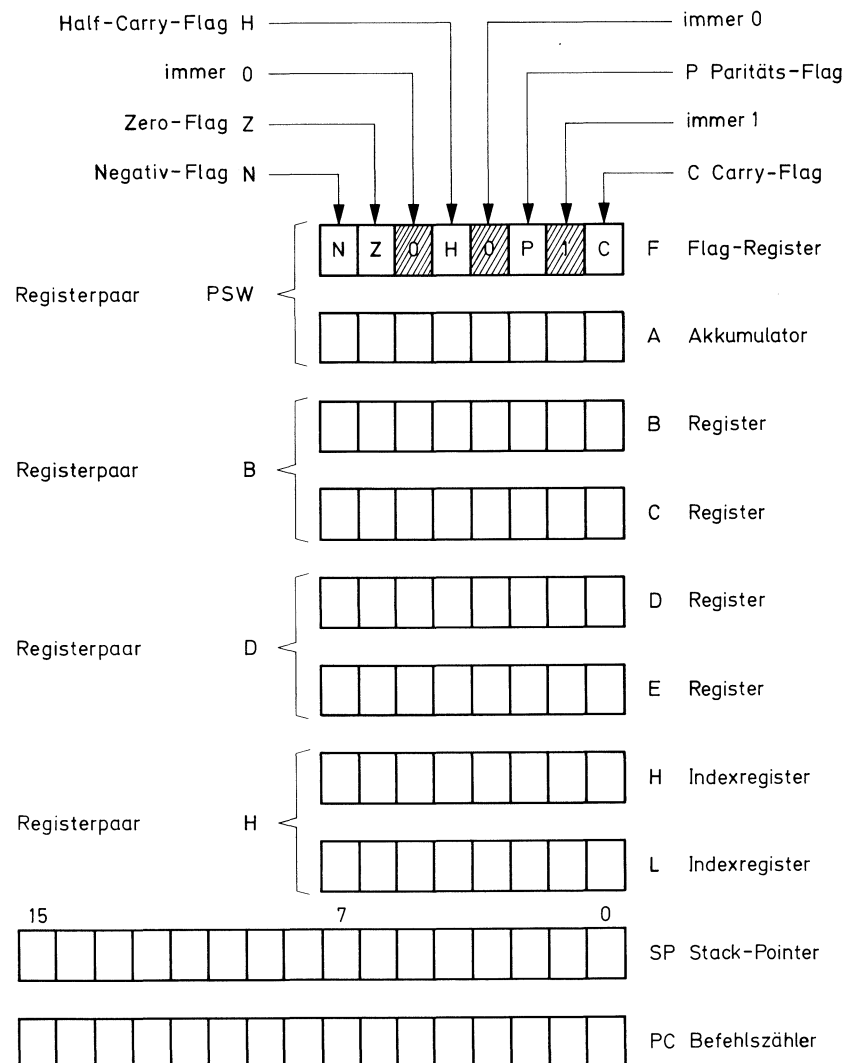
Die CPU 8080 mit dem Taktgenerator 8224 und dem System-Controller 8228 liefert alle Adreß-, Daten- und Steuersignale, die auf der linken Anschlußleiste verfügbar sind.

Im 1-k-ROM mit dem Adreßbereich 0 0 0 0₁₆ bis 0 3 F F₁₆ ist das Monitorprogramm dieses Systems (und der niederen Systeme) untergebracht. Zur Verfügung des Anwenders steht das 1/4-k-RAM im absoluten Adreßbereich 0 4 0 0₁₆ bis 0 4 F F₁₆, seine Adressen werden vom Monitor in die relativen Adressen 0 0₁₆ bis F F₁₆ umgesetzt. Am Datenbus liegen 3 8-bit-Input-Ports und 2 8-bit-Output-Ports, die mit isolierter I/O-Adressierung betrieben werden. Sie werden mit einzelnen Adreßbits adressiert, so daß maximal je 8 Input- bzw. Output-Ports adressiert werden können (Adressen 0 1, 0 2, 0 4, 0 8, 1 0, 2 0, 4 0, 8 0 davon 0 1, 0 2, 0 4 bzw. 0 1, 0 2 verwendet).

Dem Benutzer stehen die im nachstehenden Bild dargestellten Register der CPU zur Verfügung.

Im Flag-Register (F) sind die Flags „N“ (Negativ), „Z“ (Zero), „H“ (Half-Carry für Überträge vom 3. ins 4. bit, Verwendung für BCD-Operationen), „P“ (Parität, d.h. bei einer geraden

Anzahl von 1 in einem Register wird $P = 1$) und „C“ (Carry) vorhanden. Die übrigbleibenden 3 bits sind fest mit 0 bzw. 1 belegt. Das 8-bit-A-Register ist das Haupt-Arbeitsregister (Akkumulator), das gegenüber den übrigen Registern mit einigen besonderen Befehlen arbeiten kann.



Die je 8 bit breiten Register B, C, D, E, H und L sind gleichwertige Arbeitsregister. Bei einigen Befehlen werden die Register als Registerpaare benutzt, so bilden F und A das Registerpaar PSW (Program Status Word), B und C das Paar B, D und E das Paar D sowie H und L das Paar H. Die Registerpaare B, D und H erlauben dann 16-bit-Operationen. Das Paar H dient bei einigen Befehlen als Indexregister, d.h. es erlaubt auf einfache Weise eine Indexadressierung.

Weiterhin steht ein 16-bit-Stack-Pointer (Stapelzeiger) zur Verfügung, der bei Programmsprüngen usw. die Adresse des Speicherplatzes anzeigt, in dem die Rücksprungadresse steht.

Ein 16-bit-Programm-Zähler (PC) dient zur Adressierung von insgesamt 64 k (65 536) möglichen Speicherplätzen.

Das 8080-System hat ein 8-bit-Befehlsformat. Dabei ist insgesamt nicht zwischen Op-Code und Adressierung zu unterscheiden, d.h. man muß alle 8 bits als Op-Code verstehen. Damit ergäben sich theoretisch $2^8 = 256$ Befehle, von denen jedoch 12 Möglichkeiten nicht verwendet werden, so daß 244 Befehle übrigbleiben.

Durch Zusammenfassen zu Befehlsgruppen mit zusätzlicher Codierung (innerhalb der 8 bits) ergibt sich eine übersichtliche Form des Befehlsvorrates.

Aufgrund unterschiedlicher Adressierung können die Befehle 1-Byte-, 2-Byte- oder 3-Byte-Befehle sein. Die folgende Tabelle zeigt alle Befehle in der komprimierten Form der Befehlsliste unter Verwendung zusätzlicher Codierungen.

Befehl	Maschinen-Code	Byte	Funktion	Flags				
				N	Z	H	P	C
NOP	0 0 0 0 0 0 0 0	1	keine Operation	-	-	-	-	-
MOV d, s	0 1 d d d s s s	1	(s s s) → d d d	-	-	-	-	-
MVI d, #k.	0 0 d d d 1 1 0	2	konst. → d d d	-	-	-	-	-
LDA adr.	0 0 1 1 1 0 1 0	3	(adr.) → A	-	-	-	-	-
STA adr.	0 0 1 1 0 0 1 0	3	(A) → adr.	-	-	-	-	-
LDAX rp*	0 0 r r 1 0 1 0	1	(@rp) → A	-	-	-	-	-
STAX rp*	0 0 r r 0 0 1 0	1	(A) → @rp	-	-	-	-	-
LHLD adr.	0 0 1 0 1 0 1 0	3	(adr.) → HL	-	-	-	-	-
SHLD adr.	0 0 1 0 0 0 1 0	3	(HL) → adr.	-	-	-	-	-
LXI rp, #k	0 0 r r 0 0 0 1	3	konst. → rp	-	-	-	-	-
XCHG	1 1 1 0 1 0 1 1	1	(HL) ↔ (DE)	-	-	-	-	-
IN A, adr.	1 1 0 1 1 0 1 1	2	(adr.) → A	-	-	-	-	-
OUT adr., A	1 1 0 1 0 0 1 1	2	(A) → adr.	-	-	-	-	-
ADD A, s	1 0 0 0 0 s s s	1	(s s s) + (A) → A	↑	↑	↑	↑	↑
ADC A, s	1 0 0 0 1 s s s	1	(s s s) + (A) + (C) → A	↑	↑	↑	↑	↑
ADI A, #k	1 1 0 0 0 1 1 0	2	(A) + konst. → A	↑	↑	↑	↑	↑
ACI A, #k	1 1 0 0 1 1 1 0	2	(A) + konst. + (C) → A	↑	↑	↑	↑	↑
SUB A, s	1 0 0 1 0 s s s	1	(A) - (s s s) → A	↑	↑	↑	↑	↑
SBB A, s	1 0 0 1 1 s s s	1	(A) - (s s s) - (C) → A	↑	↑	↑	↑	↑
SUI A, #k	1 1 0 1 0 1 1 0	2	(A) - konst. → A	↑	↑	↑	↑	↑
SBI A, #k	1 1 0 1 1 1 1 0	2	(A) - konst. - (C) → A	↑	↑	↑	↑	↑
CMP A, s	1 0 1 1 1 s s s	1	(A) - (s s s)	↑	↑	↑	↑	↑
CPI A, #k	1 1 1 1 1 1 1 0	2	(A) - konst.	↑	↑	↑	↑	↑
DAD H, rp	0 0 r r 1 0 0 1	1	(HL) + (rp) → HL	-	-	-	-	↑
DAA A	0 0 1 0 0 1 1 1	1	(A) + korr. → A	↑	↑	↑	↑	↑
ANA A, s	1 0 1 0 0 s s s	1	(A) ∧ (s s s) → A	↑	↑	↑	↑	0
ANI A, #k	1 1 1 0 0 1 1 0	2	(A) ∧ konst. → A	↑	↑	↑	↑	0
ORA A, s	1 0 1 1 0 s s s	1	(A) ∨ (s s s) → A	↑	↑	0	↑	0
ORI A, #k	1 1 1 1 0 1 1 0	2	(A) ∨ konst. → A	↑	↑	0	↑	0
XRA A, s	1 0 1 0 1 s s s	1	(A) ⊕ (s s s) → A	↑	↑	0	↑	0
XRI A, #k	1 1 1 0 1 1 1 0	2	(A) ⊕ konst. → A	↑	↑	0	↑	0
CMA A, $\bar{A}$	0 0 1 0 1 1 1 1	1	($\bar{A}$) → A	-	-	-	-	-
INR d	0 0 d d d 1 0 0	1	(d d d) + 1 → d d d	↑	↑	↑	↑	-
DCR d	0 0 d d d 1 0 1	1	(d d d) - 1 → d d d	↑	↑	↑	↑	-
INX rp	0 0 r r 0 0 1 1	1	(rp) + 1 → rp	-	-	-	-	-
DCX rp	0 0 r r 1 0 1 1	1	(rp) - 1 → rp	-	-	-	-	-
R...A	0 0 0 n n 1 1 1	1	(A _n) → A	-	-	-	-	↑
PCHL	1 1 1 0 1 0 0 1	1	(HL) → PC	-	-	-	-	-
JMP adr.	1 1 0 0 0 0 1 1	3	(adr.) → PC	-	-	-	-	-
J...adr.	1 1 b b b 0 1 0	3	(adr.) → PC, wenn	-	-	-	-	-
CALL adr.	1 1 0 0 1 1 0 1	3	(adr.) → PC, (PC) → SP	-	-	-	-	-
C...adr.	1 1 b b b 1 0 0	3	(adr.) → PC, wenn (PC) → SP	-	-	-	-	-
RET	1 1 0 0 1 0 0 1	1	((SP)) → PC	-	-	-	-	-
R...	1 1 b b b 0 0 0	1	((SP)) → PC, wenn	-	-	-	-	-
RST a	1 1 a a a 1 1 1	1	8 x a → PC	-	-	-	-	-
POP**	1 1 r r 0 0 0 1	1	((SP)) → rp	(	↑	↑	↑	↑
PUSW**	1 1 r r 0 1 0 1	1	(rp) → (SP)	-	-	-	-	-
SPHL	1 1 1 1 1 0 0 1	1	(HL) → SP	-	-	-	-	-
XTHL	1 1 1 0 0 0 1 1	1	((SP)) ↔ (HL)	-	-	-	-	-
EI	1 1 1 1 1 0 1 1	1	Interrupt ermöglicht	-	-	-	-	-
DI	1 1 1 1 0 0 1 1	1	Interrupt gesperrt	-	-	-	-	-
STC	0 0 1 1 0 1 1 1	1	1 → C	-	-	-	-	1
CMC	0 0 1 1 1 1 1 1	1	($\bar{C}$) → C	-	-	-	-	↑
HLT	0 1 1 1 0 1 1 0	1	Prozessor hält an	-	-	-	-	-

Die verwendeten zusätzlichen Codierungen haben dabei folgende Bedeutungen:

Register-Code			
s s s od. d d d	Register	r r	Reg.-Paar rp
0 0 0	B	0 0	BC = B
0 0 1	C	0 1	DE = D
0 1 0	D	1 0	HL = H
0 1 1	E	1 1	SP
1 0 0	H		
1 0 1	L	*	nur 0 0 und 0 1
1 1 0	M = ((HL))	**	1 1 für PSW
1 1 1	A		dann bei POP Flags †

s s s = Quellregister

d d d = Zielregister

M = Speicherplatz

† = Flag wird beeinflusst

– = Flag wird nicht beeinflusst

1,0 = Flag wird 1,0

Rotate-Code (R . . .) n n		
n n	Befehl	Funktion
0 0	RLC	(A ₇)→A ₀ , (A ₇)→C
0 1	RRC	(A ₀)→A ₇ , (A ₀)→C
1 0	RAL	(A ₇)→C, (C)→A ₀
1 1	RAR	(A ₀)→C, (C)→A ₇

Sprung-Bedingungs-Code b b b		
b b b	Befehlszusatz	Bedingung
0 0 0	... NZ	wenn Z = 0, d.h. Inhalt ≠ 0
0 0 1	... Z	wenn Z = 1, d.h. Inhalt = 0
0 1 0	... NC	wenn C = 0, d.h. kein Übertrag
0 1 1	... C	wenn C = 1, d.h. Übertrag
1 0 0	... PO	wenn P = 0, d.h. Parität ungerade
1 0 1	... PE	wenn P = 1, d.h. Parität gerade
1 1 0	... P	wenn N = 0, d.h. Inhalt positiv
1 1 1	... M	wenn N = 1, d.h. Inhalt negativ

Für die genaue Kenntnis der Wirkungsweise der einzelnen Befehle sollte man die Datenunterlagen der Hersteller des 8080 oder das schriftliche Lehrmaterial des ITT MP-Lehrsystems zu Rate ziehen.

Um nun mit dem Mikrorechner 8080 arbeiten zu können, ist ein Monitorprogramm erforderlich, über das man Einblick in den Zustand bzw. die Arbeitsweise nehmen kann. Das Monitorprogramm für das 8080-System beim ITT MP-Experimenter ist ein für den Lernzweck mit diesem Gerät speziell zugeschnittenes Programm und nicht identisch mit den Monitorprogrammen der 8080-Hersteller.

Im einzelnen erlaubt dieses Programm:

- die Registerinhalte des 8080, den Stand des Befehlszählers und die Inhalte aller RAM-Adressen sichtbar zu machen
- das RAM zu laden (Funktion DEPOSIT) und den RAM-Inhalt automatisch zu kontrollieren (Funktion EXAMINE)

Das Monitorprogramm ist im ROM untergebracht und kann durch Betätigung der RESET-Taste gestartet werden.

Es wird außer Betrieb gesetzt, d.h. das Anwenderprogramm im RAM wird vom 8080 bearbeitet, wenn der RUN-Schalter (C₃) auf 1 gestellt wird.

Wenn RUN = 1 ist, ist der Monitor außer Betrieb und alle übrigen Schalter haben keinen sichtbaren Einfluß mehr. Auch nach dem Rückschalten von RUN auf 0 bleibt das Anwenderprogramm in Betrieb, d.h. das Monitorprogramm läuft nicht. Das Monitorprogramm ist nur bei RUN = 0 durch Betätigung der RESET-Taste wieder in Betrieb zu setzen.

SYSTEM

6

bei Load Adr.:	ADRESSE	INHALT
bei Examine:	ADRESSE	INHALT
bei Deposit:	ADRESSE	NEUER INHALT

sonst: L₇ L₆ L₅ L₄ L₃ L₂ L₁ L₀ R₇ R₆ R₅ R₄ R₃ R₂ R₁ R₀
 Inh. der mit A-Sch. gewählten Adr. Inh. der mit B-Sch. gewählten Adr.



Solange der Monitor in Betrieb ist, können die Inhalte der Register und der Stand des Befehlszählers zur Anzeige gebracht werden. Der Monitor setzt die echten (absoluten) Adressen des RAM-Bereiches 0 4 0 0₁₆ bis 0 4 F F₁₆ in relative Adressen – RAM-selektive Adressen – um, die dann 0 0₁₆ bis F F₁₆ lauten. Die Registerinhalte werden vom Monitor in den STACK-Teil des RAMs gebracht, der die RAM-rel. Adressen F 4₁₆ bis F D₁₆ belegt. Wählt man diese Adressen mit den B-Schaltern, so werden die Register angezeigt. Dadurch, daß die Adressen des STACK festgelegt sind, kann man die Inhalte der Register mit Hilfe der A-Schalter auch über die STACK-relativen Adressen (STACK-Offset) von 0 0₁₆ bis 0 9₁₆ (entspricht F 4₁₆ bis F D₁₆) zur Anzeige bringen. Die nachstehende Tabelle gibt die Codierung an:

abs. Adresse	RAM-rel. Adresse B-Schalter Anz. in R-LEDs	STACK-rel. Adresse A-Schalter Anz. in L-LEDs	Inhalt
0 4 F 4	F 4	0 0	Reg. L
0 4 F 5	F 5	0 1	Reg. H
0 4 F 6	F 6	0 2	Reg. E
0 4 F 7	F 7	0 3	Reg. D
0 4 F 8	F 8	0 4	Reg. C
0 4 F 9	F 9	0 5	Reg. B
0 4 F A	F A	0 6	Flags
0 4 F B	F B	0 7	Akku
0 4 F C	F C	0 8	PC (bit 7 bis 0)
0 4 F D	F D	0 9	PC (bit 15 bis 8)

In dieser Form ist der Monitor in erster Linie zum Programmladen zu verwenden. Um Ergebnisse (Registerinhalte, Speicherinhalte) nach oder während des Laufes eines Anwenderprogramms sichtbar zu machen, gibt es eine andere Möglichkeit. Das Monitorprogramm wird auch dann aufgerufen, wenn der Prozessor innerhalb des Anwenderprogramms einen RST-2-Befehl (D 7₁₆) vorfindet. Mit dem RST-2-Befehl wird ähnlich eines Interrupts der Monitor als Interrupt-Routine aufgerufen, d.h. alle Registerinhalte auf den STACK gebracht. Der STACK-Inhalt ist dann bei entsprechender Adreßwahl anzeigbar, allerdings nur, wenn „HLT am BP“ (C₄) auf 1 steht, da damit der Prozessor im Monitor bleibt, d.h. die Anzeige „steht“. Dadurch ist dann eine Art von „Single-Step-Betrieb“ möglich:

Steht C₄ auf 1, läuft der Prozessor mit jeder RUN-Betätigung bis zum nächsten RST 2 und bleibt stehen. Will man ein Programm austesten, ersetzt man einfach den auf den letzten Befehl des auszutestenden Programmteilers folgenden Befehl durch RST 2. Ist der Programmteil in Ordnung, wird wieder der Originalbefehl eingesetzt und RST 2 an der nächst interessanten Stelle eingebaut usw.

Der Monitoranruf über RST 2 erlaubt die Anzeige der Registerinhalte, aber nicht das Neuladen von Programmen; dies ist nur möglich, wenn der Monitor über RESET angerufen wird.

3.9 Erweitertes 8080-System (SYSTEM 7)

Im Rahmen der in den technischen Daten genannten Möglichkeiten kann das Rechnersystem des ITT MP-Experimenters durch den Benutzer erweitert werden. Eine Speichererweiterung ist auf der Adresse $0\ 8\ 0\ 0_{16}$ anzubringen, da diese durch das vorhandene Betriebsprogramm angesprungen wird, wenn der SYSTEM-Schalter auf 7 gestellt wird.

Ab der Speicheradresse $0\ 8\ 0\ 0_{16}$ kann dann ein vom Benutzer entwickeltes anderes Betriebsprogramm (ROM) oder aber auch RAM untergebracht werden. Das Monitorprogramm ist dann außer Betrieb.

4. Stromlaufpläne und Anschlußbelegung

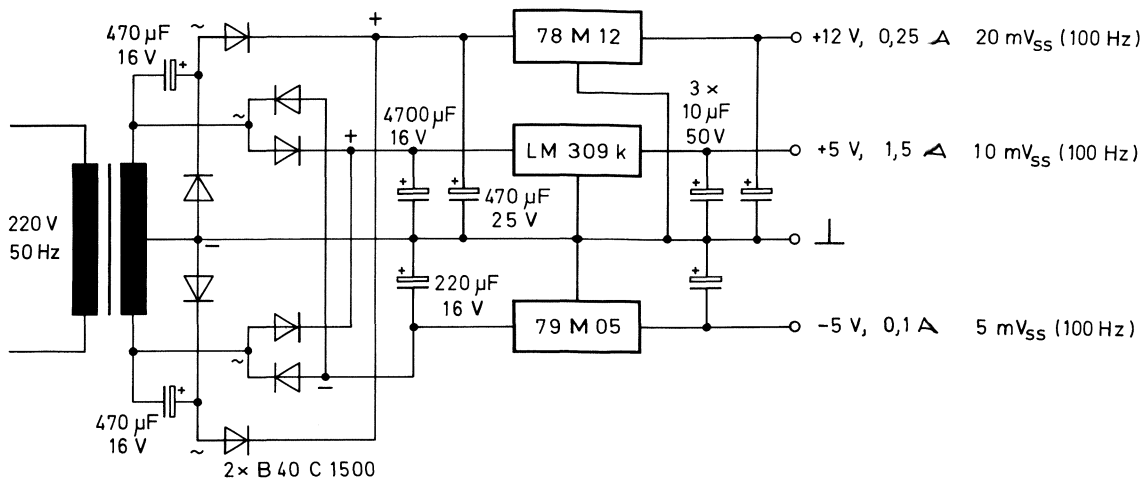


Bild 1
Stromversorgung

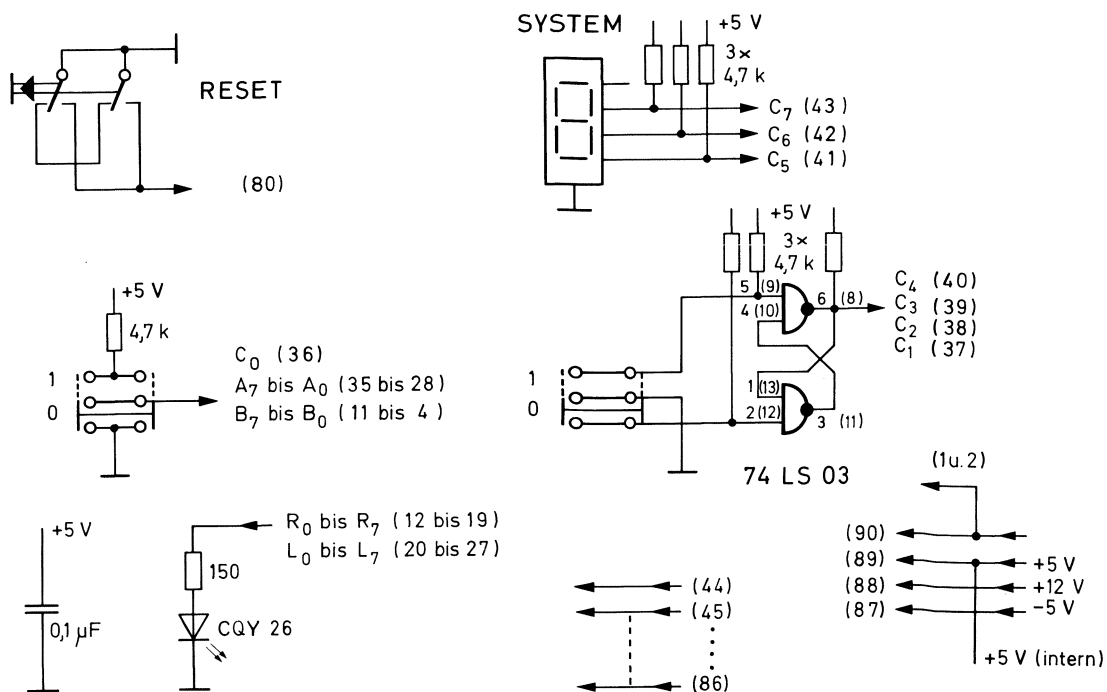


Bild 2
Frontplatte (Schaltungsauszüge)

Bild 3
Anschlußbelegung

⊥	○	90
+5V	○	89
+12V	○	88
-5V	○	87
⊥	○	86
(+5V)	○	85
(+12V)	○	84
(-5V)	○	83
STST B	○	82
RDY IN	○	81
RES IN	○	80
RESET	○	79
INT	○	78
HOLD	○	77
BUS EN	○	76
WAIT	○	75
INTE	○	74
HLDA	○	73
I/O W	○	72
MEM W	○	71
I/O R	○	70
MEM R	○	69
INTA	○	68
AB ₁₅	○	67
.	○	66
.	○	65
.	○	64
.	○	63
.	○	62
.	○	61
.	○	60
.	○	59
.	○	58
.	○	57
.	○	56
.	○	55
.	○	54
.	○	53
AB ₀	○	52
DB ₇	○	51
.	○	50
.	○	49
.	○	48
.	○	47
.	○	46
.	○	45
DB ₀	○	44

linke Leiste

von oben
auf die Frontplatte
gesehen

1	○	⊥
2	○	⊥
3	○	+5V
4	○	B ₀
5	○	.
6	○	.
7	○	.
8	○	.
9	○	.
10	○	.
11	○	B ₇
12	○	R ₀
13	○	.
14	○	.
15	○	.
16	○	.
17	○	.
18	○	.
19	○	R ₇
20	○	L ₀
21	○	.
22	○	.
23	○	.
24	○	.
25	○	.
26	○	.
27	○	L ₇
28	○	A ₀
29	○	.
30	○	.
31	○	.
32	○	.
33	○	.
34	○	.
35	○	A ₇
36	○	C ₀
37	○	.
38	○	.
39	○	.
40	○	.
41	○	.
42	○	.
43	○	C ₇

rechte Leiste

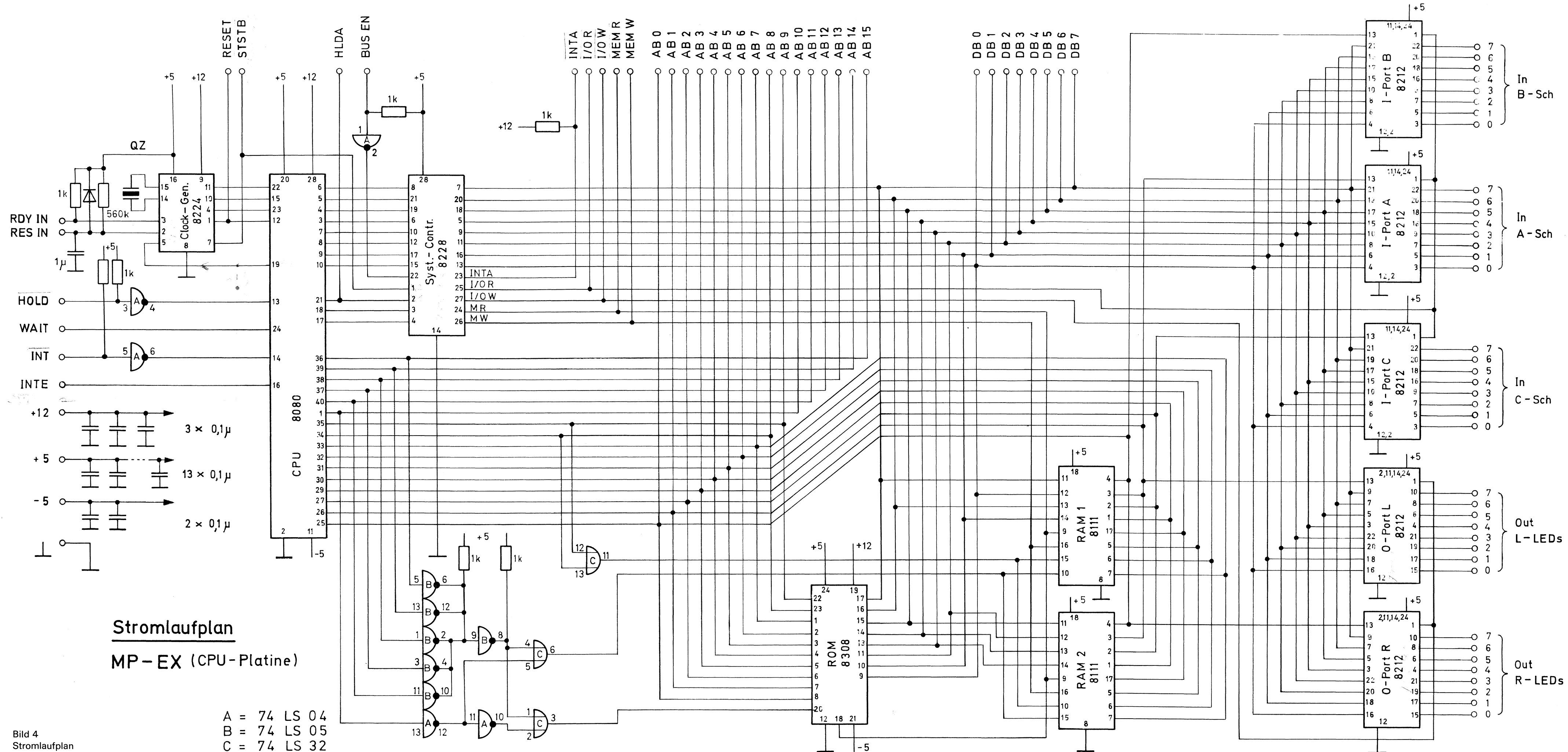


Bild 4
Stromlaufplan