ITT MP-Experimenter

Bedienungsanleitung

ITT Fachlehrgänge, Pforzheim

ITT

1., 2. und 3. Auflage, Juli 1977

Druck: Druckerei Seiter, 7535 Königsbach-Stein

MICROKIT ASSEMBLER -- VER 2.2

```
                ***     ITT VIDEO- UND LEHRSYSTEME
                ***     SEL AG, PFORZHEIM

                *       ITT MP-LEHRSYSTEM
                *       BETRIEBSPROGRAMM: "MON MF"

                *       RESISTENT IN MASKENROM TYP
                *       AM 9208 EXP01 (1 K X 8 BIT)
                *       ODER INTEL 8308 EXP01

                *                       RP/KTEW, UL, 1.10.76

                *       KONSTANTEN- UND NAMENS-TABELLE:
0002            ASCH    EQU  2          *A-SCHALTER
0001            BSCH    EQU  1          *B-SCHALTER
0004            CSCH    EQU  4          *C-SCHALTER
                *                       =FUNKTIONSSCHALTER
0002            LLAMP   EQU  2          *LINKE LED'S
0001            RLAMP   EQU  1          *RECHTE LED'S
0018            RHBP    EQU  X'18'      *BIT 3 UND 4 ALS
                *                       POSITIONSKENNUNG
                *                       DER SCHALTER "RUN"
                *                       UND "HLT AT BP"
04FF            RMTP    EQU  X'04FF'    *HOECHSTE RAM-ADR
0408            RMRST1  EQU  X'0408'    *SPRUNGZIEL IM RAM
                *                       BEI RST 1-BEFEHL
0400            RMBS    EQU  X'0400'    *UNTERSTE RAM-ADR
0410            RAMINT  EQU  X'0410'    *SPRUNGZIEL IM RAM
                *                       BEI RST 7-BEFEHL
                *                       ODER BEI INTERUPT
0002            EXMS    EQU  2          *BIT 1 FUER EXAMIN
0002            EXJA    EQU  2          *BIT 1=1, WENN
                *                       EXAMIN
0001            LDMS    EQU  1          *BIT 0 FUER LOAD-
                *                       ADR-SCHALTER
0001            LDJA    EQU  1          *BIT 0=1, WENN
                *                       LOAD-ADR
0004            DPMS    EQU  4          *BIT 2 FUER DEPOS
0004            DPJA    EQU  4          *BIT 2=1, WENN
                *                       DEPOSIT
0006            EDMS    EQU  6          *BIT 1 UND 2 FUER
                *                       EXAM OD. DEPOS
0008            RNMS    EQU  8          *BIT 3 FUER RUN
0008            RNJA    EQU  8          *BIT 3=1, WENN RUN
0000            RNNO    EQU  0          *BIT 3=0, WENN
                *                       NICHT RUN
00E0            SYMS    EQU  X'E0'      *BITS FUER SYSTEM
                *                       SCHALTER
0020            S6JA    EQU  X'20'      *BITS, WENN SYST.6
0000            S7JA    EQU  0          *BITS, WENN SYST.7
0800            SYST7   EQU  X'0800'    *STARTADR FUER
                *                       SYSTEM 7 (ERWEI-
                *                       TERUNG

0000                    ORG  X'0000'    *SYSTEMSTARTADR

0000 31FE04 MONMP   LXI  SP,RMTP-1 *STACK-POINTER
```

MICROKIT ASSEMBLER -- VER 2.2

```
                *                       INITIALISIEREN
0003 DB04               IN    CSCH      *FU.SCH. EINLESEN
0005 C39D00             JMP   MAIN      *SPRUNG ZUM HAUPT
                *                       TEIL, UM DIE NACH
                *                       FOLGENDEN START-
                *                       ADRESSEN FUER DIE
                *                       RST-BEFEHLE FREI
                *                       ZU BEKOMMEN.
0008 C30804     RAMGO   JMP   RMRST1    *NEUES ZIEL FUER
                *                       RST 1-BEFEHL
000B 00                 NOP
000C 210004     SYST6   LXI   H,RMBS    *STARTADR FUER
                *                       "RUN" LADEN
000F E5                 PUSH  H         *FUER "RUN" NACH
                *                       RUNSUB AUF STACK
0010 F5         BREAK   PUSH  PSW       *EINSPRUNG IN MON
0011 C5                 PUSH  B         *MIT RST 2 FUER
0012 D5                 PUSH  D         *"RUN" MIT "HLT
                *                        AT BP", GESAMT-
                *                       STATUS FUER DIS-
                *                       PLAY AUF DEN STACK
                *                      (PUSH H FOLGT NOCH)
0013 0618               MVI   B,RHBP    *MASKE FUER "RUN
                *                       HLT AT BP" LADEN
                *                       (FLANKENERKENNUNG)
0015 C39903             JMP   BREAK2    *ZUR FORTSETZUNG

                *       SUBROUTINE "SRCKJP"
                *PRUEFT, OB DIE DURCH DIE "MASKE" FEST-
                *GELEGTEN BITS DES INH. DES B-REG. MIT
                *DER ZAHL "VERGL" UEBEREINSTIMMEN.
                *WENN JA, ERFOLGT EIN SPRUNG ZU "SPRADR"
                *WENN NEIN, GEHT ES IM PROGR. WEITER.
                *       AUFRUF MIT:     RST   3,SRCKJP
                *                       DC    MASKE
                *                       DC    VERGL
                *                       DC    B(SPRADR)
0018 E3         SRCKJP  XTHL
0019 F5                 PUSH  PSW
001A 78                 MOV   A,B
001B A6         FLJPRT  ANA   M
001C 23                 INX   H
001D C36300             JMP   CKJP2     *ZUR FORTS.

                *       SUBROUTINE "SCKCK1"
                *PRUEFT, OB DIE DURCH "MASKE" FESTGE-
                *LEGTEN BITS DES IHN. DES B-REG. MIT DER
                *ZAHL "VERGL" UEBEREINSTIMMEN.
                *WENN JA, WERDEN DIE 3 AUF DIESE SR
                *FOLGENDEN BYTES BEARBEITET,
                *WENN NEIN UEBERSPRUNGEN.
                *       AUFRUF MIT:     RST   4,SCKCK1
                *                       DC    MASKE
                *                       DC    VERGL
0020 E3         SCKCK1  XTHL
0021 F5                 PUSH  PSW
0022 78                 MOV   A,B
```

```
0023 A6             ANA  M
0024 23             INX  H
0025 C35000         JMP  CKSK1      *ZUR FORTS.

                    *AUF DER ZIELADRESSE DES RST 5-BEFEHLS
                    *BEGINNT EINE FUER DIE SYSTEM-SIMULATION
                    *BENOETIGTE SR
0028                SIMSR  DS   8

                    *       SUBROUTINE "SRCKSK"
                    *PRUEFT, OB DIE DURCH "MASKE" FESTGE-
                    *LEGTEN BITS DES INH. DES B-REG. MIT DER
                    *ZAHL "VERGL" UEBEREINSTIMMEN,
                    *WENN JA, WERDEN DIE AUF DIESE SR FOL-
                    *GENDEN 9 BYTES BEARBEITET,
                    *WENN NEIN, WERDEN SIE UEBERSPRUNGEN.
                    *       AUFRUF MIT:     RST  6,SRCKSK
                    *                       DC   MASKE
                    *                       DC   VERGL
0030 E3             SRCKSK XTHL
0031 F5                    PUSH PSW
0032 78                    MOV  A,B
0033 A6                    ANA  M
0034 23                    INX  H
0035 C35800                JMP  CKSK2      *ZUR FORTS.
0038 C31004         INTERU JMP  RAMINT     *NEUES ZIEL FUER
                    *                       RST 7-BEFEHL, DER
                    *                       U.A. AUCH DURCH
                    *                       EINEN HARDWARE-
                    *                       INERUPT VON DER
                    *                       HARDWARE ERZEUGT
                    *                       WIRD.

                    *       SUBROUTINE "SFLANK"
                    *PRUEFT AUF DIE L > H-FLANKE DES MIT
                    *"MASKE" FESTGELEGTEM BITS ZWISCHEN DEM
                    *ALTEN ZUSTAND (IM D-REG.) UND NEUEM
                    *ZUSTAND (IM B-REG.).
                    *WENN L > H-FLANKE, GEHT ES IM PROGRAMM
                    *WEITER,
                    *WENN  K E I N E  FLANKE, SPRUNG ZU
                    *"SPRADR"
                    *       AUFRUF MIT:     CALL SFLANK
                    *                       DC   MASKE
                    *                       DC   B(SPRADR)
003B E3             SFLANK XTHL
003C F5                    PUSH PSW
003D 7A                    MOV  A,D
003E 2F                    CMA
003F A0                    ANA  B
0040 A6             NFLEND ANA  M
0041 23                    INX  H
0042 CA6800                JZ   CNFUND
0045 C35E00                JMP  CKJPND

                    *       SUBROUTINE "NFLANK"
                    *PRUEFT AUF H > L-FLANKE DER MIT "MASKE"
```

```
                *FESTGELEGTEN BITS ZWISCHEN DEM ALTEN
                *ZUSTAND (IM D-REG.) UND NEUEM ZUSTAND
                *(IM B-REG.)
                *WENN H > L-FLANKE, GEHT ES IM PROGRAMM
                *WEITER,
                *WENN  K E I N E  FLANKE, SPRUNG ZUR
                *"SPRADR".
                *       AUFRUF MIT:     CALL NFLANK
                *                       DC   MASKE
                *                       DC   B(SPRADR)
0048 E3         NFLANK XTHL
0049 F5                PUSH PSW
004A 78                MOV  A,B
004B 2F                CMA
004C A2                ANA  D
004D C34000            JMP  NFLEND

                *       FORTS. DER SR'S
0050 BE         CKSK1  CMP  M
0051 23                INX  H
0052 CA6000            JZ   CKSEND
0055 C35F00            JMP  CKSCON

0058 BE         CKSK2  CMP  M
0059 23                INX  H
005A CA6000            JZ   CKSEND
005D 23                INX  H
005E 23         CKJPND INX  H
005F 23         CKSCON INX  H
0060 F1         CKSEND POP  PSW
0061 E3                XTHL
0062 C9                RET

0063 BE         CKJP2  CMP  M
0064 23                INX  H
0065 C25E00            JNZ  CKJPND
0068 7E         CNFUND MOV  A,M
0069 23                INX  H
006A 66                MOV  H,M
006B 6F                MOV  L,A
006C F1         CNEND  POP  PSW
006D E3                XTHL
006E C9                RET

                *       SUBROUTINE "RUNSUB"
                *PRUEFT DIE STELLUNG DER FUNKTIONS-
                *SCHALTER. ES ERFOLGT EINE VERZWEIGUNG
                *ZU "EXAM" ODER ZU "DEPOS" WENN DIESE
                *SCHALTER BETAETIGT WURDEN, ODER ZUR
                *STARTADR "RUN", WENN RUN BETAETIGT,
                *ODER ZURUECK IN DEN MONITOR ZU "DISLOP"
                *       AUFRUF MIT:     CALL RUNSUB
                *                       DC   B(EXAM)
                *                       DC   B(DEPOS)
                *                       DC   B(DISLOP)
006F E3         RUNSUB XTHL
0070 F5                PUSH PSW
```

```
0071 CD9600         CALL FLNKJP
0074 02             DC   EXMS
0075 02             DC   EXJA
0076 6800           DC   B(CNFUND)
0078 23             INX  H
0079 23             INX  H
007A CD9600         CALL FLNKJP
007D 04             DC   DPMS
007E 04             DC   DPJA
007F 6800           DC   B(CNFUND)
0081 23             INX  H
0082 23             INX  H
0083 DF             RST  3,SRCKJP
0084 08             DC   RNMS
0085 00             DC   RNNO
0086 6800           DC   B(CNFUND)
0088 DF             RST  3,SRCKJP
0089 18             DC   RHBP
008A 08             DC   RNJA
008B 5E00           DC   B(CKJPND)
008D CD3B00         CALL SFLANK
0090 08             DC   RNMS
0091 6800           DC   B(CNFUND)
0093 C35E00         JMP  CKJPND

                *       SUBROUTINE "FLNKJP"
                *PRUEFT AUF FLANKE DER DURCH "MASKE"
                *FESTGELEGTEN BITS ZWISCHEN DEM ALTEN
                *ZUSTAND (IM D-REG.) UND DEM NEUEM (IM
                *B-REG.).
                *WENN FLANKE, SPRUNG ZU "SPRADR",
                *SONST WARTET DIE SR AUF DIE FLANKE.
0096 E3     FLNKJP XTHL
0097 F5            PUSH PSW
0098 78            MOV  A,B
0099 B2            ORA  D
009A C31B00        JMP  FLJPRT


009D 50     MAIN   MOV  D,B        *ALTER ZUSTAND DES
009E 47            MOV  B,A        *B-REG. NACH D
                *                  UM AUF FLANKEN
                *                  PRUEFEN ZU KOENNEN

                *       HIER WERDEN DIE SYSTEME 1 BIS 5
                *       ENTSPRECHEND WIE FOLGT DIE SYS-
                *       TEME 6 UND 7 VON DEM SYSTEM-
                *       SCHALTER ABGEFRAGT. ES ERFOLGT
                *       EINE SPRUNGVERZWEIGUNG ZUM ENT-
                *       SPRECHENDEM SIMULATIONSPROGRAMM.
009F        SY1TO5 DS   25

00B8 DF     SY6    RST  3,SRCKJP
00B9 E0            DC   SYMS
00BA 20            DC   S6JA
00BB 0C00          DC   B(SYST6)
00BD DF     SY7    RST  3,SRCKJP
```

MICROKIT ASSEMBLER -- VER 2.2

```
00BE E0             DC    SYMS
00BF 00             DC    S7JA
00C0 0008           DC    B(SYST7)
                    *     SONST GEHT DAS PROGRAMM WEITER
                    *     ZU SYSTEM 0.
                    *     NACHFOLGENDER SPEICHERRAUM IST
                    *     MIT DEN SIMULATIONSROUTINEN DER
                    *     SYSTEME 0 BIS 5 BELEGT.
00C2                SIMPR DS    727

0399 E5       BREAK2 PUSH H           *FORTS. DER RET-
                    *                 TUNG DES STATUS
039A 210000         LXI   H,0         *CLEAR HL
039D 39             DAD   SP          *HL MIT SP LADEN
039E 22FE04         SHLD  RMTP-1      *SP IN RAMTOP-1
                    *                 ABSPEICHERN
03A1 DB02     DISLOP IN   ASCH        *STACK-OFFSET
                    *                 EINLESEN
03A3 5D             MOV   E,L         *EINGANGS-ADR
                    *                  RETTEN
03A4 21FE04         LXI   H,RMTP-1    *HL MIT SP LADEN
03A7 86             ADD   M           *SP ZUM STACK-
                    *                 OFFSET ADIEREN,
                    *                 =ECHTE ADRESSE
03A8 6F             MOV   L,A         *DIESE NACH L ALS
                    *                 LOW-ADR
03A9 7E             MOV   A,M         *INH. DIESER ADR
                    *                 (=REGISTER AUF
                    *                 STACK) LADEN
03AA D302           OUT   LLAMP       *UND ANZEIGEN
03AC DB01           IN    BSCH        *GEWAEHLTE LOW-ADR
                    *                 EINLESEN
03AE 6F             MOV   L,A         *LOW-ADR NACH L
03AF 7E             MOV   A,M         *INH. DER RAM-ADR
                    *                 LADEN
03B0 D301           OUT   RLAMP       *UND ANZEIGEN
03B2 6B             MOV   L,E         *GERETTETE EIN-
                    *                  GANGS-ADR ZURUECK
03B3 DB04     BRKLOP IN   CSCH        *FUNKTIONSSCH.
                    *                 EINLESEN
03B5 50             MOV   D,B         *ALTEN INHALT DER
                    *                 FU.SCH. NACH D
03B6 47             MOV   B,A         *NEUEN INH. DER
                    *                 FU.SCH. NACH B
                    *                 ZUR ERKENNUNG VON
                    *                 AENDERUNGEN
03B7 DB02           IN    ASCH        *ASCH FUER L-ADR
                    *                 ODER DATEN LESEN
03B9 DF             RST   3,SRCKJP    *ABFRAGE OB "LOAD
03BA 01             DC    LDMS        *ADRESS"?
03BB 01             DC    LDJA        *WENN JA,
03BC CC03           DC    B(LOADAD)   *DANN ZU LOADAD
03BE CD6F00         CALL  RUNSUB      *ABFRAGE OB
03C1 D703           DC    B(EXAMIN)   *EXAMIN, JA >>>
03C3 D003           DC    B(DEPOS)    *DEPOS, JA >>>
03C5 A103           DC    B(DISLOP)   *NICHT-RUN, JA >>>
03C7 E1       CONTIN POP  H           *WENN "RUN", DANN
```

MICROKIT ASSEMBLER -- VER 2.2

```
03C8 D1            POP  D          *GESAMTSTATUS ZU-
03C9 C1            POP  B          *RUECK UND RUECK-
03CA F1            POP  PSW        *KEHR ZUM AUFRU-
03CB C9            RET             *FENDEN ANWENDER-
                   *               PROGRAMM (RST 2 !)
03CC 6F      LOADAD MOV  L,A       *ASCH (=L-ADR)
                   *               NACH L
03CD C3D703        JMP  EXAMIN     *ZUR ANZEIGE VON
                   *               L-ADR UND DEREN
                   *               INHALTS
03D0 CD3B00  DEPOS CALL SFLANK     *H>L-FLANKE VON
03D3 04            DC   DPMS       *DEPOS-SCH SUCHEN
03D4 D703          DC   B(EXAMIN)  *WENN KEINE FLANKE
                   *               DANN ZU EXAMIN
03D6 77            MOV  M,A        *INH. ASCH (=DATA)
                   *               IN GEWAEHLTER ADR
                   *               ABSPEICHERN
03D7 7E      EXAMIN MOV  A,M       *INH. VON GEWAEHL
                   *               TER ADR HOLEN
03D8 D301          OUT  RLAMP      *INH. ANZEIGEN
03DA 7D            MOV  A,L        *L-ADR HOLEN
03DB D302          OUT  LLAMP      *L-ADR ANZEIGEN
03DD CD4800        CALL NFLANK     *H>L-FLANKE VON
03E0 06            DC   EDMS       *EXAM. OD. DEPOS
03E1 B303          DC   B(BRKLOP)  *SUCHEN, WENN
                   *               KEINE FLANKE, DANN
                   *               ZUR BRKLOP ZURUECK
03E3 2C            INR  L          *L  AUF NAECHSTE
                   *               ADRESSE (AUTOIN-
                   *               CREMENT)
03E4 C3B303        JMP  BRKLOP     *ZURUECK ZUR AB-
                   *               FRAGE WEITERER
                   *               FUNKTIONEN

                   *      IM FOLGENDEM SPEICHERRAUM LIEGT
                   *DIE MULTIPLIKATIONS-SUBROUTINE DES
                   *SYSTEMS 5. DIESE WIRD VOM SYSTEM 5
                   *IN DIE UNTEREN RAM-ADRESSEN KOPIERT UND
                   *VOM SYSTEM 5 WIE EIN ROM BENUTZT, D.H.
                   *SIE IST DORT VOM ANWENDER NICHT VER-
                   *AENDERBAR ODER ZERSTOERBAR.
03E7               MULT5 DS   25

                   ***    NO ERRORS      D.ULRICH PR/KTEW
0400               END  MONMP
```

MICROKIT ASSEMBLER -- VER 2.2

```
                ***     ITT VIDEO- UND LEHRSYSTEME
                ***     SEL AG, PFORZHEIM

                *       ITT MP-LEHRSYSTEM, EXTENSION-BOX
                *       BETRIEBSPROGRAMM: "MON 7"

                *       RESISTENT IN MASKENROM TYP
                *       AM 9214 EXP02 (512 X 8 BIT)

                *                       RP/KTEW;UL;17.1.78

                *       KONSTANTEN- UND NAMENS-TABELLE:
0002            ASCH    EQU  2          *A-SCHALTER (MP-S)
0001            BSCH    EQU  1          *B-SCHALTER (MP-S)
0004            CSCH    EQU  4          *C-SCHALTER (MP-S)
                *                       =FUNKTIONSSCHALTER
0008            HSCH    EQU  8          *SCHALTER DER EX-
                *                       BOX + CAS. INPUT
0008            HLAMP   EQU  8          *LED'S FUER H-ADR
                *                       DER EX.-BOX +
                *                       CAS. OUTPUT
0002            LLAMP   EQU  2          *LINKE LED'S(MP-S)
0001            RLAMP   EQU  1          *RECHTE LED'S (MP)
0C00            USROM   EQU  X'0C00'    *ANF. ADR DES USER
                *                       -REPROM (2708)
0064            USCODE  EQU  X'64'      *CODE AUF 1. PLATZ
                *                       DES USER-ROM, WENN
                *                       DIESES AUTOMATISCH
                *                       IN BETRIEB GEHEN
                *                       SOLL
0018            RHBP    EQU  X'18'      *BIT 3 UND 4 ALS
                *                       POSITIONSKENNUNG
                *                       DER SCHALTER "RUN"
                *                       UND "HLT AT BP"
04FF            RMTP    EQU  X'04FF'    *RAMTOP DER RAM-
                *                       PAGE 0 MIT DEM
                *                       SYSTEM-STACK
0001            LDMS    EQU  1          *BIT 0 FUER LOAD-
                *                       ADR-SCHALTER
0001            LDJA    EQU  1          *BIT 0=1, WENN
                *                       LOAD-ADR
0010            SHMS    EQU  X'10'      *BIT 4 FUER SHIFT
0010            SHJA    EQU  X'10'      *BIT 4=1, WENN
                *                       SHIFT
0040            CSMS    EQU  X'40'      *BIT 6 FUER CAS.
0040            CSJA    EQU  X'40'      *BIT 6=1, WENN CAS
006F            RUNSUB  EQU  X'006F'    *ADR DER SR IM MP
003B            SFLANK  EQU  X'003B'    *ADR DER SR IM MP
0004            DPMS    EQU  4          *BIT 2 FUER DEPOS
0048            NFLANK  EQU  X'0048'    *ADR DER SR IM MP
0006            EDMS    EQU  6          *BIT 1 UND 2 FUER
                *                       DEPOS OD. EXAMIN
0006            PGMS    EQU  6          *BIT 1 UND 2 FUER
                *                       PAGE-SCH. (EX-BOX)
0018            UPMS    EQU  X'18'      *BIT 3 UND 4 FUER
                *                       SHIFT-UP (EX-BOX)
0018            UPJA    EQU  X'18'      *BIT 3=1, BIT 4=1,
```

MICROKIT ASSEMBLER -- VER 2.2

```
                *                               WENN SHIFT-UP
0060            RDMS    EQU   X'60'            *BIT 5 UND 6 FUER
                *                               READ-CAS
0040            RDJA    EQU   X'40'            *BIT 5=0, BIT 6=1
                *                               WENN READ-CAS
0050            TSMS    EQU   X'50'            *BIT 4 UND 6 FUER
                *                               CASRUN UND SHIFT
0000            TSNO    EQU   X'00'            *BIT 4=0, BIT 6=0
                *                               WENN WEDER CASRUN,
                *                               NOCH SHIFT
0016            SYN     EQU   X'16'            *"SYN"-ZEICHEN DES
                *                               ASCII-CODES
0002            STX     EQU   X'02'            *"STX"-ZEICHEN DES
                *                               ASCII-CODES
0003            ETX     EQU   X'03'            *"ETX"-ZEICHEN DES
                *                               ASCII-CODES
FFF0            SLANG   EQU   -16              *HALBWELLENANZAHL
                *                               FUER BIT=1
FFF9            SKURZ   EQU   -7               *HALBWELLENANZAHL
                *                               FUER BIT=0
000C            SWAI    EQU   12               *ZAHL FUER HALB-
                *                               WELLENZEIT CA. 250
                *                               MICROSEC, D.H. CA.
                *                               F= 2 KHZ
0007            LZEIT   EQU   7                *KONSTANTE FUER
                *                               BIT-PRUEFZEIT

0000                    ORG   X'0800'

0800 3A000C     MON7    LDA   USROM            *1. PLATZ AUS USROM
0803 FE64               CPI   USCODE           *LADEN UND AUF INH
0805 CA000C             JZ    USROM            *USROM VERGL.,
                *                               WENN JA, DANN ZU
                *                               USROM
0808 2E00       SYS7    MVI   L,0              *L-ADR=0 FUER RUN
                *                               LADEN
080A CD8108             CALL  IND              *PAGE-SCHALTER
                *                               EINLESEN UND H MIT
                *                               HIGH-ADR FUER "RUN
                *                               LADEN
080D E5                 PUSH  H                *START-ADR "RUN"
                *                               AUF STACK
080E F5         BREAK7  PUSH  PSW              *EINSPRUNG IN MON7
080F C5                 PUSH  B                *MIT RST 1 FUER
0810 D5                 PUSH  D                *"RUN" MIT "HLT AT
0811 E5                 PUSH  H                *AT BP", GESAMT-
                *                               STATUS FUER DIS-
                *                               PLAY AUF DEN STACK
0812 0618               MVI   B,RHBP           *MASKE FUER "RUN
                *                               HLT AT BP" LADEN
0814 210000             LXI   H,0              *CLEAR HL
0817 39                 DAD   SP               *HL MIT SP LADEN
0818 22FE04             SHLD  RMTP-1           *SP IN RAMTOP-1
                *                               ABSPEICHERN
081B DB02       DISLOP  IN    ASCH             *STACK-OFFSET
                *                               EINLESEN
081D 5D                 MOV   E,L              *EINGANGS-ADR
```

MICROKIT ASSEMBLER -- VER 2.2

```
081E 4C              MOV  C,H        *RETTEN
081F 21FE04          LXI  H,RMTP-1   *HL MIT SP LADEN
0822 86              ADD  M          *SP ZUM STACK-
               *                     OFFSET ADIEREN,
               *                     =ECHTE ADRESSE
0823 6F              MOV  L,A        *DIESE NACH L ALS
               *                     LOW-ADR
0824 7E              MOV  A,M        *INH. DIESER ADR
               *                     (=REGISTER AUF
               *                     STACK) LADEN
0825 D302            OUT  LLAMP      *UND ANZEIGEN
0827 CD8108          CALL IND        *PAGE-SCH. EINLE-
               *                     SEN UND HIGH-ADR
               *                     DES GEWAEHLTEN RAM
               *                     NACH H LADEN
082A 07              RLC             *BITKORREKTUR
082B D308            OUT  HLAMP      *HI-ADR AUF EX-BOX
               *                     ANZEIGEN
082D DB01            IN   BSCH       *GEWAEHLTE LOW-ADR
               *                     EINLESEN
082F 6F              MOV  L,A        *LOW-ADR NACH L
0830 7E              MOV  A,M        *INH. DER RAM-ADR
               *                     LADEN
0831 D301            OUT  RLAMP      *UND ANZEIGEN
0833 6B              MOV  L,E        *GERETTETE EIN-
0834 61              MOV  H,C        *GANGS-ADR ZURUECK
0835 DB04    BRKLOP  IN   CSCH       *FUNKTIONSSCH.
               *                     EINLESEN
0837 50              MOV  D,B        *ALTEN INHALT DER
               *                     FU.SCH. NACH D
0838 47              MOV  B,A        *NEUEN INH. DER
               *                     FU.SCH. NACH B
               *                     ZUR ERKENNUNG VON
               *                     AENDERUNGEN
0839 DB08            IN   HSCH       *HSCH ZUR ERKENN.
               *                     DER FUNKTIONEN
               *                     PAGE, SHIFT UND
               *                     CASSETTE EINLESEN
083B 4F              MOV  C,A        *NACH C KOPIEREN
083C DB02            IN   ASCH       *ASCH FUER L-ADR
               *                     ODER DATEN LESEN
083E DF              RST  3,SRCKJP   *ABFRAGE OB "LOAD
083F 01              DC   LDMS       *ADRESS"?
0840 01              DC   LDJA       *WENN JA,
0841 5E08            DC   B(LOADAD)  *DANN ZU LOADAD
0843 58              MOV  E,B        *FU.SCH. NACH E
0844 41              MOV  B,C        *HSCH NACH B FUER
               *                     ABFRAGE
0845 DF              RST  3,SRCKJP   *ABFRAGE OB
0846 10              DC   SHMS       *"SHIFT"?
0847 10              DC   SHJA       *WENN JA,
0848 8A08            DC   B(SHIFT)   *DANN ZU SHIFT
084A DF              RST  3,SRCKJP   *ABFRAGE OB
084B 40              DC   CSMS       *"CASSETTE"?
084C 40              DC   CSJA       *WENN JA,
084D C408            DC   B(CASRUN)  *DANN ZU CASRUN
084F 43              MOV  B,E        *FU.SCH. NACH B
```

```
                *                               ZURUECK
0850 CD6F00             CALL  RUNSUB    *ABFRAGE OB
0853 6D08               DC    B(EXAMIN) *EXAMIN, JA >>>
0855 6608               DC    B(DEPOS)  *DEPOS, JA >>>
0857 1B08               DC    B(DISLOP) *NICHT-RUN, JA >>>
0859 E1                 POP   H         *WENN "RUN", DANN
085A D1                 POP   D         *GESAMTSTATUS ZU-
085B C1                 POP   B         *RUECK UND RUECK-
085C F1                 POP   PSW       *KEHR ZUM AUFRU-
085D C9                 RET             *FENDEN ANWENDER-
                *                       PROGRAMM (RST 1 !)
085E 6F         LOADAD  MOV   L,A       *ASCH (=L-ADR)
                *                       NACH L
085F 79                 MOV   A,C       *HSCH (PAGE)
                *                       NACH AC
0860 CD8308             CALL  HIAD      *H-ADR BILDEN UND
                *                       NACH H
0863 C36D08             JMP   EXAMIN    *ZUR ANZEIGE VON
                *                       L- UND H-ADR UND
                *                       DEREN INHALTS
0866 CD3B00     DEPOS   CALL  SFLANK    *H>L-FLANKE VON
0869 04                 DC    DFMS      *DEPOS-SCH SUCHEN
086A 6D08               DC    B(EXAMIN) *WENN KEINE FLANKE
                *                       DANN ZU EXAMIN
086C 77                 MOV   M,A       *INH. ASCH (=DATA)
                *                       IN GEWAEHLTER ADR
                *                       ABSPEICHERN
086D 7E         EXAMIN  MOV   A,M       *INH. VON GEWAEHL
                *                       TER ADR HOLEN
086E D301               OUT   RLAMP     *INH. ANZEIGEN
0870 7D                 MOV   A,L       *L-ADR HOLEN
0871 D302               OUT   LLAMP     *L-ADR ANZEIGEN
0873 7C                 MOV   A,H       *H-ADR HOLEN
0874 07                 RLC             *BITKORREKTUR
0875 D308               OUT   HLAMP     *H-ADR ANZEIGEN
0877 CD4800             CALL  NFLANK    *H>L-FLANKE VON
087A 06                 DC    EDMS      *EXAM. OD. DEPOS
087B 3508               DC    B(BRKLOP) *SUCHEN, WENN
                *                       KEINE FLANKE, DANN
                *                       ZUR BRKLOP ZURUECK
087D 23                 INX   H         *HL AUF NAECHSTE
                *                       ADRESSE (AUTOIN-
                *                       CREMENT)
087E C33508             JMP   BRKLOP    *ZURUECK ZUR AB-
                *                       FRAGE WEITERER
                *                       FUNKTIONEN

0881 DB08       IND     IN    HSCH      *H-SCH FUER H-ADR
0883 E606       HIAD    ANI   PGMS      *(PAGE) EINLESEN,
0885 0F                 RRC             *PAGE ISOLIEREN,
0886 C604               ADI   4         *ZUR H-ADR KORRI-
0888 67                 MOV   H,A       *GIEREN UND NACH H
0889 C9                 RET             * ZURUECK VON IND
                *                       BEZW. HIAD

088A DB01       SHIFT   IN    BSCH      *SCHRITTZAHL EIN-
088C 4F                 MOV   C,A       *LESEN UND NACH C
```

MICROKIT ASSEMBLER -- VER 2.2

```
0880 DB02          IN   ASCH      *BLOCKLAENGE EIN-
                   *               LESEN
088F DF            RST  3,SRCKJP  *ABFRAGE OB
0890 18            DC   UPMS      *"UP"(VORWAERTS)?
0891 18            DC   UPJA      *WENN JA,
0892 AC08          DC   B(FORW)   *DANN ZU FORW
0894 B7      REW   ORA  A         *SETZE C-FLAG=0
0895 47            MOV  B,A       *BLOCKZAEHLER LAD.
0896 54            MOV  D,H       *ALTE ANF.ADR NACH
0897 5D            MOV  E,L       *NACH DE
0898 7D            MOV  A,L       *SUBTRAKTION DER
0899 91            SUB  C         *SCHRITTZAHL ER-
089A 6F            MOV  L,A       *GIBT NEUE ANF.ADR
089B D29F08        JNC  RSH       *WENN ANDERE H-ADR
089E 25            DCR  H         *DANN KORREKTUR
089F 1A      RSH   LDAX D         *INH. AUS ALTER
                   *               ADR LADEN
08A0 77            MOV  M,A       *IN NEUE BRINGEN
08A1 97            SUB  A         *AC LOESCHEN (=0)
08A2 12            STAX D         *"0" IN ALTE ADR
08A3 13            INX  D         *BEIDE ADR UM 1
08A4 23            INX  H         *ERHOEHEN
08A5 05            DCR  B         *BLOCKZAEHLER -1
08A6 C29F08        JNZ  RSH       *ZUR NAECHSTEN ADR
                   *               WENN BLOCK NOCH
                   *               NICHT ZU ENDE
08A9 C3E909        JMP  SHEND     *NACH ENDE DES
                   *               BLOCKES ZU SHEND
08AC 5F      FORW  MOV  E,A       *BLOCKLAENGE > E
08AD 1D            DCR  E         *KORREKTUR
08AE 1600          MVI  D,0       *D LOESCHEN (=0)
08B0 42            MOV  B,D       *B LOESCHEN
08B1 19            DAD  D         *ALTE ANF.ADR +
                   *               BLOCKL. = ALTE END
                   *               ADR
08B2 54            MOV  D,H       *ALTE END-ADR NACH
08B3 5D            MOV  E,L       *DE
08B4 09            DAD  B         *ALTE END-ADR +
                   *               SCHRITTZAHL =
                   *               NEUE END-ADR
08B5 4F            MOV  C,A       *BLOCKZAEHLER LAD.
08B6 1A      FSH   LDAX D         *INH. AUS ALTER
08B7 77            MOV  M,A       *ADR IN NEUE
08B8 97            SUB  A         *AC MIT "0" LADEN
08B9 12            STAX D         *"0" IN ALTE ADR
08BA 1B            DCX  D         *BEIDE ADR ER-
08BB 2B            DCX  H         *NIEDRIGEN
08BC 0D            DCR  C         *BLOCKZAEHLER -1
08BD C2B608        JNZ  FSH       *ZUR NAECHSTEN ADR
                   *               WENN BLOCK NOCH
                   *               NICHT ZU ENDE
08C0 C3E909        JMP  SHEND     *WENN ZU ENDE DANN
                   *               ZU SHEND
08C3 00            NOP
08C4 DF      CASRUN RST 3,SRCKJP  *ABFRAGE OB LESEN?
08C5 60            DC   RDMS      *WENN JA,
08C6 40            DC   RDJA      *DANN ZU
```

MICROKIT ASSEMBLER -- VER 2.2

```
08C7 3909          DC   B(LBLK)   *LESEN (LBLK)
08C9 78            MOV  A,B       *HSCH HOLEN
08CA CD8308        CALL HIAD      *H-ADR BILDEN > H
08CD 2E00          MVI  L,0       *L-ADR= 0
08CF 1EFF          MVI  E,255     *BLOCKLAENGE =255
08D1 1620    SBLK  MVI  D,32      *ANZAHL DER SYN-
             *                    ZEICHEN LADEN
08D3 0E16    SYNLP MVI  C,SYN     *SYN-ZEICHEN LAD.
08D5 CDFB08        CALL SBYTE     *"SYN" SCHREIBEN
08D8 15            DCR  D         *ANZAHL -1
08D9 C2D308        JNZ  SYNLP     *NAECHSTES "SYN"
             *                    BIS ALLE RAUS
08DC 0E02          MVI  C,STX     *STX-ZEICHEN LAD.
08DE CDFB08        CALL SBYTE     *"STX" SCHREIBEN
08E1 4C            MOV  C,H       *H-ADR LADEN
08E2 CDFB08        CALL SBYTE     *H-ADR SCHREIBEN
08E5 4E            MOV  C,M       *1.DAT-BYTE LADEN
08E6 23            INX  H         *ADR AUF 2.DAT-BYT
08E7 CDFB08        CALL SBYTE     *1.DAT-BYTE SCHR.
08EA 4E      SDAT  MOV  C,M       *DATA-BYTE LADEN
08EB 23            INX  H         *ADR AUF NAECHSTES
             *                    DATA-BYTE
08EC CDFB08        CALL SBYTE     *DATA-BYTE SCHR.
08EF 1D            DCR  E         *BLOCKLAENGE -1
08F0 C2EA08        JNZ  SDAT      *NAECHSTES BYTE
             *                    SCHREIBEN BIS
             *                    BLOCK ZU ENDE
08F3 0E03          MVI  C,ETX     *ETX-ZEICHEN LADEN
08F5 CDFB08        CALL SBYTE     *"ETX" SCHREIBEN
08F8 C3E909        JMP  SHEND     *RUECKSPRUNG, WENN
             *                    FERTIG

             *     SUBROUTINES ZUM SCHREIBEN
08FB 79      SBYTE MOV  A,C       *BYTE VON C HOLEN
08FC 37            STC            *SETZE C-FLAG=1
08FD 1F      SCMX  RAR            *LSB NACH C-FLAG
08FE 4F            MOV  C,A       *REST NACH C RETT.
08FF 3EF0          MVI  A,SLANG   *CODE FUER BIT=1
             *                    LADEN
0901 DA0609        JC   STAKT     *WENN LSB=C-FLAG=
             *                    1, DANN ZU STAKT
0904 3EF9          MVI  A,SKURZ   *CODE FUER BIT=0
             *                    LADEN
0906 47      STAKT MOV  B,A       *BIT-CODE NACH B
             *                    (HALBWELLENZAEHL.)
0907 CD2E09        CALL SZEIT     *POS. HALBW. DES
             *                    TAKTES SCHREIBEN
090A CA1A09        JZ   SEND      *WENN CODE-ZAEHLER
             *                    AUF 0, DANN > SEND
090D CD2E09        CALL SZEIT     *NEG.HALBW. SCHR.
0910 C20709        JNZ  STAKT+1   *WENN CODE-Z. #0,
             *                    DANN POS.HALBW.
0913 D308          OUT  HLAMP     *AUSGABE EINER
             *                    HALBWELLE
0915 3EF9          MVI  A,SKURZ   *BITCODE 0 LADEN
0917 C31E09        JMP  SVERZ-1   *CODEZAEHLER LAD.
091A D308    SEND  OUT HLAMP      *ENDE EINER HALBW.
```

MICROKIT ASSEMBLER -- VER 2.2

```
091C 3EF0          MVI  A,SLANG   *BITCODE 1 LADEN
091E 47            MOV  B,A       *CODEZ. LADEN
091F CD3109 SVERZ  CALL ZSCH      *ZEITSCHLEIFE,
             *                     SCHREIBE "0" FUER
             *                     DEN REST DER BIT-
             *                     ZEIT AUS
0922 220008        SHLD MON7      *BEFEHL OHNE WIR-
             *                     KUNG; VERZOEGERUNG
0925 C21F09        JNZ  SVERZ     *VERZOEGERE BIS
             *                     CODEZ =0
0928 79            MOV  A,C       *BYTE-REST HOLEN
0929 B7            ORA  A         *WENN AC=0, DANN
092A C8            RZ             *RETURN
092B C3FD08        JMP  SCMX      *SONST SCHREIBE
             *                     DAS NAECHSTE BIT
092E 78      SZEIT MOV  A,B       *CODE NACH AC
092F D308          OUT  HLAMP     *SCHREIBE HALBW.
0931 3E0C    ZSCH  MVI  A,SWAI    *HALBWELLENZEIT
             *                     LADEN
0933 3D            DCR  A         *ZAEHLE ZEIT AB
0934 C23309        JNZ  ZSCH+2    *ZEITSCHLEIFE
0937 04            INR B          *CODEWORT -1
0938 C9            RET            *RUECKSPRUNG

0939 0E80    LBLK  MVI  C,X'80'   *MSB=1 SETZEN
093B 1E16          MVI  E,SYN     *SYN-ZEICHEN LADEN
093D CD8709        CALL LBIT      *1 BIT EINLESEN
0940 79            MOV  A,C       *MSB=1 NACH AC
0941 BB            CMP  E         *VERGL., OB BISHER
             *                     EINGELESENE BITS
             *                     "SYN" ERGEBEN
0942 CA4D09        JZ   LSYN+2    *WENN JA, DANN
             *                     ZUM BYTE-LESEN
0945 F601    LSTB  ORI  1         *LSB=1 ALS STOP-
             *                     BIT SETZEN
0947 4F            MOV  C,A       *LSB=1 NACH C, UM
0948 C33B09        JMP  LBLK+2    *NAECHSTES BIT ZU
             *                     LESEN
094B 1602    LSYN  MVI  D,STX     *STX-ZEICHEN LADEN
094D CD7909        CALL LBYTE     *BYTE EINLESEN
0950 79            MOV  A,C       *BYTE NACH AC
0951 BB            CMP  E         *AUF "SYN" VERGL.
0952 CA4B09        JZ   LSYN      *WENN NOCH "SYN",
             *                     DANN WEITER AUF
             *                     "STX" WARTEN
0955 BA            CMP  D         *AUF "STX" VERGL.
0956 C24509        JNZ  LSTB      *WENN NICHT "STX",
             *                     DANN SYNCHR. VER-
             *                     LOREN, ZU LSTB
0959 CD7909        CALL LBYTE     *BYTE EINLESEN
             *                     (MUSS H-ADR SEIN)
095C 61            MOV  H,C       *H-ADR NACH H
095D 2E00          MVI  L,0       *L-ADR= 0 SETZEN
095F 16FF          MVI  D,255     *BLOCK LAENGE LADEN
0961 CD7909        CALL LBYTE     *1. DATA-BYTE LESEN
0964 71            MOV  M,C       *ABSPEICHERN
0965 23            INX  H         *ADR AUF NAECHSTEN
```

```
                *                       RAM-PLATZ
0966 CD7909 LDAT    CALL LBYTE          *DATA-BYTE LESEN
0969 71             MOV  M,C            *ABSPEICHERN
096A 23             INX  H              *ADR ERHOEHEN
096B 15             DCR  D              *BLOCKZ. -1
096C C26609         JNZ  LDAT           *NAECHSTES BYTE,
                *                       WENN BLOCK ZU ENDE
096F CD7909         CALL LBYTE          *DANN LIES BYTE
0972 79             MOV  A,C            *UND VERGLEICHE,
0973 D603           SUI  ETX            *OB "ETX", WENN JA
                *                       WIRD Z-FLAG=1 UND
0975 37             STC                 *SETZE C-FLAG=1,
                *                       D.H. BLOCK RICH-
                *                       TIG GELESEN
0976 C3E909         JMP  SHEND          *FERTIG

                *       SUBROUTINEN FUER LESEN
                *       DIESE SR'S SIND NICHT MEHR EIN-
                *       ZELN KOMMENTIERT, DA DAS SEHR
                *       SCHWER VERSTAENDLICH IST.
                *SIE LESEN DIE DEMODULIERTE RECHTECK-
                *SCHWINGUNG EIN, STELLEN DEREN LAENGE
                *DURCH AUSZAEHLEN FEST, D.H. DAMIT WIRD
                *ENTSCHIEDEN, OB DAS GELESENE BIT LANGE
                *ZEIT "1" UND KURZE ZEIT "0" (BIT=1)
                *ODER KURZE ZEIT "1" UND LANGE ZEIT "0"
                *(BIT=0) IST. DIE EINZELBITS WERDEN DANN
                *ZU BYTES ZUSAMMENGESETZT UND VOM LESE-
                *HAUPTPROGRAMM ENTSPRECHEND VERARBEITET.

0979 0E80   LBYTE   MVI  C,X'80'
097B DB08           IN   HSCH
097D 0F             RRC
097E DA7B09         JC   LBYTE+2
0981 CDDD09         CALL LWAR-2
0984 DA7B09         JC   LBYTE+2
0987 CDD009 LBIT    CALL LSFL
098A 3E01           MVI  A,1
098C 47             MOV  B,A
098D 80     LZAUF   ADD  B
098E 47             MOV  B,A
098F DAE709         JC   LSTR
0992 3E08           MVI  A,LZEIT+1
0994 CDDF09 LNXT    CALL LWAR
0997 3E01           MVI  A,1
0999 DA8D09         JC   LZAUF
099C 00             NOP
099D CDDD09         CALL LWAR-2
09A0 3E02           MVI  A,2
09A2 DA8D09         JC   LZAUF
09A5 3EFE           MVI  A,-2
09A7 80     LZAB    ADD  B
09A8 47             MOV  B,A
09A9 D2CA09         JNC  LNUL
09AC 3E08           MVI  A,LZEIT+1
09AE CDDF09         CALL LWAR
09B1 3EFF           MVI  A,-1
```

MICROKIT ASSEMBLER -- VER 2.2

```
09B3 D2A709          JNC  LZAB
09B6 00              NOP
09B7 CDDD09          CALL LWAR-2
09BA 3EFE            MVI  A,-2
09BC D2A709          JNC  LZAB
09BF 79              MOV  A,C
09C0 1F              RAR
09C1 4F              MOV  C,A
09C2 D8              RC
09C3 0602            MVI  B,2
09C5 3E07            MVI  A,LZEIT
09C7 C39409          JMP  LNXT

09CA 79      LNUL    MOV  A,C
09CB 1F              RAR
09CC 4F              MOV  C,A
09CD D28709          JNC  LBIT
09D0 DB08    LSFL    IN   HSCH
09D2 0F              RRC
09D3 D2D009          JNC  LSFL
09D6 CDDD09          CALL LWAR-2
09D9 D2D009          JNC  LSFL
09DC C9              RET

09DD 3E09            MVI  A,LZEIT+2
09DF 3D      LWAR    DCR  A
09E0 C2DF09          JNZ  LWAR
09E3 DB08            IN   HSCH
09E5 0F              RRC
09E6 C9              RET

09E7 AF      LSTR    XRA  A
09E8 C9              RET

09E9 F5      SHEND   PUSH PSW        *VERZOEGERUNGS-
09EA 110000          LXI  D,0        *SCHLEIFE FUER
09ED 7E      SHLOP   MOV  A,M        *3 SEC, FUER
09EE DB08            IN   HSCH       *BESSERE BEOBACH-
09F0 47              MOV  B,A        *TUNG DER FLAGS
09F1 1B              DCX  D          *UND AUF DEUT-
09F2 7A              MOV  A,D        *LICHES
09F3 B3              ORA  E          *ERKENNEN DER BE-
09F4 C2ED09          JNZ  SHLOP      *ENDIGUNG VON
09F7 F1              POP  PSW        *"SHIFT".
09F8 DF              RST  3,SRCKJP   *ABFRAGE, OB
09F9 50              DC   TSMS       *TASTEN "SHIFT"
09FA 00              DC   TSNO       *OD. "CAS" NOCH
09FB 0E08            DC   B(BREAK7)  *GEDRUECKT SIND,
             *                       WENN JA, DANN
             *                       WEITERE 3 SEC
             *                       VERZOEGERUNG,
09FD C3E909          JMP  SHEND      *SONST ZUM MONITOR

             ***     NO ERRORS       D.ULRICH RP/KTEW
0A00                 END  MON7
```

ITT MP-Experimenter

Inhalt **Seite**

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 x 8,5 V_{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 µs 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 $0\ 0\ 0\ 0_{16}$ bis $0\ 3\ F\ F_{16}$ und dem RAM der Adreßbereich von $0\ 4\ 0\ 0_{16}$ bis $0\ 4\ F\ F_{16}$ durch den Adreßdecoder (2/6 74 LS 04, 6/6 74 LS 05 und 3/4 74 LS 32) zugeordnet. Der verbleibende Adreßbereich von $0\ 5\ 0\ 0_{16}$ bis $F\ F\ F\ F_{16}$ (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 $0\ 0\ 0\ 1_{16}$, $0\ 0\ 0\ 2_{16}$ und $0\ 0\ 0\ 4_{16}$ für die Input-Ports und $0\ 0\ 0\ 1_{16}$ und $0\ 0\ 0\ 2_{16}$ für die Output-Ports adressiert.
Da keine vollständige Adreßcodierung angewendet wurde, sondern Adreß-bit-Adressierung, stehen für weitere I/O-Adressen nur noch $0\ 0\ 0\ 8_{16}$, $0\ 0\ 1\ 0_{16}$, $0\ 0\ 2\ 0_{16}$, $0\ 0\ 4\ 0_{16}$ usw. für Inputs und $0\ 0\ 0\ 4_{16}$, $0\ 0\ 0\ 8_{16}$, $0\ 0\ 1\ 0_{16}$ 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-kΩ-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 kΩ 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 x 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 x 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 µs
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 x TTL ≙ 1,6 mA_L, 40 μA_H)

Adreßbus	AB_0 . . . AB_{15}	1 LE
Datenbus	DB_0 . . . DB_7 (bidirektional)	5 LE
Steuerbus	$\overline{\text{MEM R}}$, $\overline{\text{MEM W}}$, $\overline{\text{I/O R}}$, $\overline{\text{I/O W}}$, $\overline{\text{INTA}}$	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	$DB_0 \ldots DB_7$ (bidirektional)	1/4 LE
BUS EN	(BUS-Freigabe-Steuerung)	3 1/4 LE
$\overline{\text{HOLD}}$	(HALT-Anforderung)	3 1/2 LE
$\overline{\text{INT}}$	(Unterbrechungs-Anforderung)	3 1/2 LE
RDY IN	(Bereit-Eingang)	3 1/2 LE
$\overline{\text{RES IN}}$	(Rücksetz-Eingang)	1/4 LE (+ 1 µF)

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

$L_7 \ldots L_0$	(I/O-Port mit Adr. 0 0 0 2_{16})	10	LE
$R_7 \ldots R_0$	(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)

$C_7 \ldots C_0$	(I/O-Port mit Adr. 0 0 0 4_{16})	1	LE
$B_7 \ldots B_0$	(I/O-Port mit Adr. 0 0 0 1_{16})	1	LE
$A_7 \ldots A_0$	(I/O-Port mit Adr. 0 0 0 2_{16})	1	LE

Als Eingänge nur verwendbar, wenn $C_4 \ldots C_0$, $B_7 \ldots B_0$ und $A_7 \ldots A_0$ 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 Simulierung 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 Löschtaste 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:

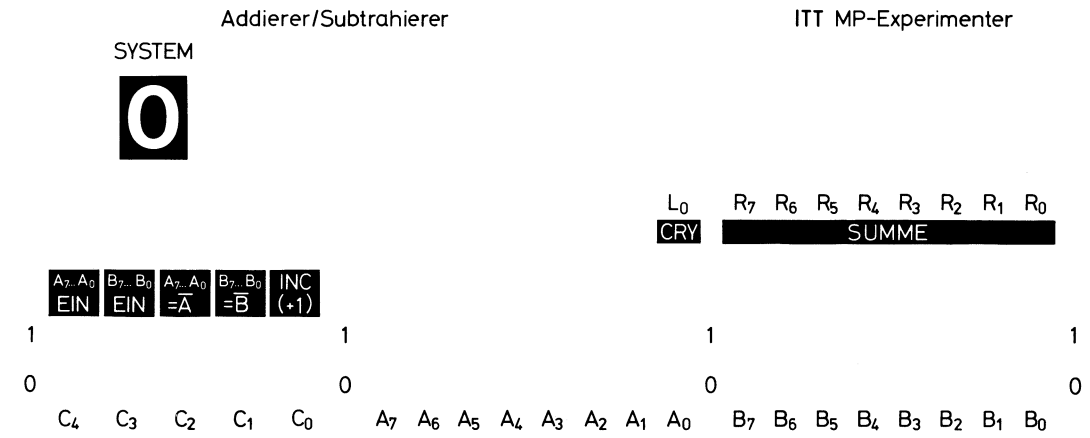
Leuchtdiode leuchtet = logisch 1
Leuchtdiode dunkel = logisch 0

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

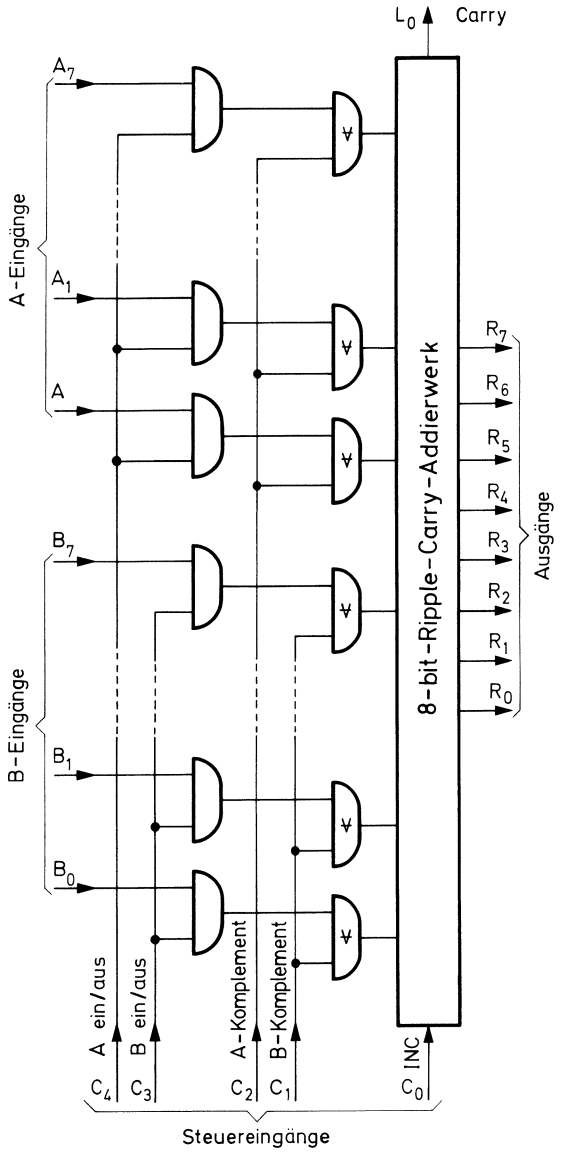
Bei falscher Schalterbetätigung können grundsätzlich keine Schäden am Experimentiersystem entstehen. Allerdings können dadurch die selbst eingegebenen Programme und Daten verändert werden, so daß ein falsches Ergebnis entsteht. Bei umfangreichen und komplizierten Experimenten kann eine falsche Betätigung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



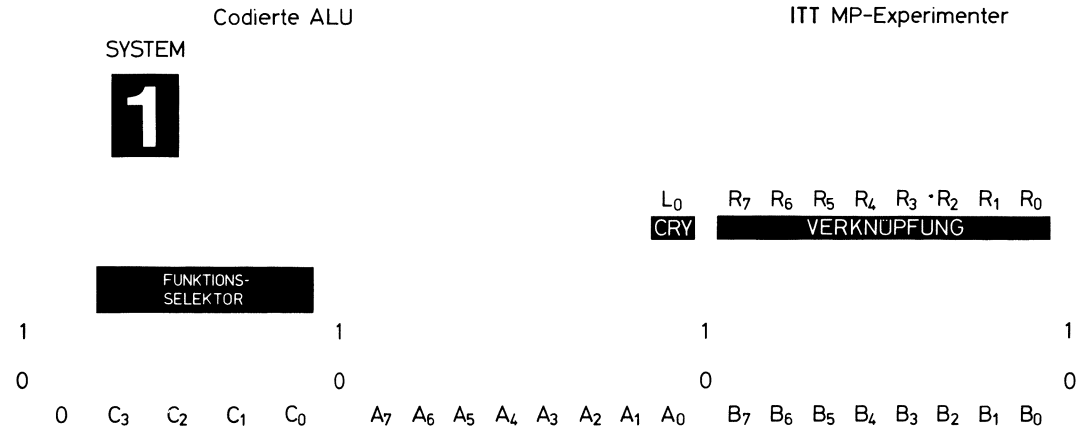
Nach der Experimentiervorbereitung Schalter C_4 und C_3 auf 1. Damit können die an den Schaltern A_7 bis A_0 und B_7 bis B_0 eingestellten Informationen in das System gelangen. Mit den Schaltern $C_2 = \overline{A}$ und $C_1 = \overline{B}$ können die eingegebenen Informationen komplementiert werden (Einerkomplement). Der Schalter C_0 = 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

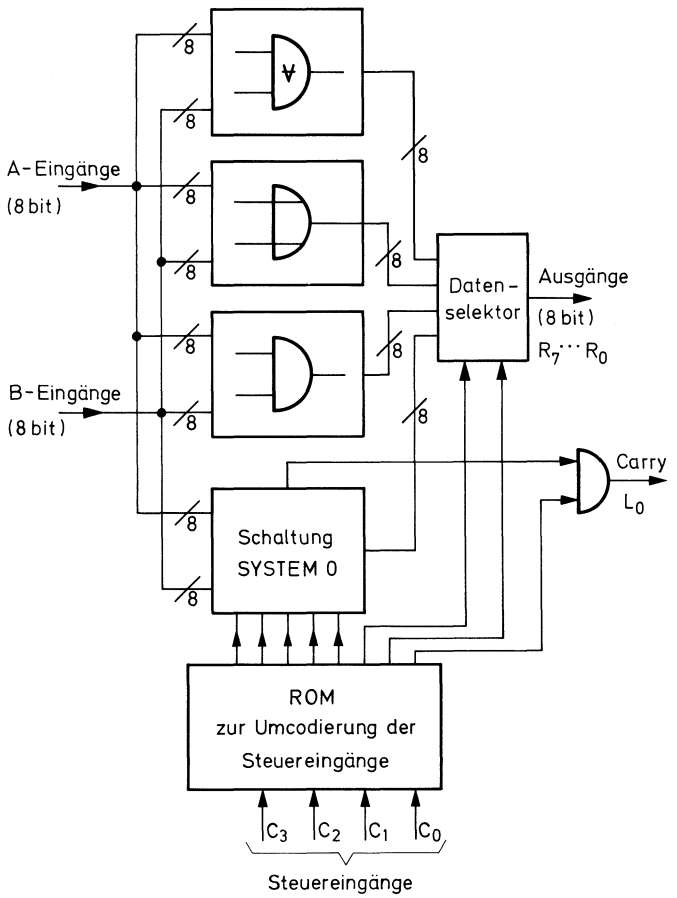
3.3 Codierte ALU (SYSTEM 1)

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.



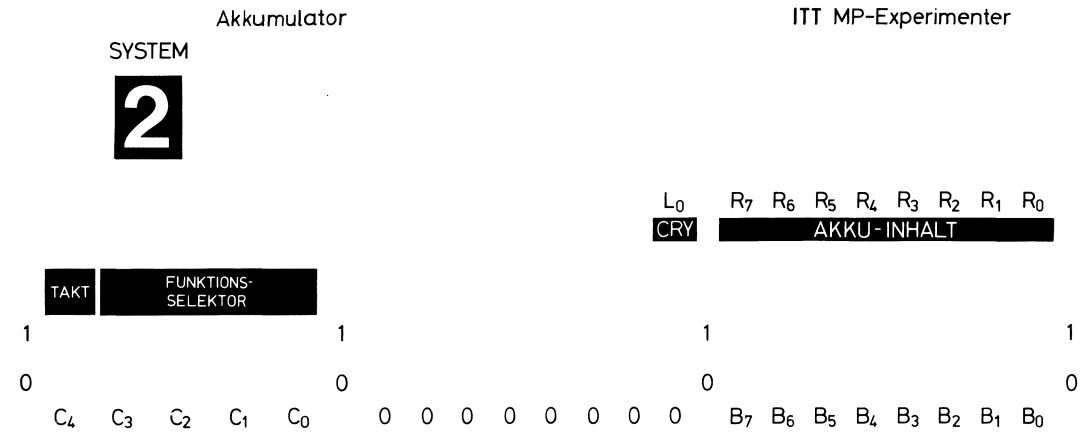
Die Funktionen werden mit den Schaltern C_3 bis C_0 festgelegt. Diese Tabelle ist auch auf der Karte „Codierte ALU“ zu finden.
Die Funktionen sind alle ausreichend bekannt und bedürfen daher keiner weiteren Erläuterung.

C_3	C_2	C_1	C_0	Funktion
0	0	0	0	A
0	0	0	1	1
0	0	1	0	$\overline{A}$
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 ∧ B
1	0	1	0	A ∨ B
1	0	1	1	A ⊽ B
1	1	0	0	−1
1	1	0	1	für Ausbau
1	1	1	0	
1	1	1	1	

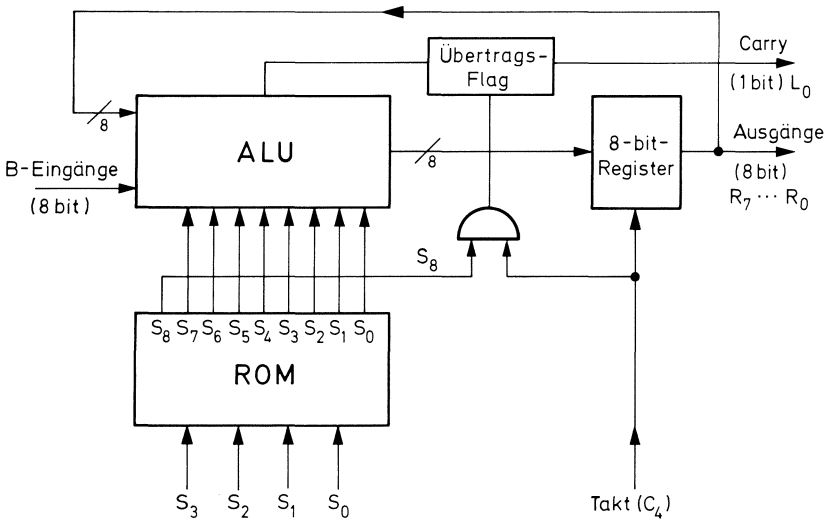


3.4 Akkumulator (SYSTEM 2)

Mit diesem Programm wird ein Akkumulator simuliert. Die in der Tabelle gezeigten Funktionen werden mit den Schaltern C_3 bis C_0 ausgewählt. Der Schalter C_4 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 R_7 bis R_0 angezeigt, ein Übertrag in L_0.



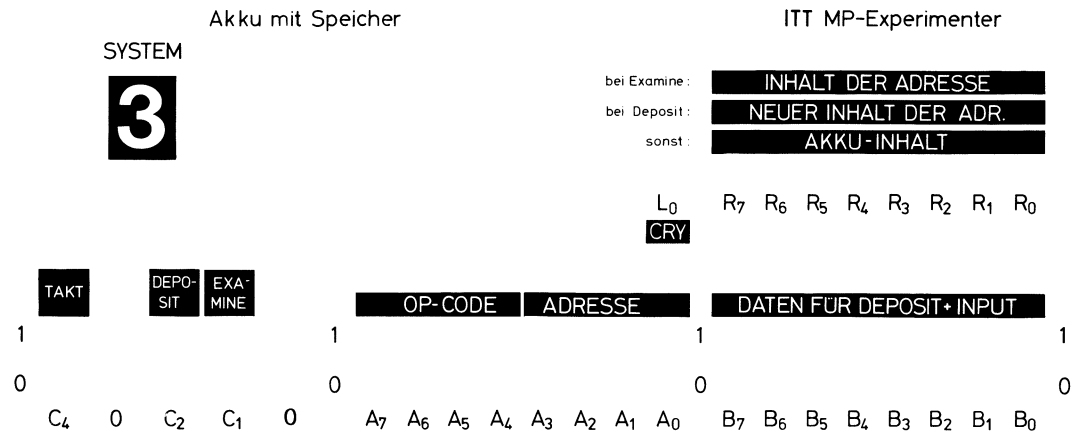
Außer den Funktionen der ALU enthält diese Schaltung als wesentliche Bestandteile noch ein Register sowie ein Übertrags-Flag. Das Register dient zum Zwischenspeichern der Ergebnisse. Hierzu wird eine der Eingangsgruppen (im Beispiel die A-Eingänge) mit den Ausgängen des Registers verbunden, so daß eine Art Rückkopplung entsteht. Jetzt werden die Informationen an den B-Eingängen mit dem Inhalt des Registers verknüpft. Durch Taktimpulse wird das Verknüpfungsergebnis in das Register geladen, wobei der alte Registerinhalt verlorengeht. Damit ein möglicher Übertrag nicht nur kurzzeitig erscheint, wird er in einem Flag ebenfalls zwischengespeichert. Ob dieses Flag getaktet wird oder nicht, wird von einem zusätzlichen bit S_8 im ROM bestimmt. Dies ist erforderlich, da ein Übertrag nur bei sinnvollen ALU-Funktionen gespeichert wird.



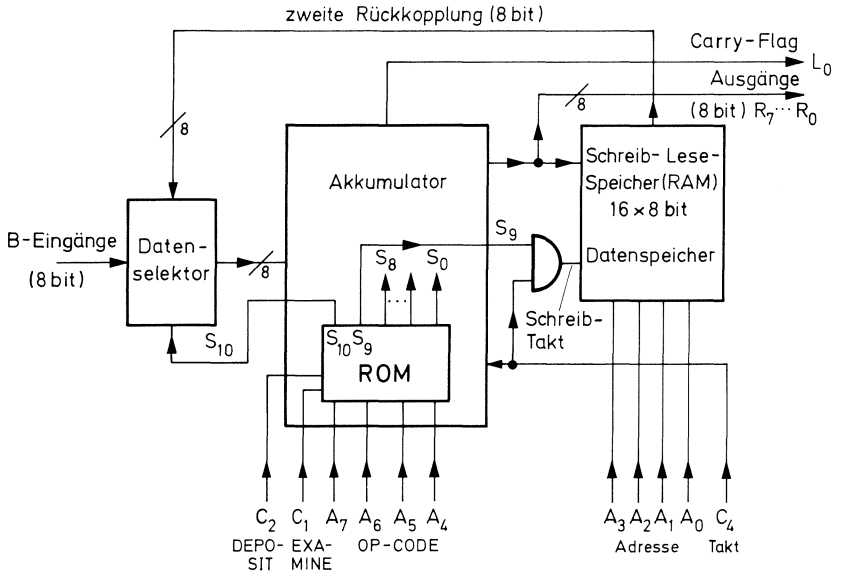
Aus der Spalte Übertrags-Flag kann entnommen werden, ob das Flag getaktet wird oder nicht. In vielen Mikroprozessoren wird dieses Flag auch bei logischen Operationen getaktet. Da hierbei aber normalerweise kein Übertrag entsteht, wird das Flag gelöscht.

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)



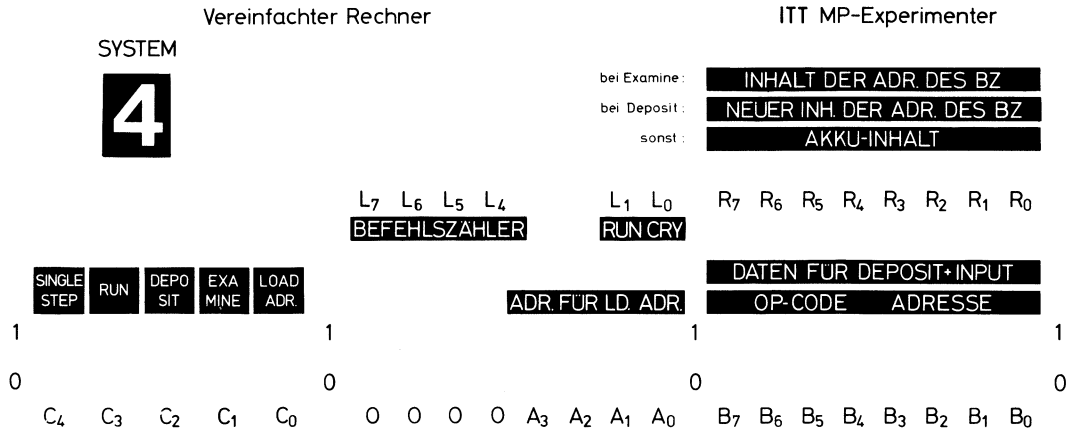
Das System 3 enthält grundsätzlich die gleichen Funktionen wie das System 2. Der Unterschied besteht darin, daß entsprechend dem Bild die Daten nicht mehr von den B-Schaltern kommen sondern von einem RAM. Mit dem neuen Befehl STA (Speichere Akku-Inhalt in Adresse a a a a ab), können die Daten in das RAM zurückgeschrieben werden. Mit dem Befehl INP (Lade B-Eingänge in den Akku) werden jetzt die Daten an B_7 bis B_0 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 C_2 DEPOSIT (Laden). Wird dieser Schalter betätigt, d.h. auf 1 und dann wieder auf 0 geschaltet, werden die Daten, die an B_7 bis B_0 liegen, im Speicher bei der Adresse abgespeichert, die von den Schaltern A_3 bis A_0 spezifiziert ist.
Mit dem Schalter C_1 EXAMINE (Abfragen) kann der Speicherinhalt, der mit A_3 bis A_0 spezifizierten Adresse in R_7 bis R_0 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)



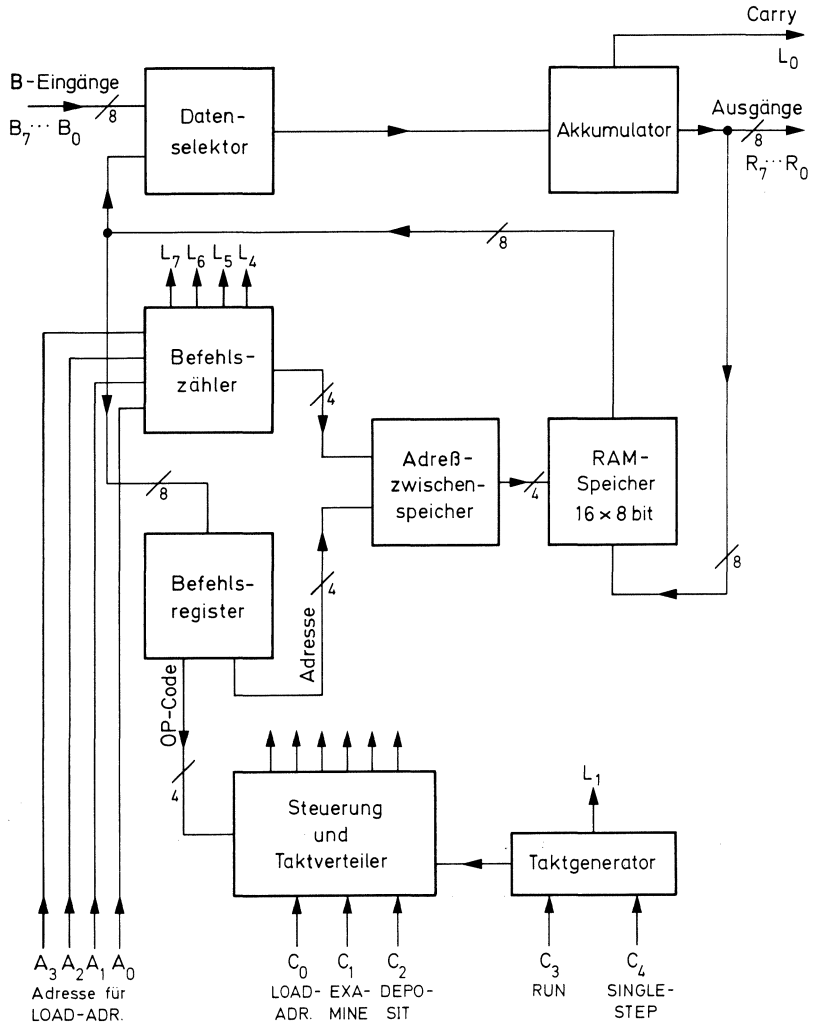
Bei diesem System wird ein vereinfachter, aber kompletter Rechner simuliert. Er hat denselben Befehlsvorrat wie der Akkumulator mit Datenspeicher im System 3. Zusätzlich hat er einen HALT-Befehl (HLT), damit der Rechner am Ende eines Programms angehalten werden kann. Im Gegensatz zum System 3 werden 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.

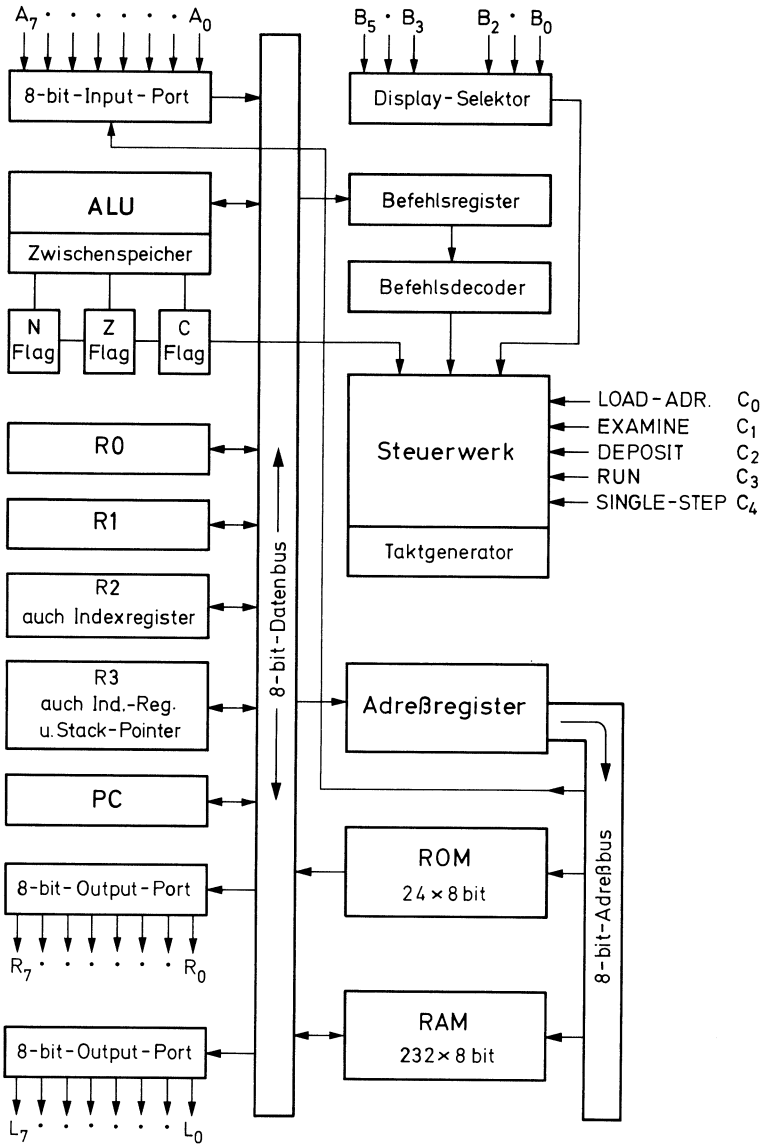
Damit der Rechner anhält, wenn das Programm abgearbeitet worden ist, muß am Ende eines Programms ein HALT-Befehl den Ablauf stoppen. Ohne diesen Befehl hätte das Programm kein Ende. Der Rechner würde auch die Daten ausführen und am Ende des Speichers wieder von vorne beginnen. Dem HALT-Befehl ist der 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 (B_7 . . . B_4) und dem Adreßteil (B_3 . . . B_0) 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 Adreßzwischenspeicher 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 (RO 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-Conditionen verwendet werden.
Der 8 bit breite Programmzähler (PC) kann $2^8 = 256$ Adressen spezifizieren. Im Adreßbereich 0 bis 24 (0 0_{16} bis 1 8_{16}) ist ein ROM untergebracht mit Betriebsprogramm. Dem Benutzer steht ein RAM mit 231 Plätzen im Adreßbereich von 25 bis 255 (1 9_{16} bis F F_{16}) zur Verfügung.
In diesen RAM-Plätzen ist bei Bedarf auch der Stack unterzubringen.
Die Adresse F 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 Kostante (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) Λ (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) Λ (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

s s/d d	Register
0 0	R0
0 1	R1
1 0	R2
1 1	R3

OP-Code-Erweiterung e e *)

e e	Befehl	Reg.-Funktion	N Z C
0 0	INCR	(s s) + 1→s s	↕ ↕ –
0 1	DECR	(s s) – 1→s s	↕ ↕ –
1 0	RACL	ROT s s u. C links	– – ↕
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

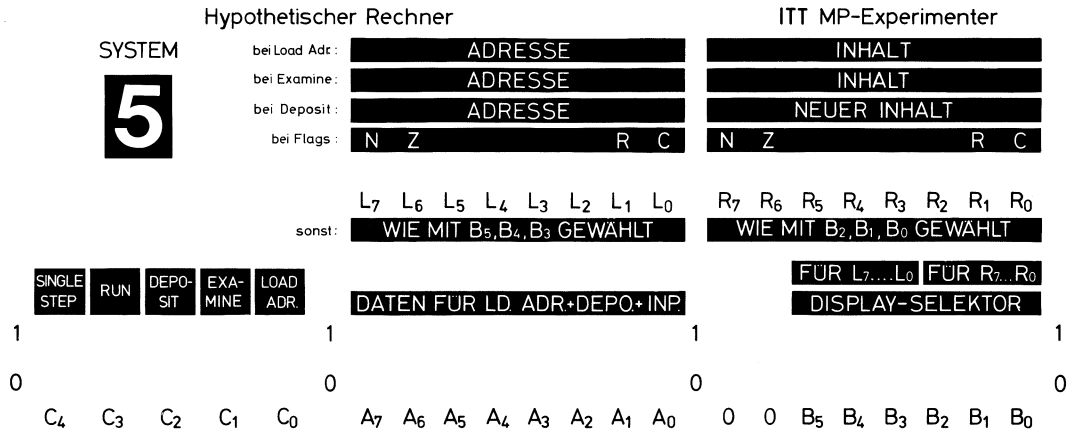
Sprung-Bedingungen c c

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-Flug = 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 Adressierungsmode 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 C_4 bis C_0 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 A_7 bis A_0 eingestellte Adresse angewählt. Solange C_0 = 1 ist, erscheint in den LEDs L_7 bis L_0 die angewählte Adresse, der Inhalt dieser Adresse wird in R_7 bis R_0 angezeigt. Das gleiche gilt für die Betätigung von EXAMINE und DEPOSIT. Bei DEPOSIT = 1 erscheint in R_7 bis R_0 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 A_5 bis A_0 abgerufen werden. Hierbei kann mit den Schaltern B_5 bis B_3 die Anzeige L_7 bis L_0 und mit den Schaltern B_2 bis B_0 die Anzeige R_7 bis R_0 angewählt werden.

Die Schalter B_7 und B_6 haben keine Bedeutung. Bei gleicher Einstellung von B_5 bis B_3 und B_2 bis B_0 erscheinen in R_7 bis R_0 und L_7 bis L_0 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 R0
0 0 1	Inhalt von R1
0 1 0	Inhalt von R2
0 1 1	Inhalt von R3
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 A_7 bis A_0 frei wählbar ist.

Wesentlich ist, daß die Schalter B_5 bis B_0 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. A_7 bis A_0 auf 4 0_{16} einstellen
3. LOAD-ADR. auf 1 stellen (nicht takten!)
In L_7 bis L_0 erscheint die an A_7 bis A_0 eingestellte Adresse, deren momentaner Inhalt in R_7 bis R_0 angezeigt wird
4. LOAD-ADR. auf 0 stellen, in L_7 bis L_0 und R_7 bis R_0 erscheint der zufällige Inhalt von R0, da mit den Schaltergruppen B_5 bis B_3 und B_2 bis B_0 jeweils der Code 0 0 0 eingestellt ist
5. Schalter B_2 bis B_0 auf 1 0 0 einstellen. In R_7 bis R_0 erscheint die Adresse 4 0_{16}, die über LOAD-ADR. geladen wurde
6. A_7 bis A_0 auf 1 0 0 0 0 1 0 0 einstellen und DEPOSIT auf 1 stellen. In L_7 bis L_0 erscheint die Adresse 4 0_{16}, in R_7 bis R_0 die an A_7 bis A_0 eingestellten Daten. Schalter DEPOSIT auf 0 zurückstellen. Bei B_2 bis B_0 = 1 0 0 erscheint jetzt in R_7 bis R_0 die nächste Adresse 4 1_{16}
7. A_7 bis A_0 auf F E_{16} einstellen und DEPOSIT takten
8. A_7 bis A_0 auf 0 1_{16} einstellen und DEPOSIT takten
9. A_7 bis A_0 auf 1 1_{16} einstellen und DEPOSIT takten
10. A_7 bis A_0 auf 0 6_{16} einstellen und DEPOSIT takten
11. A_7 bis A_0 auf 0 0_{16} einstellen und DEPOSIT takten
12. A_7 bis A_0 auf 4 0_{16} 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 0_{16} zurückgestellt. Mit EXAMINE überprüfen Sie die Eingaben:
1. Alle Schalter auf 0
2. EXAMINE auf 1 stellen. In L_7 bis L_0 erscheint die Adresse 4 0_{16}, in R_7 bis R_0 deren Inhalt 8 4_{16}, also der erste Befehl des Programms. Hierbei handelt es sich um den Befehl LOAD R0, F E_{16} (lade Akkumulator R0 mit dem Inhalt der Adresse F E_{16}). 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 R1, R0. Dieser Befehl bewirkt, daß die Daten aus R0 in R1 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 R1, R0. Hierbei werden die Daten des Registers R0 mit dem Inhalt R1 in R1 addiert
6. Punkt 5. wiederholen. In der Adresse 4 4_{16} ist der Befehl 0 6_{16}, d.h. MOVE R2, R1, abgespeichert. Hierbei werden die Daten von R1 in R2 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.

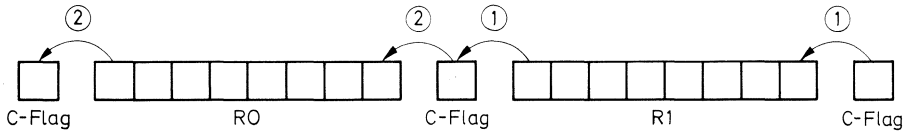
Adresse (hexadez.)	Inhalt Maschinencode	Befehl	Kommentar
4 0	1 0 0 0 0 1 0 0	LOAD R0, F E	
4 1	1 1 1 1 1 1 1 0		Datenadresse
4 2	0 0 0 0 0 0 0 1	MOVE R1, R0	
4 3	0 0 0 1 0 0 0 1	ADDR R1, R0	
4 4	0 0 0 0 0 1 1 0	MOVE R2, R1	
4 5	0 0 0 0 0 0 0 0	HLT	
.			
.			
.			
F E	0 0 0 1 0 0 0 0	Daten	

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 R0). 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 R0 erscheinen jetzt auch in R1. 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 R1 erscheinen auch in R2. 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:

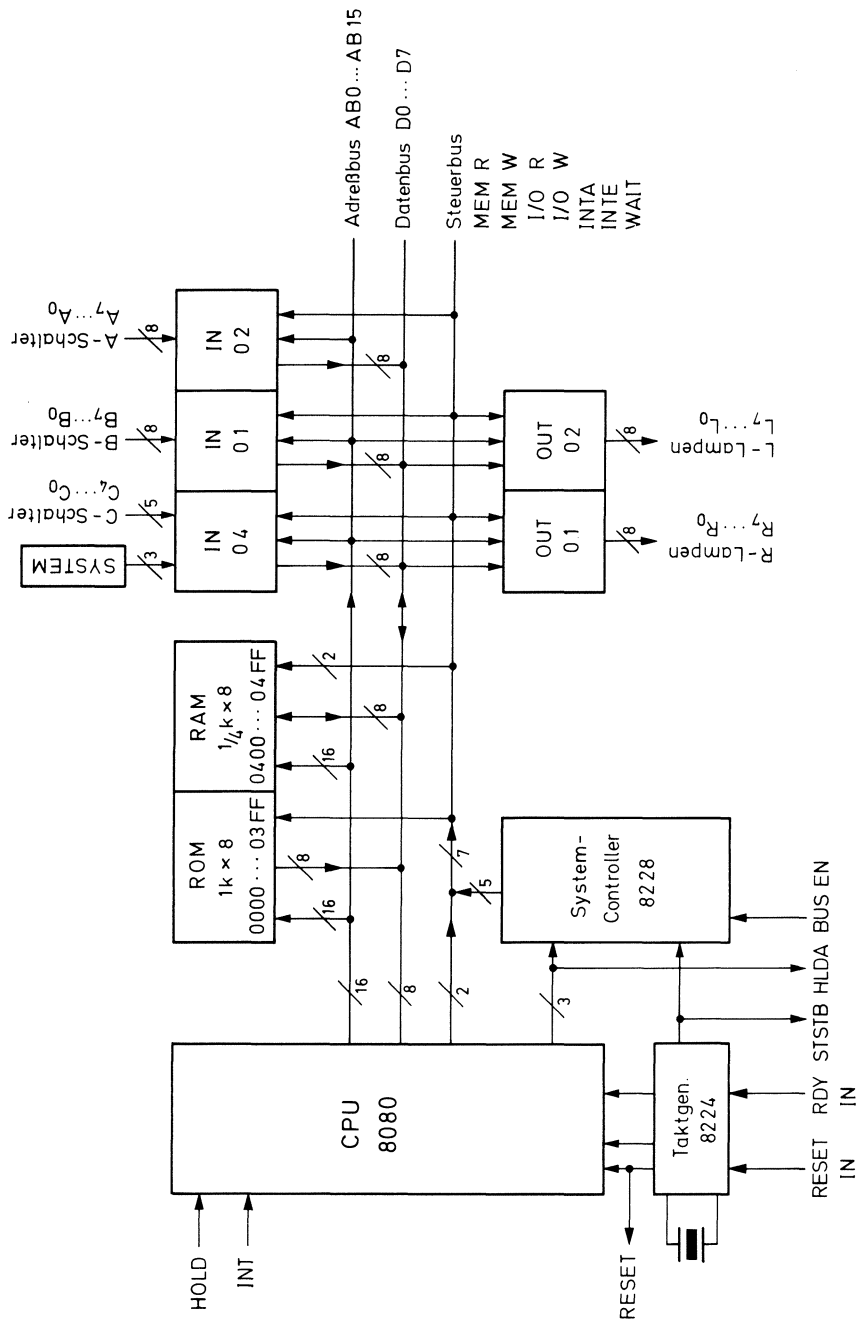


Dabei ist noch zu beachten, daß durch den Befehl RACL 2 1 der Inhalt des Carry-Flags von rechts in R1 hineingeschoben wird. Über das Programm muß deshalb sichergestellt werden, daß vor dieser Instruktion das Carry-Flag 0 enthält. Dies wird beim ersten Programmdurchlauf durch die Instruktion XORR R1, 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-	
0 2		programm	
.		im ROM-Bereich	
.			
.			
.			
.			
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 ↑	

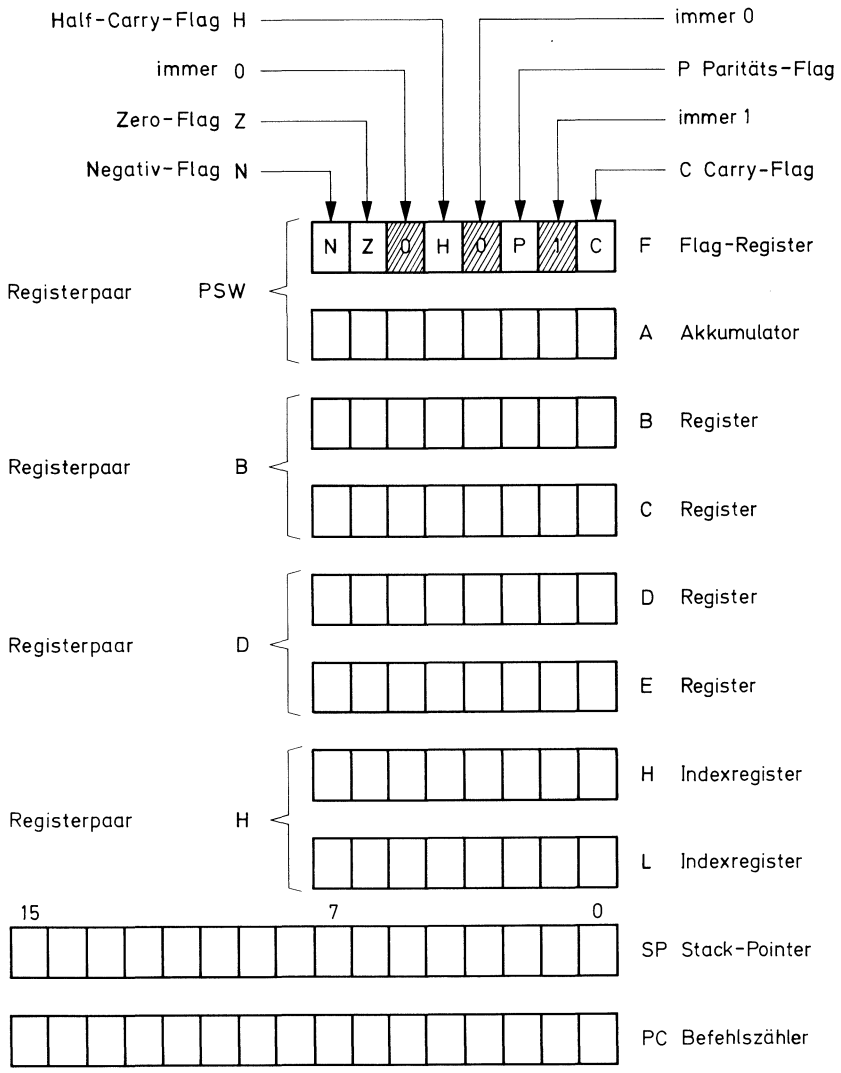
3.8 Mikrorechner-System 8080 (SYSTEM 6)
Das nachstehende Blockschaltbild zeigt den Aufbau des Mikrorechners mit der CPU 8080.



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 0_{16} bis 0 3 F F_{16} 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 0_{16} bis 0 4 F F_{16}, seine Adressen werden vom Monitor in die relativen Adressen 0 0_{16} bis F F_{16} 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, $\overline{A}$	0 0 1 0 1 1 1 1	1	($\overline{A}$) →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	($\overline{C}$) →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 dann bei POP Flags ↕
1 1 1	A		

s s s = Quellregister
d d d = Zielregister
M = Speicherplatz
↕ = Flag wird beeinflußt
– = Flag wird nicht beeinflußt
1,0 = Flag wird 1,0

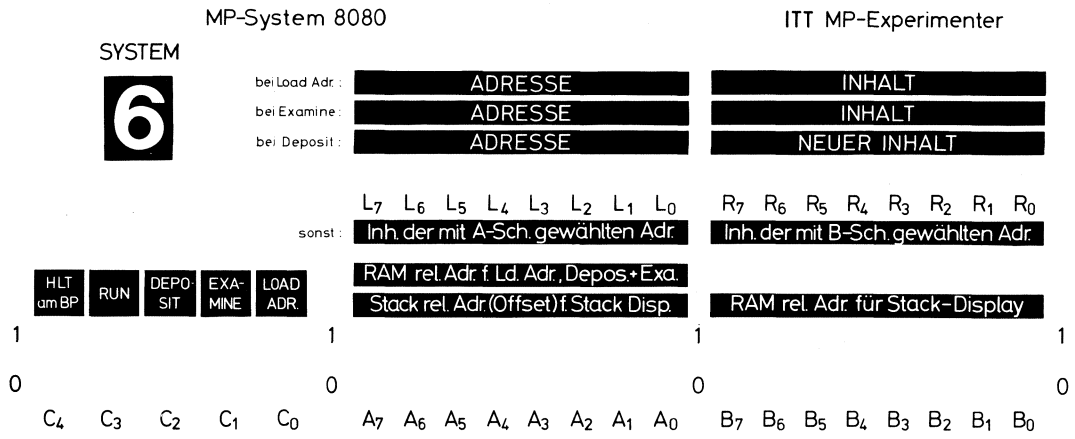
Rotate-Code (R . . .) n n		
n n	Befehl	Funktion
0 0	RLC	$(A_7) \rightarrow A_0$, $(A_7) \rightarrow C$
0 1	RRC	$(A_0) \rightarrow A_7$, $(A_0) \rightarrow C$
1 0	RAL	$(A_7) \rightarrow C$, $(C) \rightarrow A_0$
1 1	RAR	$(A_0) \rightarrow C$, $(C) \rightarrow A_7$

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 (C_3) 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.



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 0_{16} bis 0 4 F F_{16} in relative Adressen – RAM-selektive Adressen – um, die dann 0 0_{16} bis F F_{16} lauten. Die Registerinhalte werden vom Monitor in den STACK-Teil des RAMs gebracht, der die RAM-rel. Adressen F 4_{16} bis F D_{16} 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 0_{16} bis 0 9_{16} (entspricht F 4_{16} bis F D_{16}) 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 7_{16}) 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" (C_4) 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 C_4 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

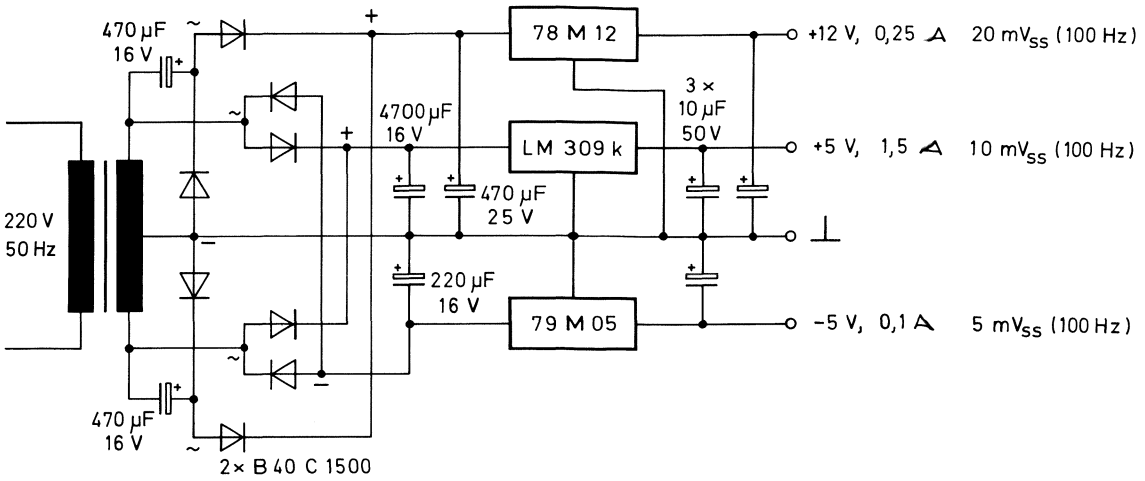


Bild 1
Stromversorgung

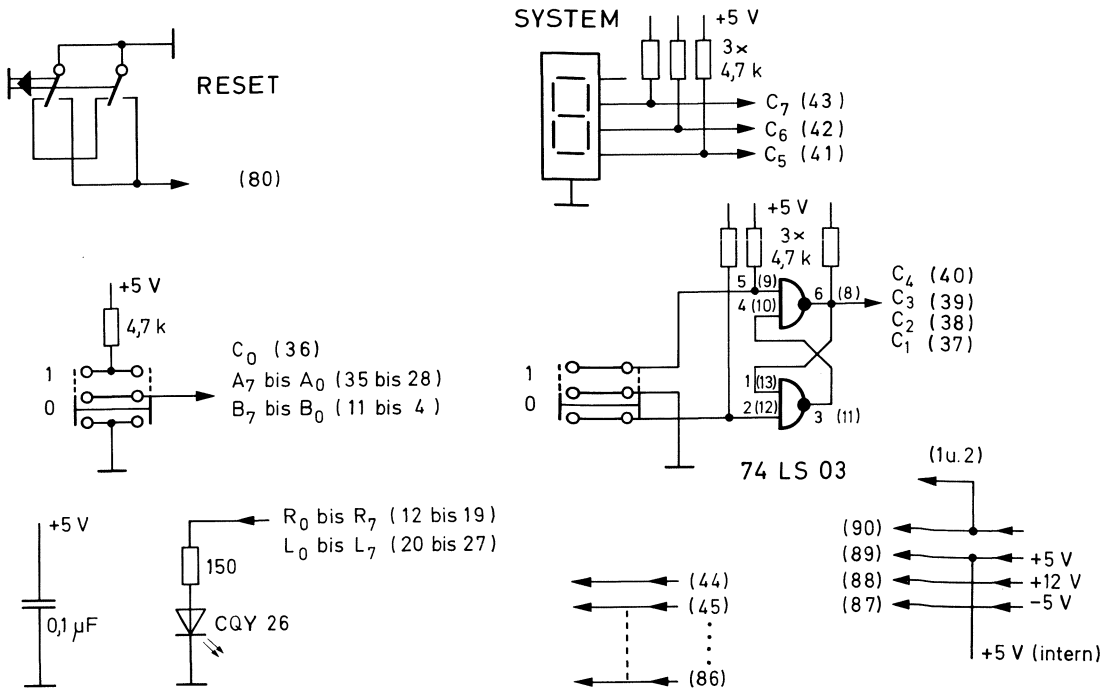
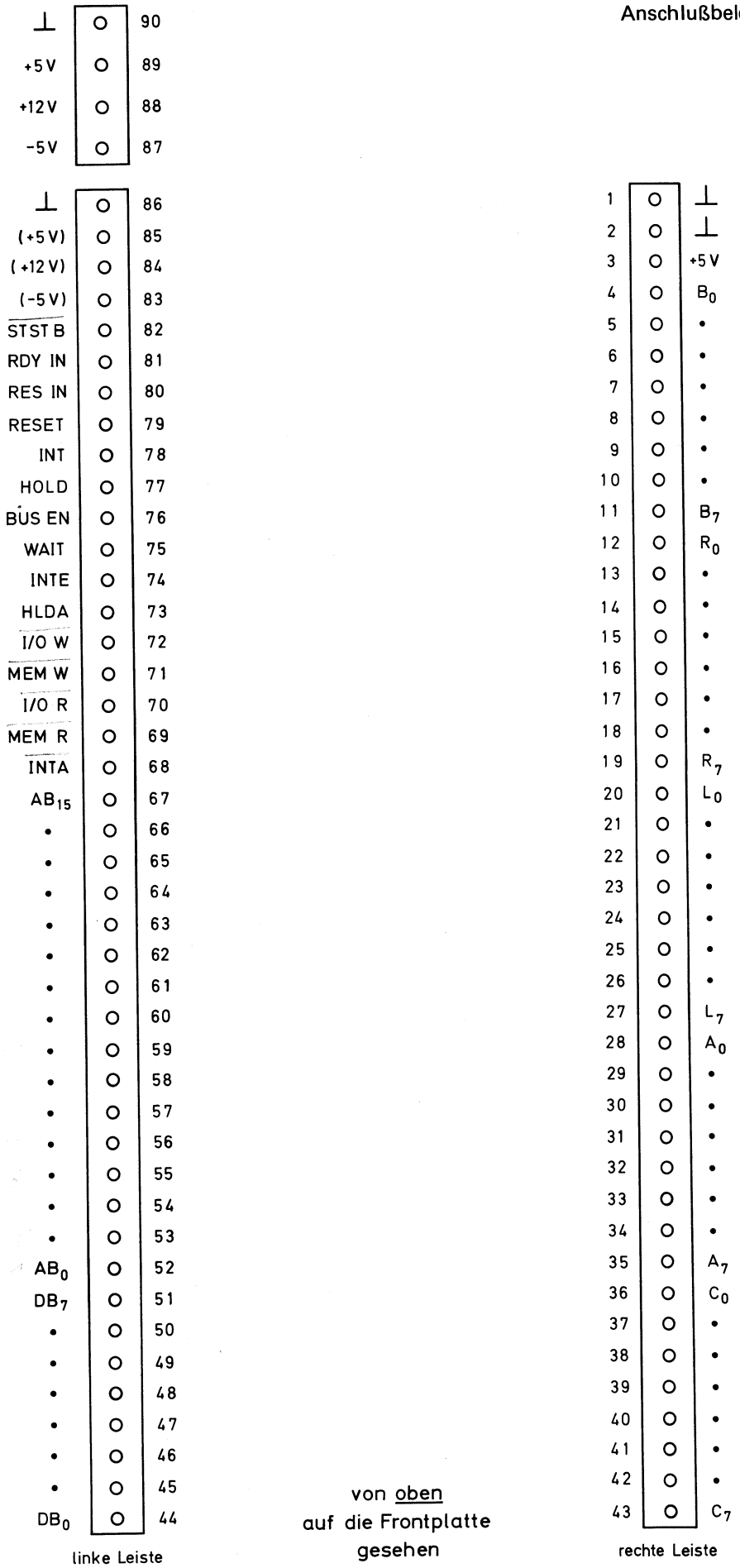
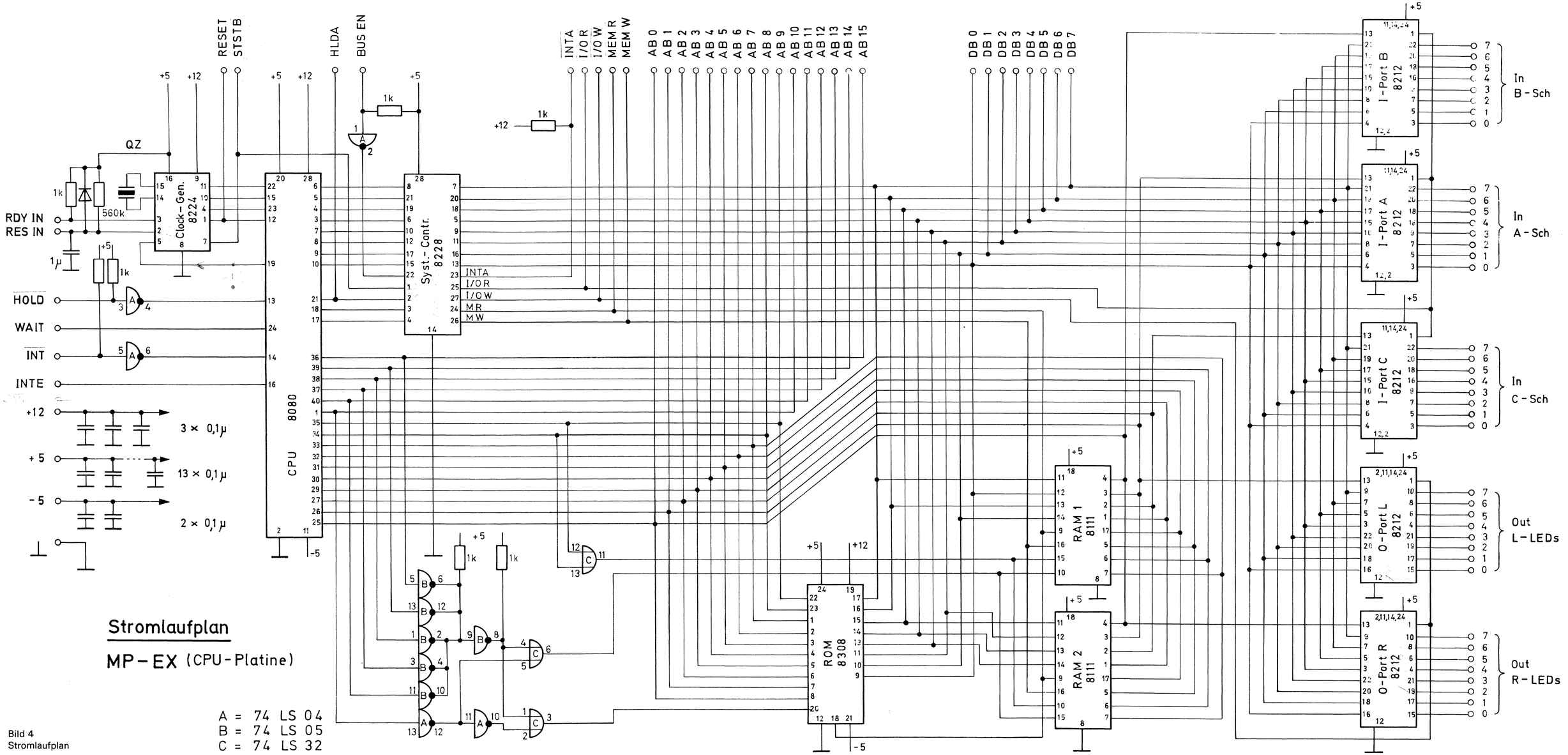


Bild 2
Frontplatte (Schaltungsauszüge)

Bild 3
Anschlußbelegung





Stromlaufplan

MP-EX (CPU-Platine)

A = 74 LS 04
B = 74 LS 05
C = 74 LS 32

Bild 4
Stromlaufplan